第十章作业

> 专业：计算机科学与技术
>
> 学号：17341178
>
> 姓名：薛伟豪

10-3

> **在7.2.5节的BP神经网络逼近算法仿真实例中，试采用遗传算法进行BP神经网络学习参数及权值的优化设计，并进行Maltab仿真。**

1. 算法流程

#1 确定神经网络的拓扑结构

#2 对神经网络权值和学习率编码，得到初始种群

（此部分为神经网络算法部分）

#3 解码得到权值和学习率

#4 将权重和学习率赋给新建的BP网络

#5 利用BP网络进行逼近

#6 计算逼近误差误差

#7 得到误差绝对值的平均值

#8 计算适应度

#9 选择适应度高的染色体进行复制

#10 进行交叉

#11 进行变异

#12 得到新群体

#13 判断检测新群体是否满足终止条件，未满足转#3，满足转#14

#14 解码

#15 得到最佳神经网络权值和学习率

遗传算法优化BP网络主要是利用遗传算法优化权值和学习率。其中，BP神经网络的拓扑结构是根据样本的输入/输出参数个数确定，这样就可以确定遗传算法优化参数的个数，从而确定种群个体的编码长度。因为遗传算法优化参数是BP神经网络的初始权值和学习率，只要网络结构已知，权值和学习率的个数就已知了。神经网络的权值和学习率一般是通过随机初始化为$[0,1]$区间的随机数，这个初始化参

数对网络训练的影响很大，但是又无法准确获得，对于相同的初始权重值和学习率，网络的训练结果是一样的，使用遗传算法就是为了优化出最佳的初始权值和学习率。

2. 神经网络设计

与7.2.5节的BP神经网络逼近算法仿真实例中的算法一致。

3. 遗传算法实现

遗传算法优化BP神经网络是用遗传算法来优化BP神经网络的初始权重值和学习率，使优化后的BP神经网络能够更好地进行逼近。遗传算法优化BP神经网络的要素包括种群初始化、适应度函数、选择算子、交叉算子和变异算子。

（1）种群初始化

个体编码使用二进制编码，每个个体均为一个二进制串，由输入层与隐含层连接权值、隐含层与输出层链接权值、学习率三部分组成，每个权值和学习率使用M位的二进制编码，将所有权值和学习率的编码连接起来即为一个个体的编码。在本题中，神经网络的结构为2-6-1结构，所以权值和学习率的个数如下表所示：

输入层与隐含层连接权值	隐含层与输出层连接权值	学习率
12	6	1

假定权值和学习率的编码均为10位二进制数，那么个体的二进制编码长度为250。其中，前120位为输入层与隐含层连接权值编码；121-180为隐含层与输出层连接权值编码；181-190学习率编码。

（2）适应度函数

为了使BP网络在逼近时，逼近值与期望值的残差尽可能小，所以选择逼近值与期望值的误差矩阵的差作为目标函数的输出。适应度函数采用排序的适应度分配函数：$FitnV = ranking(obj)$，其中obj为目标函数的输出。

（3）选择算子

选择算子采用随机遍历抽样（sus）。

（4）交叉算子

交叉算子采用最简单的单点交叉算子。

（5）变异算子

变异以一定概率产生变异基因数，用随机方法选出发生变异的基因。如果所选的基因的编码为1，则变为0；反之，则变为1。

本方案的遗传运行参数设定如下表所示：

种群大小	最大遗传代数	变异的二进制位数	交叉概率	变异概率	代沟
40	50	10	0.7	0.01	0.95

4. Matlab仿真

注：这里使用了遗传算法gatbx工具箱

- **遗传算法主函数GABPmain.m**

```
clc;
clear all;
close all;
%% 神经网络参数
% 样本数据就是前面问题描述中列出的数据
% 初始隐含层神经元个数
hiddennum = 6;
% 输入向量的最大值和最小值
threshold = [0.1;0.1];
inputnum = 2;                       % 输入层神经元个数
outputnum = 1;                      % 输出层神经元个数
w1num = 12;                         % 输入层到隐含层的权值个数
w2num = 6;                          % 隐含层到输出层的权值个数
N = w1num + w2num + 1;              % 待优化的变量个数（权值及学习速率）

%% 定义遗传算法参数
NIND = 40;                          % 种群大小
MAXGEN = 50;                        % 最大遗传代数
PRECI = 10;                         % 个体长度
GGAP = 0.95;                        % 代沟
px = 0.7;                           % 交叉概率
pm = 0.01;                          % 变异概率
trace = zeros(N+1,MAXGEN);          % 寻优结果的初始值

FieldD = [repmat(PRECI,1,N);repmat([0;1],1,N);repmat([1;0;1;1],1,N)];  % 区域描述器
Chrom = crtbp(NIND,PRECI*N);        % 创建任意离散随机种群
%% 优化
gen = 0;                            % 代计数器
X = bs2rv(Chrom,FieldD);            % 计算初始种群的十进制转换
ObjV = Objfun(X);                   % 计算目标函数值
while gen<MAXGEN
    fprintf('%d\n',gen);
    FitnV = ranking(ObjV);                      % 分配适应度值
    SelCh = select('sus',Chrom,FitnV,GGAP);     % 选择
    SelCh = recombin('xovsp',SelCh,px);         % 重组
    SelCh = recombin('xovsp',SelCh,px);         % 变异
    X = bs2rv(SelCh,FieldD);                    % 子代个体的二进制到十进制转换
    ObjVSel = Objfun(X);                        % 计算子代的目标函数值
    [Chrom,ObjV] = reins(Chrom,SelCh,1,1,ObjV,ObjVSel); % 将子代重插入父代，得到新种群
    X = bs2rv(Chrom,FieldD);
    gen = gen+1;                                % 代计数器增加
    %获取每代最优解及其序号，Y为最优解，I为个体的序号
    [Y,I] = min(ObjV);
    trace(1:N,gen) = X(I,:);                    % 记下每代的最优值
    trace(end,gen) = Y;                         % 记下每代的最优值
end
%% 画进化图
figure(1);
plot(1:MAXGEN,trace(end,:));
grid on
```

```
xlabel('遗传代数');
ylabel('误差的变化');
title('进化过程');
bestX = trace(1:end-1,end);
bestErr = trace(end,end);
fprintf(['最优初始权值和学习率：\nX = ',num2str(bestX'),'\n最小误差err =
',num2str(bestErr),'\n']);
```

- **计算种群中各个个体的目标值Objfun.m**

```
function Obj = Objfun(X)
%% 用来分别求解种群中各个个体的目标值
%% 输入
% X:所有个体的初始权值和阈值
%% 输出
% Obj:所有个体每个时刻误差的绝对值的平均数
    [M,N] = size(X);
    Obj = zeros(M,1);
    for i=1:M
        Obj(i) = Bpfun(X(i,:));
    end
end
```

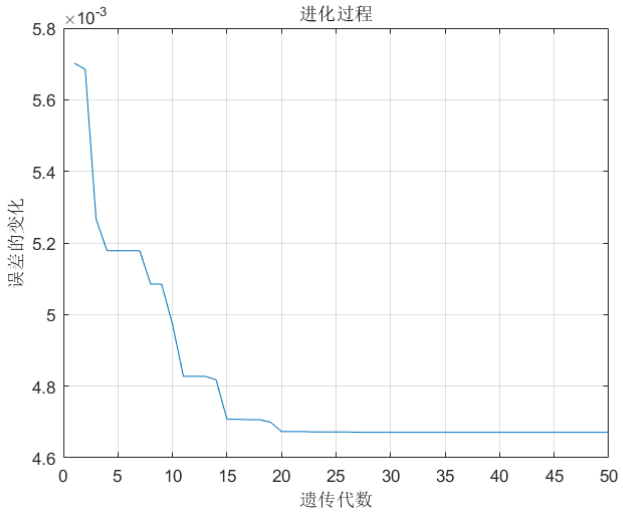
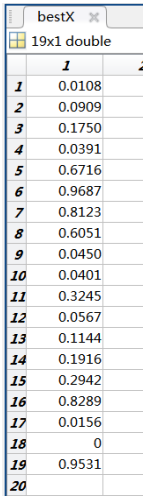
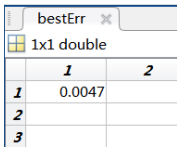
- **计算单个个体的目标值Bpfun.m**

```
function err = Bpfun(xi)
%% 训练和测试BP网络
%% 输入
    % xi:一个个体的初始权值和阈值
%% 输出
    % err:该个体每个时刻误差的绝对值的平均数
%% err计算
    xite = xi(19);
    alfa = 0.05;
    w2 = xi(13:18);
    w2 = reshape(w2,6,1);
    w2_1 = w2;
    w2_2 = w2_1;
    w1 = xi(1:12);
    w1 = reshape(w1,2,6);
    w1_1 = w1;
    w1_2 = w1;
    dw1 = 0*w1;

    x = [0,0]';
    u_1 = 0;
    y_1 = 0;
    I = [0,0,0,0,0,0]';
    Iout = [0,0,0,0,0,0]';
    FI = [0,0,0,0,0,0]';

    ts = 0.001;
    for  k = 1:1:1000
```

```
        time(k) = k*ts;
        u(k) = 0.50*sin(3*2*pi*k*ts);
        y(k) = u_1^3+y_1/(1+y_1^2);
        for  j = 1:1:6
            I(j) = x'*w1(:,j);
            Iout(j) = 1/(1+exp(-I(j)));
        end
        yn(k) = w2'*Iout;          % Output of NNI networks
        e(k) = y(k)-yn(k);    % Error calculation
        w2 = w2_1+(xite*e(k))*Iout+alfa*(w2_1-w2_2);
        for j = 1:1:6
            FI(j) = exp(-I(j))/(1+exp(-I(j)))^2;
        end
        for i = 1:1:2
            for j = 1:1:6
                dw1(i,j) = e(k)*xite*FI(j)*w2(j)*x(i);
            end
        end
        w1 = w1_1+dw1+alfa*(w1_1-w1_2);

        %%%%%%%%%%%%%%Jacobian%%%%%%%%%%%%%%%%
        yu = 0;
        for j = 1:1:6
            yu = yu+w2(j)*w1(1,j)*FI(j);
        end
        dyu(k) = yu;
        x(1) = u(k);
        x(2) = y(k);
        w1_2 = w1_1;
        w1_1 = w1;
        w2_2 = w2_1;
        w2_1 = w2;
        u_1 = u(k);
        y_1 = y(k);
    end
    err = mean(abs(y-yn));
end
```

5. 实验结果

- **遗传算法进化过程&运算结果**

进化过程	最优参数	最小平均误差
		

bestX

19x1 double

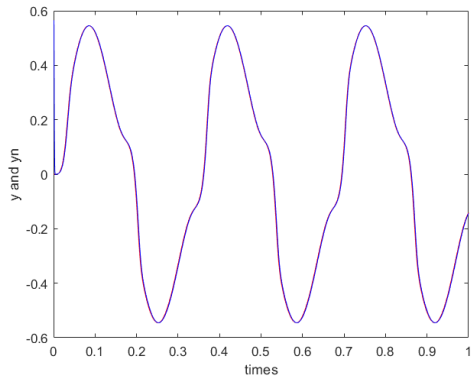
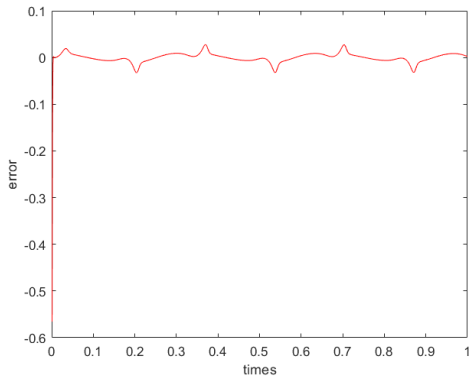
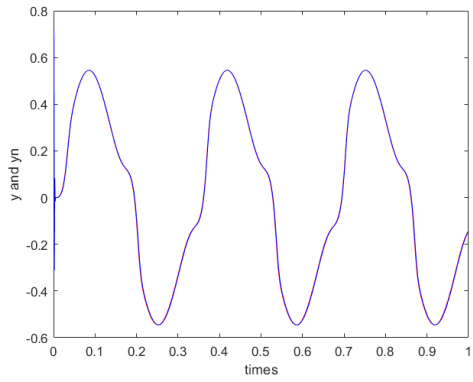
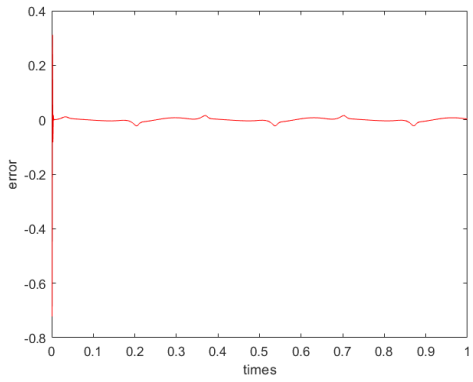
	1
1	0.0108
2	0.0909
3	0.1750
4	0.0391
5	0.6716
6	0.9687
7	0.8123
8	0.6051
9	0.0450
10	0.0401
11	0.3245
12	0.0567
13	0.1144
14	0.1916
15	0.2942
16	0.8289
17	0.0156
18	0
19	0.9531
20	

bestErr

1x1 double

	1	2
1	0.0047	
2		
3		

- **使用遗传算法前后结果比较**

未使用遗传算法时BP网络逼近效果	未使用遗传算法时BP网络逼近误差
	
使用遗传算法后BP网络逼近效果	**使用遗传算法后BP网络逼近误差**
	

可以看到，在使用遗传算法得到最优参数后，BP网络的逼近误差明显减小，逼近效果进一步提升。

10-4

> **参考2.3节专家PID控制的整定方法，对PID调节参数进行二进制编码，采用遗传算法实现PID调节参数的在线整定，试给出遗传算法设计过程，并进行Matlab仿真。**

1. 算法流程

#1 确定神经网络的拓扑结构

#2 对PID调节参数编码，得到初始种群

（此部分为专家PID控制部分）

#3 解码得到PID调节参数

#4 将PID调节参数赋给新的专家PID控制系统

#5 得到PID控制阶跃响应曲线

#6 得到误差响应曲线

#7 计算误差绝对值的平均值

#8 计算适应度

#9 选择适应度高的染色体进行复制

#10 进行交叉

#11 进行变异

#12 得到新群体

#13 判断检测新群体是否满足终止条件，未满足转#3，满足转#14

#14 解码

#15 得到最佳神经网络权值和学习率

遗传算法优化PID控制系统主要是利用遗传算法优化PID调节参数K_p、K_i、K_d。由此便可以确定遗传算法优化参数的个数，从而确定种群个体的编码长度。设定的PID调节参数对仿真结果的影响很大，但是又无法准确获得，对于相同的初始PID调节参数，系统的仿真结果是一样的，使用遗传算法就是为了优化出最佳的PID调节参数。

2. PID控制系统设计

见2.3.2节PID控制系统。

3. 遗传算法实现

遗传算法优化PID控制系统主要是利用遗传算法优化PID调节参数K_p、K_i、K_d，使优化后的PID控制系统能够得到更好的仿真结果。遗传算法优化PID控制系统的要素包括种群初始化、适应度函数、选择算子、交叉算子和变异算子。

（1）种群初始化

个体编码使用二进制编码，每个个体均为一个二进制串，由K_p、K_i、K_d三部分组成，每个权值和学习率使用M位的二进制编码，将所有权值和学习率的编码连接起来即为一个个体的编码。参数个数如下表所示：

K_p	K_i	K_d
1	1	1

假定权值和学习率的编码均为10位二进制数，那么个体的二进制编码长度为30。其中，前10位为K_p编码；11-20为K_i编码；21-30为K_d编码。

（2）适应度函数

为了使得PID控制系统在工作时，阶跃响应与期望值的残差尽可能小，所以选择阶跃响应与期望值的差作为目标函数的输出。适应度函数采用排序的适应度分配函数：$FitnV = ranking(obj)$，其中obj为目标函数的输出。

（3）选择算子

选择算子采用随机遍历抽样（sus）。

（4）交叉算子

交叉算子采用最简单的单点交叉算子。

（5）变异算子

变异以一定概率产生变异基因数，用随机方法选出发生变异的基因。如果所选的基因的编码为1，则变为0；反之，则变为1。

本方案的遗传运行参数设定如下表所示：

种群大小	最大遗传代数	变异的二进制位数	交叉概率	变异概率	代沟
40	50	10	0.7	0.01	0.95

4. Matlab仿真

注：这里使用了遗传算法gatbx工具箱

- **遗传算法主函数GABPmain.m**

```
clc;
clear all;
close all;
%% PID控制系统参数
N = 3;                          % 待优化的变量个数（kp,ki,kd）

%% 定义遗传算法参数
NIND = 40;                      % 种群大小
MAXGEN = 50;                    % 最大遗传代数
PRECI = 10;                     % 个体长度
GGAP = 0.95;                    % 代沟
```

```
px = 0.7;                          % 交叉概率
pm = 0.01;                         % 变异概率
trace = zeros(N+1,MAXGEN);         % 寻优结果的初始值

FieldD = [repmat(PRECI,1,N);repmat([0;1],1,N);repmat([1;0;1;1],1,N)];  % 区域描述器
Chrom = crtbp(NIND,PRECI*N);       % 创建任意离散随机种群
%% 优化
gen = 0;                           % 代计数器
X = bs2rv(Chrom,FieldD);           % 计算初始种群的十进制转换
ObjV = Objfun(X);                  % 计算目标函数值
while gen<MAXGEN
    fprintf('%d\n',gen);
    FitnV = ranking(ObjV);                          % 分配适应度值
    SelCh = select('sus',Chrom,FitnV,GGAP);         % 选择
    SelCh = recombin('xovsp',SelCh,px);             % 重组
    SelCh = recombin('xovsp',SelCh,px);             % 变异
    X = bs2rv(SelCh,FieldD);                        % 子代个体的二进制到十进制转换
    ObjVSel = Objfun(X);                            % 计算子代的目标函数值
    [Chrom,ObjV] = reins(Chrom,SelCh,1,1,ObjV,ObjVSel); % 将子代重插入父代，得到新种群
    X = bs2rv(Chrom,FieldD);
    gen = gen+1;                                    % 代计数器增加
    %获取每代最优解及其序号，Y为最优解，I为个体的序号
    [Y,I] = min(ObjV);
    trace(1:N,gen) = X(I,:);                        % 记下每代的最优值
    trace(end,gen) = Y;                             % 记下每代的最优值
end
%% 画进化图
figure(1);
plot(1:MAXGEN,trace(end,:));
grid on
xlabel('遗传代数');
ylabel('误差的变化');
title('进化过程');
bestX = trace(1:end-1,end);
bestErr = trace(end,end);
fprintf(['最优初始权值和学习率：\nX = ',num2str(bestX'),'\n最小误差err = ',num2str(bestErr),'\n']);
```

- **计算种群中各个个体的目标值Objfun.m**

```
function Obj = Objfun(X)
%% 用来分别求解种群中各个个体的目标值
%% 输入
% X:所有个体的初始权值和阈值
%% 输出
% Obj:所有个体每个时刻误差的绝对值的平均数
    [M,N] = size(X);
    Obj = zeros(M,1);
    for i=1:M
        Obj(i) = PIDfun(X(i,:));
    end
end
```

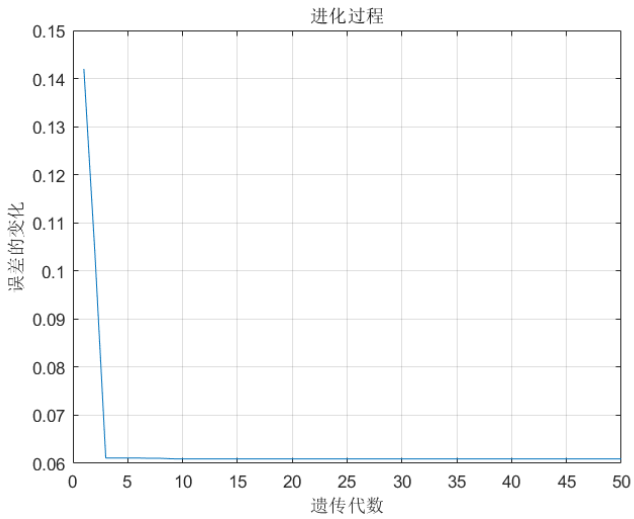
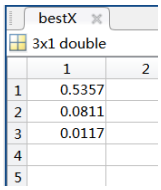
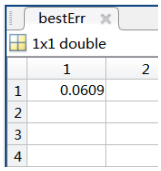
- **计算单个个体的目标值PIDfun.m**

```
function err = PIDfun(xi)
%% 训练和测试PID控制系统
%% 输入
    % xi:一个个体的初始权值和阈值
%% 输出
    % err:该个体每个时刻误差的绝对值的平均数
%% err计算
    ts = 0.001;
    sys = tf(5.235e005,[1,87.35,1.047e004,0]);  %Plant
    dsys = c2d(sys,ts,'z');
    [num,den] = tfdata(dsys,'v');

    u_1 = 0;
    u_2 = 0;
    u_3 = 0;
    y_1 = 0;
    y_2 = 0;
    y_3 = 0;
    x = [0,0,0]';
    x2_1 = 0;

    kp = xi(1);
    ki = xi(2);
    kd = xi(3);

    error_1 = 0;
    for k = 1:1:500
        time(k) = k*ts;
        r(k) = 1.0;                     %Tracing Step Signal
        u(k) = kp*x(1)+kd*x(2)+ki*x(3); %PID Controller

        %Expert control rule
        if abs(x(1))>0.8      %Rule1:Unclosed control rule
           u(k) = 0.45;
        elseif abs(x(1))>0.40
           u(k) = 0.40;
        elseif abs(x(1))>0.20
           u(k) = 0.12;
        elseif abs(x(1))>0.01
           u(k) = 0.10;
        end

        if x(1)*x(2)>0|(x(2)==0)       %Rule2
           if abs(x(1))>=0.05
              u(k) = u_1+2*kp*x(1);
           else
              u(k) = u_1+0.4*kp*x(1);
           end
        end

        if (x(1)*x(2)<0&x(2)*x2_1>0)|(x(1)==0)   %Rule3
            u(k) = u(k);
        end
```

```
54
55          if x(1)*x(2)<0&x(2)*x2_1<0   %Rule4
56             if abs(x(1))>=0.05
57                u(k) = u_1+2*kp*error_1;
58             else
59                u(k) = u_1+0.6*kp*error_1;
60             end
61          end
62
63          if abs(x(1))<=0.001   %Rule5:Integration separation PI control
64             u(k) = 0.5*x(1)+0.010*x(3);
65          end
66
67          %Restricting the output of controller
68          if u(k)>=10
69             u(k) = 10;
70          end
71          if u(k)<=-10
72             u(k) = -10;
73          end
74
75          %Linear model
76          y(k) = -den(2)*y_1-den(3)*y_2-
den(4)*y_3+num(1)*u(k)+num(2)*u_1+num(3)*u_2+num(4)*u_3;
77          error(k) = r(k)-y(k);
78          %----------Return of parameters------------%
79          u_3 = u_2;
80          u_2 = u_1;
81          u_1 = u(k);
82          y_3 = y_2;
83          y_2 = y_1;
84          y_1 = y(k);
85          x(1) = error(k);                % Calculating P
86          x2_1 = x(2);
87          x(2) = (error(k)-error_1)/ts;   % Calculating D
88          x(3) = x(3)+error(k)*ts;        % Calculating I
89          error_1 = error(k);
90      end
91      err = mean(abs(error));
92  end
```

5. 实验结果

- **遗传算法进化过程&运算结果**

进化过程	最优参数	最小平均误差
		

bestX

3x1 double

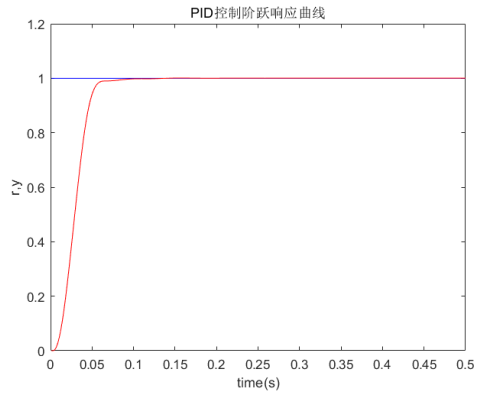
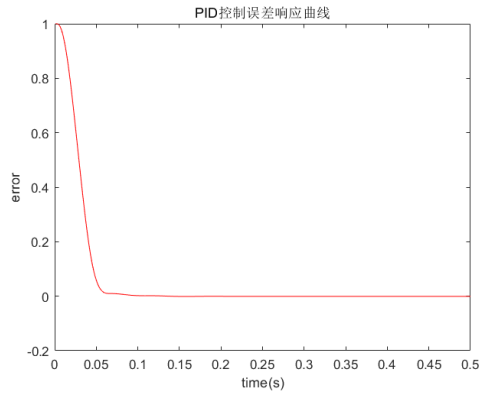
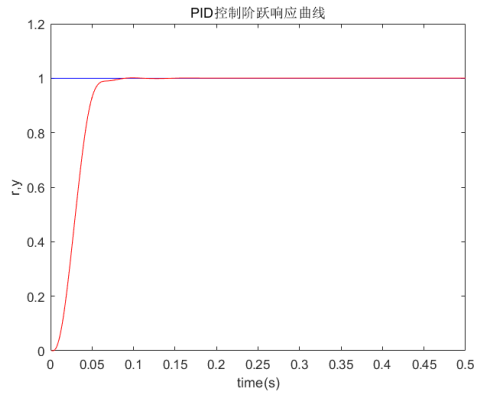
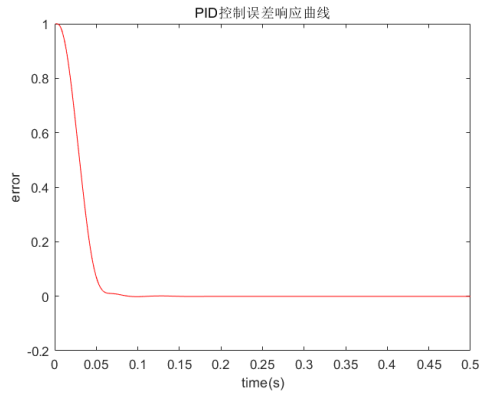
	1	2
1	0.5357	
2	0.0811	
3	0.0117	
4		
5		

bestErr

1x1 double

	1	2
1	0.0609	
2		
3		
4		

- **使用遗传算法前后结果比较**

未使用遗传算法时PID控制阶跃响应曲线	未使用遗传算法时PID控制误差响应曲线
	
使用遗传算法后PID控制阶跃响应曲线	**使用遗传算法后PID控制误差响应曲线**
	

10-5

> **分别利用粒子群算法和差分进化方法辨识如下非线性动态模型参数并进行比较分析。**

$$G(s)=\frac{K}{(T_1 s+1)(T_2 s+1)}e^{-Ts}$$

> **其中，参数真实值为$K=2$，$T_1=1$，$T_2=20$，$T=0.8$。**
>
> **试给出差分进化算法设计过程，并进行Matlab仿真。**

1. 粒子群算法

辨识集参数为$\hat{\theta}=[\hat{K}\quad \hat{T}_1\quad \hat{T}_2\quad \hat{T}]$，真实参数为$\theta=[K\quad T_1\quad T_2\quad T]=[2\quad 1\quad 20\quad 0.8]$。设待辨识参数$K$、$T_1$、$T_2$分布在$[0,30]$之间，$T$分布在$[0,1]$之间。

采用实数编码，辨识误差指标取

$$J=\sum_{i=1}^{N}\frac{1}{2}(y_i-\hat{y}_i)^T(y_i-\hat{y}_i)$$

式中，N为测试数据的数，y_i为第i个测试样本的输出。

首先运行模型测试程序model_test_main.m，对象的输入信号取伪随机二进制序列（PRBS）为输入，从而得到用于辨识的模型测试数据。

设待辨识的参数向量记为X，取粒子群个数为$Size=80$，最大迭代次数$G=100$，采用实数编码，向量X中四个参数的搜索范围为$[0,10],[0,10],[0,30],[0,3]$。粒子运动最大速度为$V_{max}=1.0$，即速度范围为$[-1,1]$。学习因子取$c_1=1.3$，$c_2=1.7$，采用线性递减的惯性权重，惯性权重采用从0.90线性递减到0.10的策略。目标函数的倒数作为粒子群的适应度函数。将辨识误差指标直接作为粒子的目标函数，越小越好。

按照式子$V_i^{kg+1}=w(t)\times V_i^{kg}+c_1r_1(p_i^{kg}-X_i^{kg})+c_2r_2(BestS_i^{kg}-X_i^{kg})$和$X_i^{kg+1}=X_i^{kg}+V_i^{kg+1}$（其中，$kg=1,2,\ldots,Size$，$r_1$和$r_2$为0到1的随机数，$c_1$为局部学习因子，$c_2$为全局学习因子，一般取$c_2$大一些）更新粒子的速度和位置，产生新种群。

具体代码如下所示：

- **model_test_main.m**

```
clear all;
close all;

n = 8;              % 移位寄存器的个数
A = 4;              % M序列的输出幅值，1->A,0->+A
ts = 0.10;          % 时钟周期（即采样时间）
N = 1000;           % 输出的个数
%Generate M-sequence
ut = genPRBS(n,A,N);
G = tf([2],[20 21 1],'inputdelay',0.8);
t = 0:ts:N*ts-ts;
y = lsim(G, ut, t);  %lsim函数求模型输出

figure(1);
stairs(t,ut,'r');
figure(2);
plot(t,y,'r');
```

```
save pso2_file t N ut y;
```

- **genPRBS.m**

```
function Out = genPRBS(n,A,N)
    Out = [];   %二进制序列初始化
    %移位寄存器初始化
    for i = 1:n
        R(i) = 1;
    end
    if R(n)==1
        Out(1) = -A;
    end
    if R(n)==0
        Out(1) = A;
    end
    for i=2:N
        temp = R(1);
        R(1) = xor(R(n-2),R(n));   %采用异或产生
        j = 2;
        while j<=n
             temp1 = R(j);
               R(j) = temp;
               j = j+1;
               temp = temp1;
        end
        if R(n)==1
            Out(i) = -A;
        end
        if R(n)==0
            Out(i) = A;
        end
    end
end
```

- **identify.m**

```
clear all;
close all;
load pso2_file;

%限定位置和速度的范围
MinX = [0 0 0 0];           %参数搜索范围
MaxX = [30 30 30 1];
Vmax = 1;
Vmin = -1;                  % 限定速度的范围

%设计粒子群参数
Size = 80;   %种群规模
CodeL = 4;   %参数个数

c1 = 1.3;
c2 = 1.7;                   % 学习因子: [1,2]
```

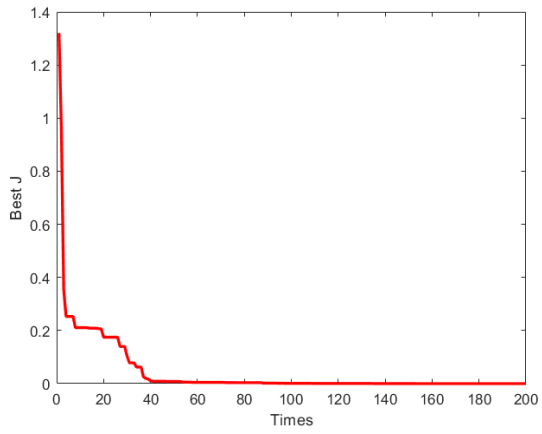
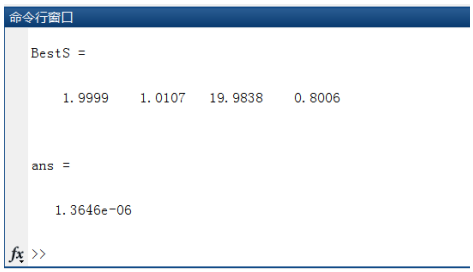
```
wmax = 0.90;
wmin = 0.10;                    % 惯性权重最小值:(0,1)
G = 200;                        % 最大迭代次数
%(1)初始化种群的个体
for i = 1:G                     %采用时变权重
    w(i) = wmax-((wmax-wmin)/G)*i;
end
for i = 1:1:CodeL               %十进制浮点制编码
    X(:,i) = MinX(i)+(MaxX(i)-MinX(i))*rand(Size,1);
    v(:,i) = Vmin+(Vmax-Vmin)*rand(Size,1);%随机初始化速度
end
%（2）初始化个体最优和全局最优：先计算各个粒子的目标函数，并初始Ji和BestS
for i = 1:Size
    Ji(i) = obj(X(i,:),t,N,ut,y);
    Xl(i,:) = X(i,:);           %Xl用于局部优化
end

BestS = X(1,:);                 %全局最优个体初始化
for i = 2:Size
    if obj(X(i,:),t,N,ut,y)<obj(BestS,t,N,ut,y)
        BestS = X(i,:);
    end
end
%（3）进入主要循环，直到满足精度要求
 for kg = 1:1:G
     times(kg) = kg;
    for i = 1:Size
       v(i,:) = w(kg)*v(i,:)+c1*rand*(Xl(i,:)-X(i,:))+c2*rand*(BestS-
X(i,:));%加权，实现速度的更新
          for j = 1:CodeL       %检查速度是否越界
            if v(i,j)<Vmin
                v(i,j) = Vmin;
            elseif  v(i,j)>Vmax
                v(i,j) = Vmax;
            end
          end
        X(i,:) = X(i,:)+v(i,:); %实现位置的更新
        for j = 1:CodeL         %检查位置是否越界
            if X(i,j)<MinX(j)
                X(i,j) = MinX(j);
            elseif X(i,j)>MaxX(j)
                X(i,j) = MaxX(j);
            end
        end
%自适应变异,避免陷入局部最优
       if rand>0.8
            k = ceil(4*rand);   %ceil为向上取整
            X(i,k) = 30*rand;
            if k ==4
                X(i,k) = rand;
            end
       end
%（4）判断和更新
       if obj(X(i,:),t,N,ut,y)<Ji(i) %局部优化：判断当此时的位置是否为最优的情况
          Ji(i) = obj(X(i,:),t,N,ut,y);
          Xl(i,:) = X(i,:);
       end

```

```
        if Ji(i)<obj(BestS,t,N,ut,y)  %全局优化
          BestS = Xl(i,:);
        end
    end
Best_J(kg) = obj(BestS,t,N,ut,y);
end
display('true value: K = 2;T1 = 1;T2 = 20;T = 0.8');

BestS           %最佳个体
Best_J(kg)      %最佳目标函数值
figure(1);      %目标函数值变化曲线
plot(times,Best_J(times),'r','linewidth',2);
xlabel('Times');ylabel('Best J');
```

- **obj.m**

```
function J = obj(X,t,N,ut,y)
%计算个体目标函数值
    Kp = X(1);
    T1p = X(2);
    T2p = X(3);
    Tp = X(4);
    %%%%%%%%%%%%%%%%
    Gp = tf([Kp],[T2p T1p+T2p 1],'inputdelay',Tp);
    yp = lsim(Gp,ut,t);
    E = yp-y;
    J = 0;
    for i = 1:1:N
        J = J+0.5*E(i)*E(i);
    end
end
```

辨识误差函数J的优化过程和辨识结果如下所示：

辨识函数误差J的优化过程	辨识结果
	

辨识结果为$\hat{X}=[1.9999 \quad 1.0107 \quad 19.9838 \quad 0.8006]$，最终的辨识误差指标为$J=1.3646\times10^{-6}$。

2. 差分进化算法

辨识集参数为$\hat{\theta}=[\hat{K}\quad \hat{T_1}\quad \hat{T_2}\quad \hat{T}]$，真实参数为$\theta=[K\quad T_1\quad T_2\quad T]=[2\quad 1\quad 20\quad 0.8]$。设待辨识参数$K$、$T_1$、$T_2$分布在$[0,30]$之间，$T$分布在$[0,1]$之间。

采用实数编码，辨识误差指标取

$$J=\sum_{i=1}^{N}\frac{1}{2}(y_i-\hat{y}_i)^T(y_i-\hat{y}_i)$$

式中，N为测试数据的数，y_i为第i个测试样本的输出。

首先运行模型测试程序model_test_main.m，对象的输入信号取伪随机二进制序列（PRBS）为输入，从而得到用于辨识的模型测试数据。

设待辨识的参数向量记为X，取粒子群个数为$Size=80$，最大迭代次数$G=100$，采用实数编码，向量X中四个参数的搜索范围为$[0,10],[0,10],[0,30],[0,3]$。在差分进化算法仿真中，取$F=0.95$，$CR=0.60$。按照以下步骤设计差分进化算法：

（1）生成初始群体

在n维空间里随机产生满足约束条件的M个个体，实施措施如下：

$$x_{ij}(0)=rand_{ij}(0,1)(x_{ij}^U-x_{ij}^L)+x_{ij}^L$$

式中，x_{ij}^U和x_{ij}^L分别是第j个染色体的上界和下界，$rand_{ij}(0,1)$是$[0,1]$之间的随机小数。

（2）变异操作

在群体中随机选择3个个体x_{p1}，x_{p2}和x_{p3}，且$i\neq p_1\neq p_2\neq p_3$，则基本的变异操作为

$$h_{ij}(t+1)=x_{p_1j}(t)+F(x_{p_2j}(t)-x_{p_3j}(t))$$

如果无局部优化问题，变异操作可写为

$$h_{ij}(t+1)=x_{bj}(t)+F(x_{p_2j}(t)-x_{p_3j}(t))$$

式中，$x_{p_2j}(t)-x_{p_3j}(t)$为差异化向量，此差分操作时差分进化算法的关键，F为缩放因子，p_1，p_2，p_3为随机整数，表示个体在种群中的序号，$x_{bj}(t)$为当前代中种群中最好的个体。由于上式借鉴了当前种群中最好的个体信息，可加快收敛速度。

（3）交叉操作

交叉操作是为了增加群体的多样性，具体操作如下：

$$v_{ij}(t+1)=\begin{cases}h_{ij}(t+1), & rand\ l_{ij}\leq CR\\ x_{ij}(t), & rand\ lij>CR\end{cases}$$

式中，$rand\ l_{ij}$为$[0,1]$之间的随机小数，CR为交叉概率，$CR\in[0,1]$。

（4）选择操作

为了确定$x_i(t)$是否成为下一代的成员，试验向量$v_i(t+1)$和目标向量$x_i(t)$对评价函数进行比较：

$$x_i(t+1)=\begin{cases}v_i(t+1), & f(v_{i1}(t+1),\ \dots,v_{in}(t+1)>f(x_{i1},\ \dots,x_{in}(t))\\ x_{ij}(t), & f(v_{i1}(t+1),\ \dots,v_{in}(t+1)\leq f(x_{i1},\ \dots,x_{in}(t))\end{cases}$$

反复执行步骤（2）至步骤（4）操作，直至达到最大的进化代数G，差分进化基本运算流程如下所示：

#1 初始化种群，DE算法参数

#2 计算群体中每个个体适应度

#3 判断是否满足终止条件，满足则得到最优结果，不满足则转#4

#4 变异操作

#5 交叉操作

#6 选择操作

#7 判断是否达到最大的进化代数，如果达到则结束，否则转#2

具体代码如下所示：

- **model_test_main.m**

```
clear all;
close all;

n = 8;            % 移位寄存器的个数
A = 4;            % M序列的输出幅值，1->A,0->+A
ts = 0.10;        % 时钟周期（即采样时间）
N = 1000;         % 输出的个数
%Generate M-sequence
ut = genPRBS(n,A,N);
G = tf([2],[20 21 1],'inputdelay',0.8);
t = 0:ts:N*ts-ts;
y = lsim(G, ut, t);  %lsim函数求模型输出

figure(1);
stairs(t,ut,'r');
figure(2);
plot(t,y,'r');
save de2_file t N ut y;
```

- **genPRBS.m**

```
function Out = genPRBS(n,A,N)
    Out = [];   %二进制序列初始化
    %移位寄存器初始化
    for i = 1:n
        R(i) = 1;
    end
    if R(n)==1
        Out(1) = -A;
    end
    if R(n)==0
        Out(1) = A;
    end
    for i=2:N
        temp = R(1);
        R(1) = xor(R(n-2),R(n));   %采用异或产生
        j = 2;
        while j<=n
             temp1 = R(j);
               R(j) = temp;
               j = j+1;
```

```
                temp = temp1;
        end
        if R(n)==1
            Out(i) = -A;
        end
        if R(n)==0
            Out(i) = A;
        end
    end
end
```

- **identify.m**

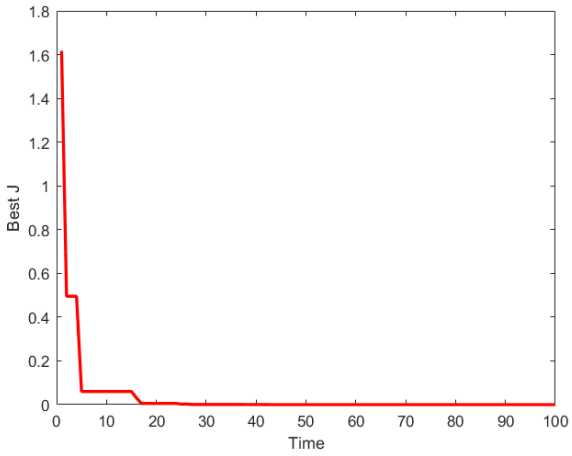
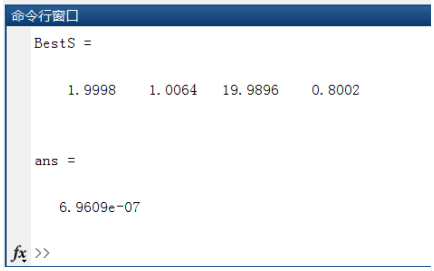
```
clear all;
close all;
load de2_file;

%限定位置和速度的范围
MinX = [0 0 0 0];        %参数搜索范围
MaxX = [10 10 30 3];

%设计粒子群参数
Size = 30;               %种群规模
CodeL = 4;               %参数个数

F = 0.95;                % 变异因子：[1,2]
cr = 0.6;                % 交叉因子
G = 100;                 % 最大迭代次数
%初始化种群的个体
for i=1:1:CodeL
    X(:,i) = MinX(i)+(MaxX(i)-MinX(i))*rand(Size,1);
end
BestS = X(1,:);          %全局最优个体
for i=2:Size
    if obj(X(i,:),t,N,ut,y)<obj(BestS,t,N,ut,y)
        BestS = X(i,:);
    end
end
Ji = obj(BestS,t,N,ut,y);
%进入主要循环，直到满足精度要求
for kg=1:1:G
     time(kg) = kg;
%变异
    for i=1:Size
        r1 = 1;
        r2 = 1;
        r3 = 1;
        r4 = 1;
        while r1==r2 || r1==r3 || r2==r3 || r1==i || r2==i || r3==i ||
r4==i || r1==r4 || r2==r4 || r3==r4
            r1 = ceil(Size * rand(1));
            r2 = ceil(Size * rand(1));
            r3 = ceil(Size * rand(1));
            r4 = ceil(Size * rand(1));
        end
```

```
        h(i,:) = BestS+F*(X(r2,:)-X(r3,:));
        %h(i,:) = X(r1,:)+F*(X(r2,:)-X(r3,:));

       for j=1:CodeL  %检查值是否越界
            if h(i,j)<MinX(j)
                h(i,j) = MinX(j);
            elseif h(i,j)>MaxX(j)
                h(i,j) = MaxX(j);
            end
        end
%交叉
        for j = 1:1:CodeL
              tempr = rand(1);
              if tempr<cr
                  v(i,j) = h(i,j);
               else
                  v(i,j) = X(i,j);
               end
        end
%选择
        if obj(v(i,:),t,N,ut,y)<obj(X(i,:),t,N,ut,y)
            X(i,:) = v(i,:);
        end
%判断和更新
       if obj(X(i,:),t,N,ut,y)<Ji %判断当此时的指标是否为最优的情况
          Ji = obj(X(i,:),t,N,ut,y);
          BestS = X(i,:);
        end
    end
    Best_J(kg) = obj(BestS,t,N,ut,y);
end
display('true value: K=2;T1=1;T2=20;T=0.8');

BestS    %最佳个体
Best_J(kg)%最佳目标函数值
figure(1);%指标函数值变化曲线
plot(time,Best_J(time),'r','linewidth',2);
xlabel('Time');ylabel('Best J');
```

- **obj.m**

```
function J = obj(X,t,N,ut,y)%*********计算个体目标函数值
    Kp = X(1);
    T1p = X(2);
    T2p = X(3);
    Tp = X(4);
    %%%%%%%%%%%%%%%%
    Gp = tf([Kp],[T2p T1p+T2p 1],'inputdelay',Tp);
    yp = lsim(Gp,ut,t);
    E = yp-y;
    J = 0;
    for i = 1:1:N
        J = J+0.5*E(i)*E(i);
    end
end
```

辨识误差函数J的优化过程和辨识结果如下所示：

辨识函数误差J的优化过程	辨识结果
	

辨识结果为$\hat{X} = [1.9998 \quad 1.0064 \quad 19.9896 \quad 0.8002]$，最终的辨识误差指标为$J = 6.9609 \times 10^{-7}$。

从两个算法的仿真结果可以看出，辨识本题中非线性动态模型参数，差分进化算法的效果要优于粒子群算法。