中国石油大学（华东）
CHINA UNIVERSITY OF PETROLEUM

系统辨识大作业

班级：自动化 1605 班

姓名：韩祖成

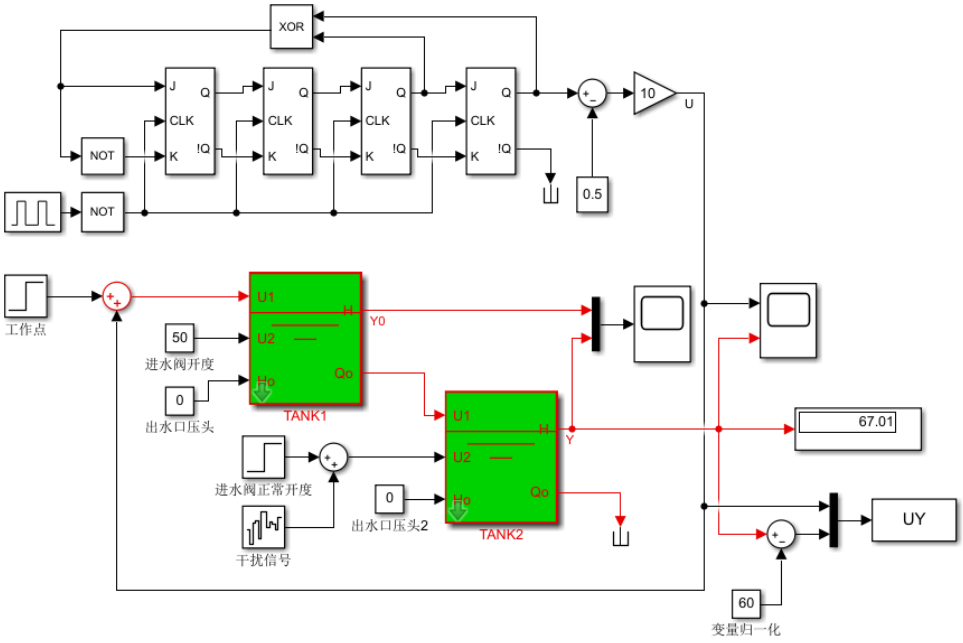
学号：1605010409

日期：2019 年 12 月 1 日

第一题　模仿 index2，搭建对象

（1）由相关分析法，获得脉冲响应序列 $\hat{g}(k)$，由 $\hat{g}(k)$，参照讲义，获得系统的脉冲传递函数 $G(z)$ 和传递函数 $G(s)$；

模仿 index2，搭建类似二阶水箱的模型对象，该对象的传递函数为二阶，并且微调了相关参数。



在进行编程之前，要明确辨识时所用到的系统辨识的方法，简单介绍一下编程过程中所用到的公式，简单分析推导过程。

$$R_{yM}(\tau)=\sum_{j=0}^{N-1}\hat{g}(j)R_M(t-j)\Delta t$$

$$\mathbf{g}=\begin{bmatrix} g(0) \\ g(1) \\ \vdots \\ g(N-1) \end{bmatrix},\qquad \mathbf{R}_{yM}=\begin{bmatrix} R_{yM}(0) \\ R_{yM}(1) \\ \vdots \\ R_{yM}(N-1) \end{bmatrix}$$

$$\mathbf{R}_{yM}=\mathbf{R}\mathbf{g}\Delta t$$

$$g=\frac{1}{\Delta t}\mathbf{R}^{-1}\mathbf{R}_{yM}$$

由上述公式推导得到脉冲响应函数 $\hat{g}(k)$

$$\mathbf{g}=\frac{N}{a^2(N+1)\Delta t}\begin{bmatrix}2 & 1 & \Lambda & 1\\ 1 & 2 & \Lambda & 1\\ \mathrm{M} & \mathrm{M} & & \mathrm{M}\\ 1 & 1 & \Lambda & 2\end{bmatrix}\mathbf{R}_{\mathbf{y}M}$$

其中，Rym 可以化简为

$$\mathbf{Y}=\begin{bmatrix}y(0)\\ y(1)\\ \vdots\\ y(N-1)\end{bmatrix},\qquad \mathbf{R}_{\mathbf{y}M}=\begin{bmatrix}R_{yM}(0)\\ R_{yM}(1)\\ \vdots\\ R_{yM}(N-1)\end{bmatrix}$$

$$M=\begin{bmatrix}M(0) & M(1) & \cdots & M(N-1)\\ M(-1) & M(0) & \cdots & M(N-2)\\ \vdots & \vdots & & \vdots\\ M(-N+1) & M(-N+2) & \cdots & M(0)\end{bmatrix}$$

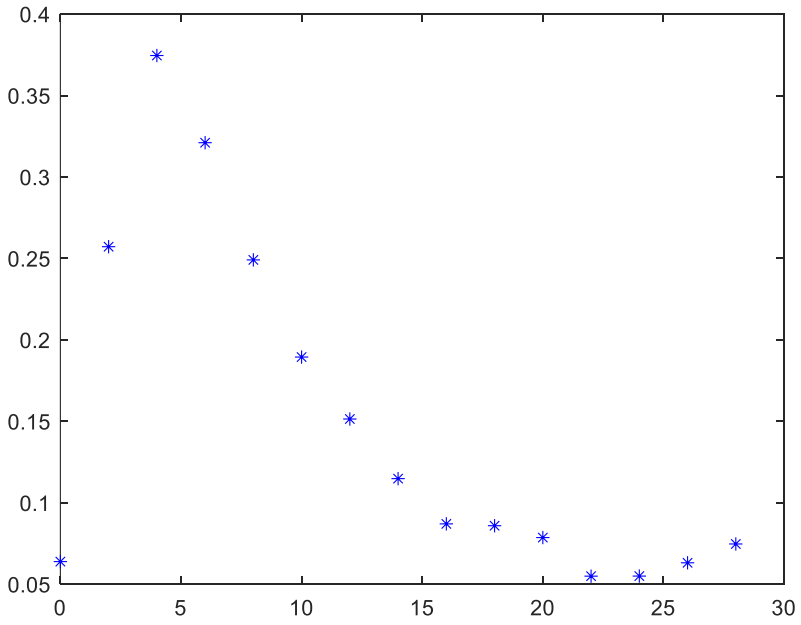
$$\mathbf{R}_{\mathbf{y}M}=\frac{1}{N}M\mathbf{Y}$$

相关程序编写如下

```
% 相关分析法获得脉冲响应序列
N = 2^4-1;
deta = 2;  %deta 为脉冲的周期
a = 5;  %a 为脉冲正输出值
M = zeros(N,N);
for i=1:1:N
    for j=1:1:N
        M(i,j) = UY(N-i+j,1); %给矩阵 M 赋值
    end
end
Y = zeros(N,1);
for i=1:1:N
    Y(i,1)=UY(N+i-1,2); %给矩阵 Y 赋值
end
R = ones(N,N);
for i = 1:1:N
    R(i,i) = 2;      %给矩阵 Rym 赋值
end
G = (1/(a*a*(N+1)*deta))*R*M*Y;  %这是相关分析法辨识脉冲响应的最重要公式
t = 0:2:28;  %由于周期为 2，故按 2 进位
plot(t',G,'b*');   %绘制响应曲线
%有脉冲响应转换为传递函数 由于矩阵阶数为 2，水箱为 2 阶水箱，故 n=2
```

```
g_tem=[G(2,1) G(3,1);G(3,1) G(4,1)];
a_bianshi=zeros(2,1);b_bianshi=zeros(2,1);%a，b 参数辨识
tem=[-G(4,1);-G(5,1)];
a_bianshi=inv(g_tem)*tem;  %a 参数辨识
 b_bianshi=[a_bianshi(1,1) 1 ;-a_bianshi(2,1) 0]*[G(2);G(3)];
% b_bianshi=[0 a_bianshi(1,1) 1;a_bianshi(1,1) a_bianshi(2,1)
0]*[G(1);G(2);G(3)];
tem=a_bianshi(1);
a_bianshi(1)=a_bianshi(2); %a 参数位置转换
a_bianshi(2)=tem;
G1=tf([b_bianshi],[1,a_bianshi],2)
Gs = d2c(G1,'zoh')
```

根据老师 ppt 上的内容，我进行了相关分析法的编程，并得到了脉冲响应序列 $\hat{g}(k)$，并按照 ppt 上的公式求得脉冲传递函数 G(z)，利用 Matlab 自带的公式求得传递函数 G(s)。相关结果见下面的截图。



上图为脉冲响应序列 g(k)，需要说明的是，由于系统存在一定高斯白噪声的干扰，所以导致高阶的脉冲响应序列出现了升高的趋势。经过编程计算，我们求解出传递函数连续和离散的形式，结果见程序结果截图。

```
G1 =

    0.4231 z + 0.2508
  ----------------------
  z^2 - 1.006 z + 0.2219

Sample time: 2 seconds
Discrete-time transfer function.


Gs =

    0.003662 s + 0.3375
  ------------------------
  s^2 + 0.7528 s + 0.1082

Continuous-time transfer function.
```

（2）应用最小二乘辨识,获得脉冲响应序列$\hat{g}(k)$;同图显示两种方法的辨识效果图；

同样的，首先参考 ppt 上公式，明确编程思路。在分析推导编程所用公式的时候，我发现 ppt 上最小二乘辨识的脉冲响应序列的公式存在错误，重新推导以后在编程中进行了修改，得到了正确的结果。对比两种辨识方式，两种方式辨识所得的脉冲响应序列完全相同，这证明我们所求的脉冲响应序列是正确的。

$$Z_l(k) = H_l(k)g + w_l(k)$$

$$Z_l = \begin{bmatrix} z(1) \\ z(2) \\ \cdots \\ z(L) \end{bmatrix}, g = \begin{bmatrix} g(0) \\ g(1) \\ \cdots \\ g(N) \end{bmatrix}, w_l = \begin{bmatrix} w(1) \\ w(2) \\ \cdots \\ w(L) \end{bmatrix}$$

$$H_l = \begin{bmatrix} u(1) & u(0) & \cdots u(1-N) \\ u(2) & u(1) & \cdots u(2-N) \\ \cdots & \cdots & \cdots \cdots \\ u(L) u(L-1) & \cdots & u(L-N) \end{bmatrix}$$

$$\hat{g}_{LS} = (H_L{}^T H_L) H_L{}^T z_L$$

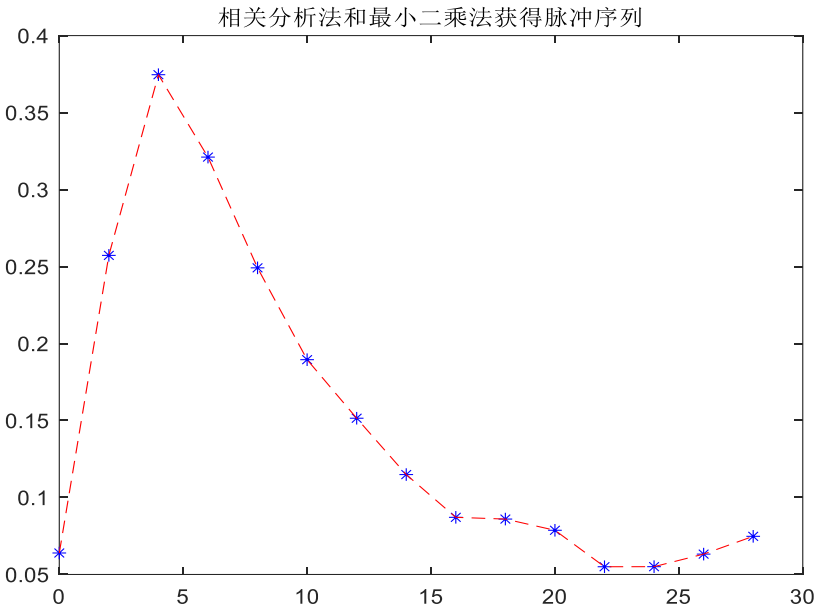
在 ppt 上，求脉冲响应函数 GIs 公式为上式，但是经过重新推导，我认为，实际的脉冲响应函数公式应该为

$$g_{ls} = inv(H_l^T H_l) * H_l^T z_l$$

相关程序编写如下

```
% 相关分析法获得脉冲响应序列
N = 2^4-1;deta = 2;  %deta 为脉冲的周期
a = 5;  %a 为脉冲正输出值
M = zeros(N,N);
for i=1:1:N
    for j=1:1:N
        M(i,j) = UY(N-i+j,1); %给矩阵 M 赋值
    end
end
Y = zeros(N,1);
for i=1:1:N
    Y(i,1)=UY(N+i-1,2); %给矩阵 Y 赋值
end
R = ones(N,N);
for i = 1:1:N
    R(i,i) = 2;      %给矩阵 Rym 赋值
end
G = (1/(a*a*(N+1)*deta))*R*M*Y;  %这是相关分析法辨识脉冲响应的最重要公式
t = 0:2:28;  %由于周期为 2，故按 2 进位
plot(t',G,'b*');   %绘制响应曲线
hold on
L=15;N=15;
Yy=UY(N:L+N-1,2)/2;
for i=1:L
    Hl(i,:)=UY(N+i-1:-1:i,1);
end
GG=inv(Hl'*Hl)*Hl'*Yy;  %ppt 上的公式存在错误，重新推导以后将其修改
 plot(0:2:28,GG,'r--')
title('相关分析法和最小二乘法获得脉冲序列');
```

利用修正以后的公式，得到了最小二乘法脉冲序列的结果。对比相关分析法和最小二乘法获得的脉冲序列，二者的结果完全一样，验证了编程进行模型参数辨识结果的正确性。



在上图中，星号表示相关分析法的辨识结果，折线段辨识最小二乘法的辨识结果。其中，星号为蓝色，折线段为红色，但是由于打印问题，相关颜色无法显示，如果有兴趣可以尝试跑一下程序去测试比较一下。

（3）应用相关最小二乘法，拟合对象的差分方程模型；

对于最小二乘法的脉冲响应序列，其求解差分方程的方式和相关分析法相同，故不再单独阐述。我采用最小二乘法，设定差分方程为 2 阶，之间进行编程求解差分方程参数。经过编程，我们所得的差分方程模型的相关参数为

```
>> question1_3
差分方程的参数a1 a2 b1 b2为

Theta =

   -0.9667
    0.2185
    0.4005
    0.2482
```

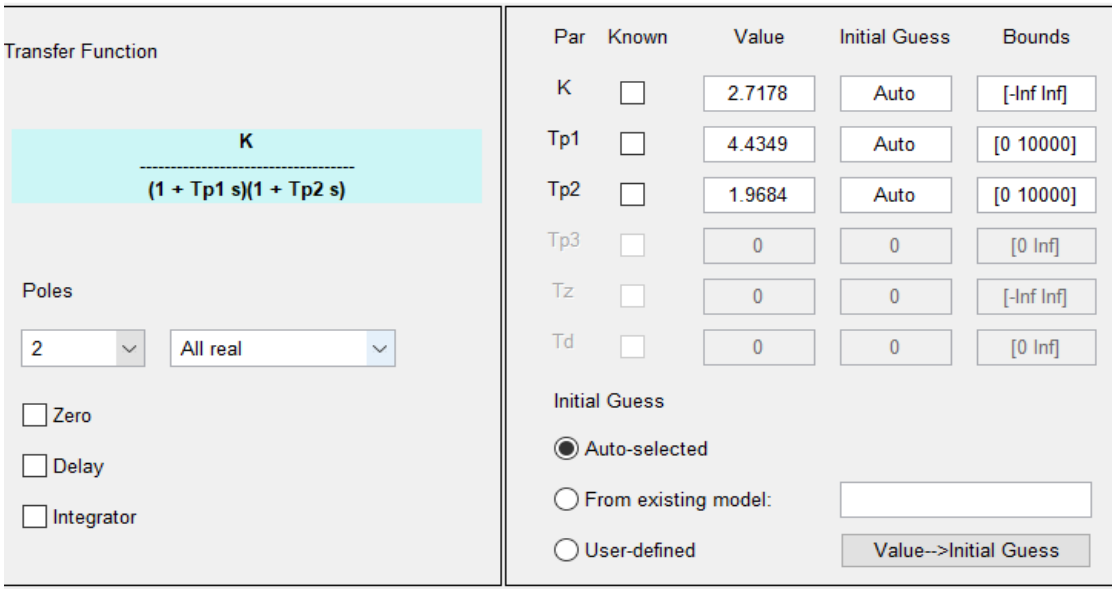
相关程序编写如下：

```
%相关最小二乘法辨识
u=UY(1:201,1)';        %输入矩阵
```

```
    z=UY(1:201,2)';       %输出矩阵
    H=zeros(150,4);
    for i=1:150
        H(i,1)=-z(i+1);
        H(i,2)=-z(i);
        H(i,3)=u(i+1);
        H(i,4)=u(i);
    end
    disp('差分方程的参数 a1 a2 b1 b2 为');
Theta=inv(H'*H)*H'*(z(3:152))'
```

对比相关分析法的辨识参数，二者辨识参数相近，验证了所得参数。但是二者在参数 a2 上的存在一定的差异，其原因在相关分析法在脉冲响应序列转化为传递函数时，推导时舍弃了高阶差分部分，所以导致辨识结果中 a2 的误差较大。

此外，我还使用了系统辨识工具箱辨识的系统参数，辨识结果为



此时系统的传递函数为

$$\frac{2.7178}{(1+4.4349*s)(1+1.9684*s)}=\frac{0.3113}{s\text{^}2+0.733\,s+0.1145}$$

相关分析法和最小二乘法得到的传递函数 G(s)为(最小二乘法和相关分析法脉冲响应序列相同，故传递函数也相同)

$$\frac{0.003662\,s\;+\;0.3375}{s\text{^}2\;+\;0.7528\,s\;+\;0.1082}$$

考虑到分子上 s 的参数极小，与其他参数相比可以忽略不计。所以辨识工具箱的辨识参数与相关分析法或最小二乘法所得的传递函数 G(s)极为相似，验证了辨识结果和参数的正确性。综上所述，我通过编程成功的辨识出二阶水箱的模型参数，并通过多种方式验证。

（4）构建时变对象，用最小二乘法和带遗忘因子的最小二乘法，(可以用辨识工具箱) 辨识模型的参数，比较两种方法的辨识效果差异；

在构建时变对象之前，首先应该明白时变对象的含义，在不同时刻输入相同的信号会得到不同的输出的系统被称为时变系统。在本题中，我设计线性时变系统进行分析，该线性时变系统为差分方程或者传递函数，其相关的参数会随着时间推移发生变化。

参数变化两种设计方法，一种是分段设计，在不同时间段为不同的常数；另一种是随着时间变化缓慢增加或减少。第一种采取分段计算的方式即可辨识出准确的参数，而用迭代的方法体现不出辨识参数的变化，故没有讨论的价值，所以我采用第二种方式设计参数变化，并采用迭代的方式对比参数变化。此外，由于如果在 Simulink 中构建时变对象，需要运用 S-Function 函数，（类似于老师在实验中构建的 tank 模型），较为复杂。所以我采用编程的方式构建时变对象，直接编程去进行分析

在进行编程之前，首先我简单阐述一下最小二乘法的相关知识和在本次编程中所用到的知识

一次完成的最小二乘法

$$Z_m=\begin{bmatrix} z(1) \\ z(2) \\ \vdots \\ z(m) \end{bmatrix} \quad H_m=\begin{bmatrix} h(1) \\ h(2) \\ \vdots \\ h(m) \end{bmatrix}=\begin{bmatrix} -y(0) & \cdots & -y(1-n) & u(0) & \cdots & u(1-n) \\ -y(1) & \cdots & -y(2-n) & u(1) & \cdots & u(2-n) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -y(m-1) & \cdots & -y(m-n) & u(m-1) & \cdots & u(m-n) \end{bmatrix}$$

$$\theta=\begin{bmatrix} a_1 & \cdots & a_n & b_1 & \cdots & b_n \end{bmatrix}^T \quad V_m=\begin{bmatrix} v(1) & v(2) & \cdots & v(m) \end{bmatrix}^T$$

$$Z_m = H_m\theta + V_m$$

$$\hat{\theta}=(H_m^T H_m)^{-1} H_m^T Z_m$$

相关编程方法如下：

```
%最小二乘法一次实现的结果
%设置时变参数
num=zeros(800,3);
for i=1:800
    num(i,2)=0.8+i/400;   %;构建 b1 值从 0.8 变化为 1.2
    num(i,3)=0.4+i/400;   %构建 b2 值从 0.4 变化为 0.8
end
num(400:800,2)=1.2;
num(400:800,3)=0.8;
b1=num(:,2);b2=num(:,3);
a1=-1.5;a2=0.7;
%设置 M 序列
L=800; % M 序列的长度
y1=1;y2=1;y3=1;y4=1;%4 个移位积存器的输出初始值
for i=1:L %开始循环，长度为 L
    x1=xor(y3,y4);
    x2=y1; x3=y2; x4=y3; y(i)=y4;
    if y(i)>0.5
        U(i)=1; %如果 M 序列的值为"1"时,%辨识的输入信号取“-1”
    else
        U(i)=0; %当 M 序列的值为"0"时.辨识的输入信号取“1”
    end
    y1=x1;y2=x2;y3=x3;y4=x4; %为下一次的输入信号做准备
end %大循环结束，产生输入信号 u
Y(2)=0;Y(1)=0; %取 Y 的前两个初始值为零
for k=3:800
Y(k)=-a1*Y(k-1)-a2*Y(k-2)+b1(k)*U(k-1)+b2(k)*U(k-2);
end
UY=[U' Y'];
u=UY(1:750,1)';        %输入矩阵
z=UY(1:750,2)';        %输出矩阵
H=zeros(700,4);
```

```
for i=1:700       %采用最小二乘法一次实现
    H(i,1)=-z(i+1);
    H(i,2)=-z(i);
    H(i,3)=u(i+1);
    H(i,4)=u(i);
end
Theta=inv(H'*H)*H'*(z(3:702))'
```

迭代完成最小二乘法

$$\hat{\theta}_{m+1} = \hat{\theta}_m + K_{m+1}[z(m+1) - h(m+1)\hat{\theta}_m]$$

$$P_{m+1} = P_m - P_m h^T(m+1)[w^{-1}(m+1) + h(m+1)P_m h^T(m+1)]^{-1} h(m+1)P_m$$

$$K_{m+1} = P_m h^T(m+1)[w^{-1}(m+1) + h(m+1)P_m h^T(m+1)]^{-1}$$

相关编程如下：

```
%最小二乘法迭代实现的结果
%设置时变参数
num=zeros(800,3);
for i=1:800
    num(i,2)=0.8+i/800;   %;构建 b1 值从 0.8 变化为 1.2
    num(i,3)=0.4+i/800;   %构建 b2 值从 0.4 变化为 0.8
end
num(400:800,2)=1.2;
num(400:800,3)=0.8;
b1=num(:,2);b2=num(:,3);
a1=-1.5;a2=0.7;
%设置 M 序列
L=800; % M 序列的长度
y1=1;y2=1;y3=1;y4=1;%4 个移位积存器的输出初始值
for i=1:700 %开始循环，长度为 L
    x1=xor(y3,y4);
    x2=y1; x3=y2; x4=y3; y(i)=y4;
    if y(i)>0.5
        U(i)=1; %如果 M 序列的值为"1"时,%辨识的输入信号取“-1”
    else
        U(i)=0; %当 M 序列的值为"0"时.辨识的输入信号取“1”
    end
    y1=x1;y2=x2;y3=x3;y4=x4; %为下一次的输入信号做准备
end %大循环结束，产生输入信号 u
Y(2)=0;Y(1)=0; %取 Y 的前两个初始值为零
for k=3:700
Y(k)=-a1*Y(k-1)-a2*Y(k-2)+b1(k)*U(k-1)+b2(k)*U(k-2);
end
```

```
UY=[U' Y'];
L=680;
%---------RLS 递推最小二乘辨识----------------------------------------
c0=[0.001 0.001 0.001 0.001]'; %直接给出被辨识参数的初始值,
%即一个充分小的实向量
p0=10^6*eye(4); %直接给出初始状态 P0，
%即一个充分大的实数单位矩阵
%E=0.000000005; %相对误差 E=0.000000005
c=[c0,zeros(4,L)]; %被辨识参数矩阵的初始值
e=zeros(4,L); %相对误差的初始值及大小
z(1:L)=UY(1:L,2);
u(1:L)=UY(1:L,1);
for k=3:L %开始求 K
    h1=[-z(k-1),-z(k-2),u(k-1),u(k-2)]';
     x=h1'*p0*h1+1;
     x1=inv(x);
k1=p0*h1*x1; %求出 K 的值
d1=z(k)-h1'*c0; c1=c0+k1*d1; %求被辨识参数 c
e1=c1-c0; %求参数当前值与上一次的值的差值
e2=e1./c0; %求参数的相对变化
e(:,k)=e2; %把当前相对变化的列向量加入误差矩阵的最后一列
c0=c1; %新获得的参数作为下一次递推的旧参数
c(:,k)=c1; %把辨识参数 c  列向量加入辨识参数矩阵的最后一列
p1=p0-k1*k1'*[h1'*p0*h1+1]; %求出 p(k)的 s 值
p0=p1; %给下次用
%if e2<=E break; %若参数收敛满足要求，终止计算
end
%end
c(:,L),e(:,L) %显示被辨识参数,显示辨识结果的收敛情况
% %------分离参数----------------------------------------------------------
  a1=c(1,1:L); a2=c(2,1:L); b1=c(3,1:L); b2=c(4,1:L);
ea1=e(1,1:L); ea2=e(2,1:L); eb1=e(3,1:L); eb2=e(4,1:L);
% %------画图--------------------------------------------------------------
%画出 a1，a2，b1，b2 的各次辨识结果
figure(2);i=1:L; plot(i,a1,'r',i,a2,'y',i,b1,'g.',i,b2,'b-')
legend('a1','a2','b1','b2');
title('a1，a2，b1，b2 的各次辨识结果')
%画出 a1，a2，b1，b2 的各次辨识结果的收敛情况
figure(3); i=1:L; plot(i,ea1,'r',i,ea2,'g',i,eb1,'b',i,eb2,'r:')
title('辨识结果的收敛情况')
%-------------------------------------------------------------------------
```

带遗忘因子的最小二乘法

$$\hat{\theta}_{m+1} = \hat{\theta}_m + K_{m+1}[z(m+1) - h(m+1)\hat{\theta}_m]$$

$$K_{m+1} = P_m h^T(m+1)[\lambda I + h(m+1)P_m h^T(m+1)]^{-1}$$

$$P_{m+1} = \frac{1}{\lambda}[I - K_{m+1}h(m+1)]^{-1}P_m$$

$$\lambda = 0.95 \sim 0.98$$

带遗忘因子的最小二乘法的编程如下

```
%带遗忘因子的最小二乘法
%设置时变参数
num=zeros(800,3);
for i=1:800
    num(i,2)=0.8+i/1000;   %;构建 b1 值从 0.8 变化为 1.2
    num(i,3)=0.4+i/1000;   %构建 b2 值从 0.4 变化为 0.8
end
num(400:800,2)=1.2;
num(400:800,3)=0.8;
b1=num(:,2);b2=num(:,3);
a1=-1.5;a2=0.7;
%设置 M 序列
L=800; % M 序列的长度
y1=1;y2=1;y3=1;y4=1;%4 个移位积存器的输出初始值
for i=1:L %开始循环，长度为 L
    x1=xor(y3,y4);
    x2=y1; x3=y2; x4=y3; y(i)=y4;
    if y(i)>0.5
        U(i)=1; %如果 M 序列的值为"1"时,%辨识的输入信号取“-1”
    else
        U(i)=0; %当 M 序列的值为"0"时.辨识的输入信号取“1”
    end
    y1=x1;y2=x2;y3=x3;y4=x4; %为下一次的输入信号做准备
end %大循环结束，产生输入信号 u
Y(2)=0;Y(1)=0; %取 Y 的前两个初始值为零
for k=3:800
Y(k)=-a1*Y(k-1)-a2*Y(k-2)+b1(k)*U(k-1)+b2(k)*U(k-2);
end
UY=[U' Y'];
%带遗忘因子的最小二乘法
na=2; nb=2;d=0;
L=700;u=0.95;       %u 为遗忘因子
uk=UY(1:L,1)';
```

```
yk=UY(1:L,2)';
thetae_1=zeros(na+nb,1);
P=10^6*eye(na+nb);
for k=3:L     %采用迭代的方式进行推导计算
phi=[-yk((k-1):-1:(k-2)) uk((k-1):-1:(k-2))];              K=P*phi'/(u+phi*P*phi');
      thetae(:,k)=thetae_1+K*(yk(k)-phi*thetae_1);
      P=(eye(na+nb)-K*phi)*P/u;
      thetae_1=thetae(:,k);   %thita 值迭代计算
      for i=d+nb:-1:2
            uk(i)=uk(i-1);
      end
      for i=na:-1:2
            yk(i)=yk(i-1);
      end
end
plot([1:L],thetae);
theta = thetae(:,700)
legend('a1','a2','b1','b2');
title('带遗忘因子的最小二乘法辨识参数')
```

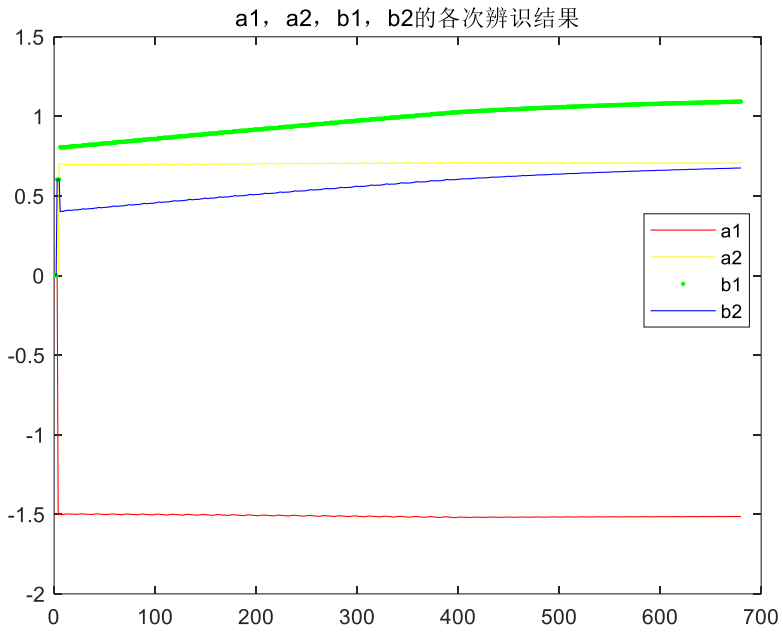
之后我进行编程实现相关算法。其中，一般最小二乘法分别采用一次实现和迭代实现的方法，带遗忘因子的最小二乘法采用迭代实现的方法，便于后续的结果对比。此外，我还采用辨识工具箱进行辅助辨识。在这个实验中，设定参数 b1 从 0.8 变化到 1.2 并保持稳定，参数 b2 从 0.4 变化到 0.8 并保持稳定。参数 a1=-1.5，参数 a2=0.7。进行相关辨识。

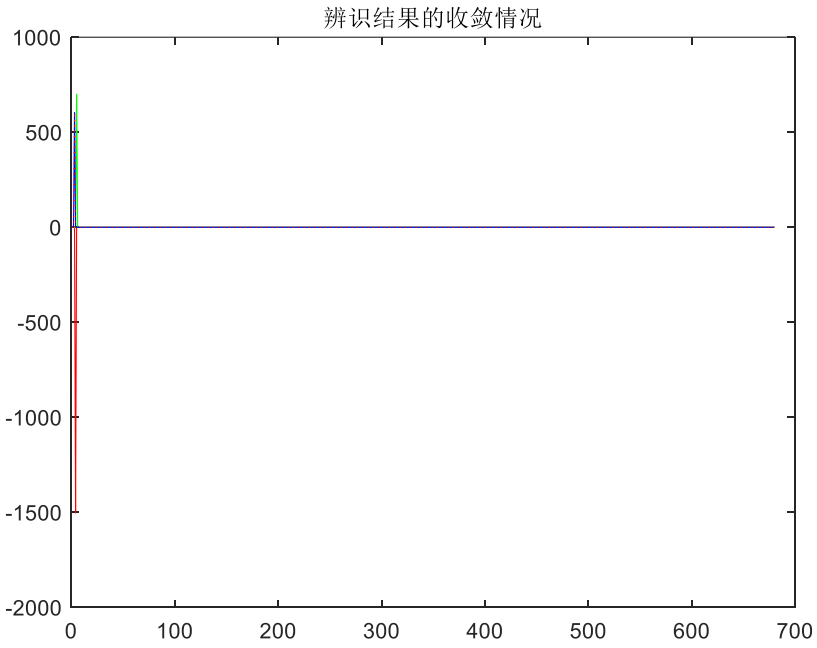
最小二乘法一次实现的结果：

```
Theta =

   -1.5284
    0.7127
    1.2137
    0.7747
```

最小二乘法迭代实现的结果：

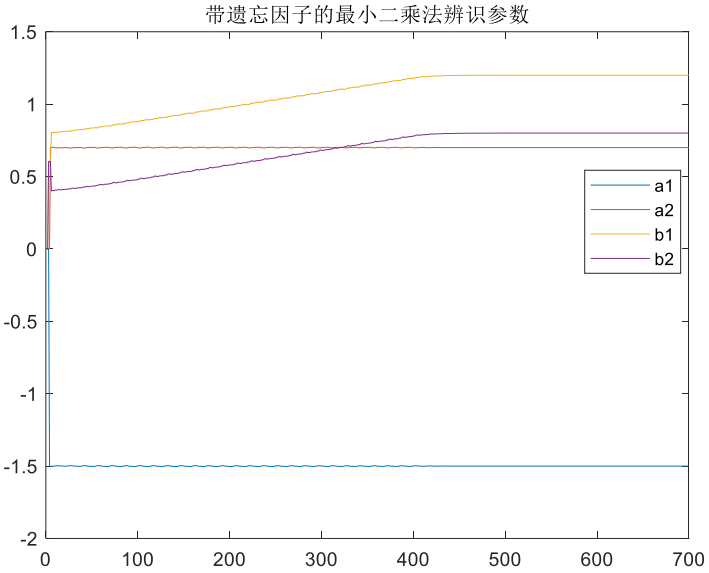
```
>> question1_4_3

ans =

   -1.5143
    0.7061
    1.0928
    0.6754
```





带遗忘因子的最小二乘法采用迭代实现的结果：

```
>> question1_4_2

theta =

   -1.5000
    0.7000
    1.2000
    0.8000
```



系统辨识工具箱辨识结果（只能进行一次辨识）为：

```
  From input "u1" to output "y1":
  1.252 z^-1 + 0.8291 z^-2
  -------------------------
  1 - 1.502 z^-1 + 0.7 z^-2
Name: tf1
Sample time: 1 seconds
Discrete-time identified transfer function.
```

对比三种方式的辨识结果和系统辨识工具箱的辅助辨识结果，我发现，带遗忘因子的最小二乘法采用迭代实现这种方法辨识的结果最为准确，并可以有效反映时变对象中参数 b1，b2 的变化。此外，对比最小二乘法一次完成算法和迭代完成算法，我发现，最小二乘法一次完成算法对于时变参数最后稳定值的辨识结果较为准确，但是不能反映参数随时间的变化情况。迭代算法可以有效反应时变参数的变化情况，但是辨识的准确性受到了影响。

（5）给出水箱模型的阶的辨识

对于这个问题，我采用了两种定阶方法确定模型的阶次

1.AIC 定阶方法

$$a(z^{-1})y(k)=b(z^{-1})u(k)+e(k)$$

$$a(z^{-1})=1+a_1z^{-1}+\cdots+a_{n1}z^{-n1}$$

$$b(z^{-1})=b_0+b_1z^{-1}+\cdots+b_{n2}z^{-n2}$$

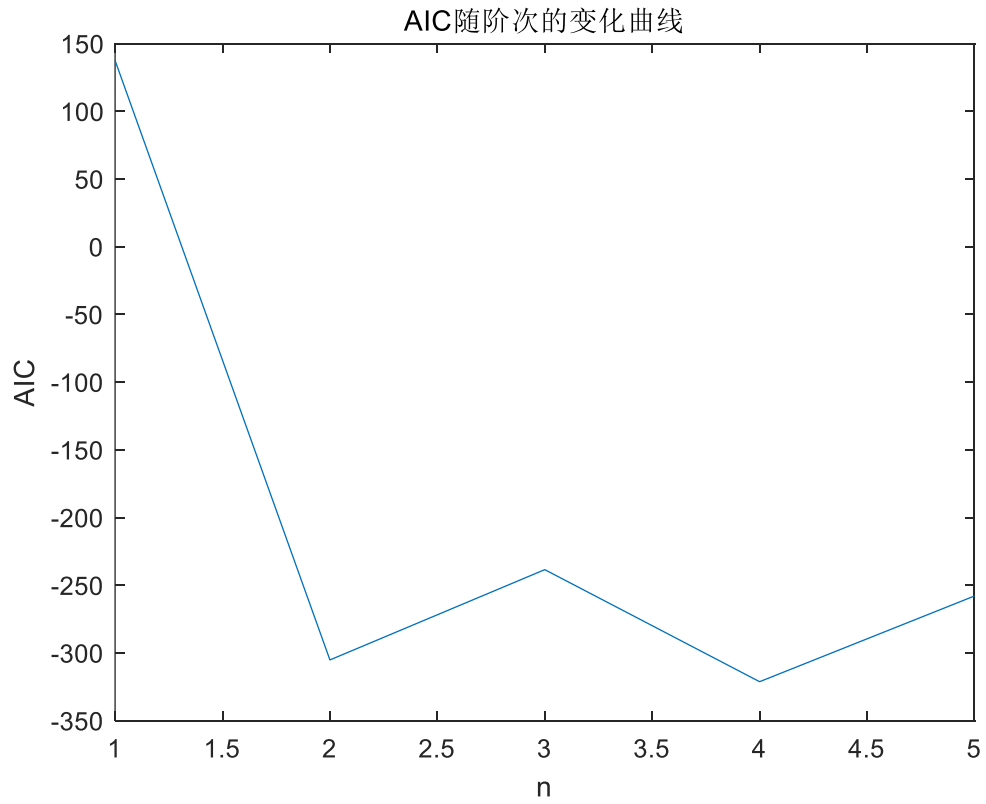
$$L(\mathbf{Y}\mid\boldsymbol{\theta})=(2\pi\sigma\hat{\sigma}_e^2)^{-N/2}\exp\left\{-\frac{1}{2\sigma_e^2}(\mathbf{Y}-\boldsymbol{\Phi\theta})^T(\mathbf{Y}-\boldsymbol{\Phi\theta})\right\}$$

$$AIC=-2\ln L+2p=N\ln\hat{\sigma}_e^2+2(n1+n2)+C$$

编程方法如下

```
L = 100;%第一种方法——AIC 定阶方法
U = UY(:,1);
Y = UY(:,2);
for n = 1:1:5
        N = 100-n;
        yd(1:N,1) = Y(n+1:n+N);
        for i=1:N
            for l=1:1:2*n
                if(l<=n)
                  FIA(i,l) = -Y(n+i-l);  %矩阵构造
                  else
                  FIA(i,l) = U(2*n+i-l);
                end
            end
        end
        thita = inv(FIA'*FIA)*FIA'*yd;
        omiga = (yd-FIA*thita)'*(yd-FIA*thita)/N; %求解 omiga
        AIC(n) = N*log(omiga)+4*n;
end
plot(AIC,'-');
title('AIC 随阶次的变化曲线');
xlabel('n');
ylabel('AIC');
```

经过编程，所得结果如下



观察所得结果易知，系统阶可能为 2 或 4，但是考虑系统为 2 阶水箱模型，所以系统阶次为 2。

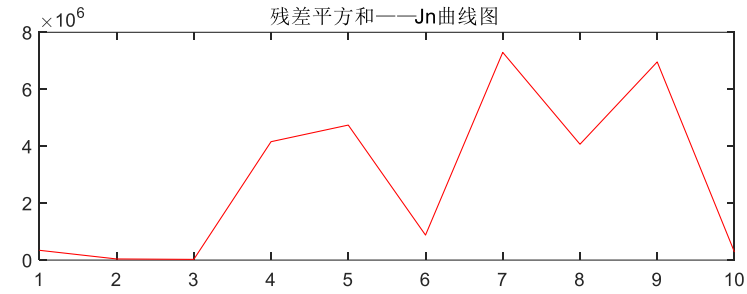
2 残差定阶方法

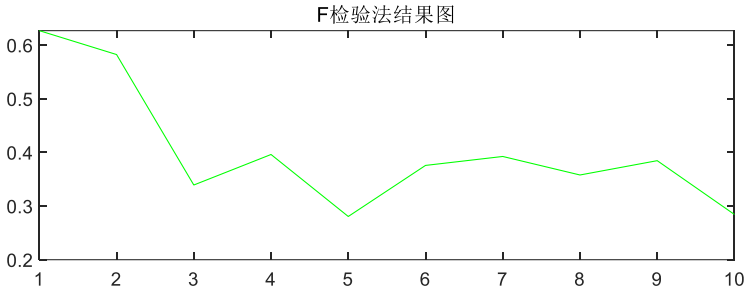
编程方法如下：

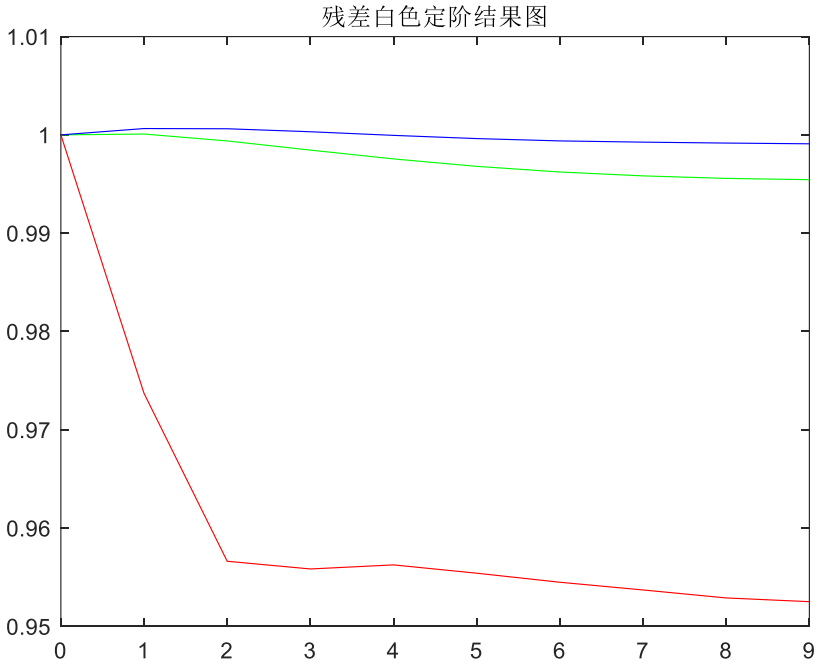
```
%第二种方法，残差定阶法
Data = UY; L = length(Data);%输入输出数据长度
Jn = zeros(1,10);
t = zeros(1,10);
rm = zeros(10,10);
for n=1:1:10;
    N = L-n;
    FIA = zeros(N,2*n);%构造测量矩阵
    du = zeros(n,1);
    dy = zeros(n+1,1);
    r1 = 0;r0 = 0;
for i = 1:N          %测量矩阵赋值
    for l = 1:n*2
        if(l<=n)
            FIA(i,l) = -Data(n+i-l,2);
        elseif(n+i-l+2>0)
```

```
                FIA(i,l) = Data(n+i-l+2,1);
            end
        end
end
Y = Data(n+1:n+N,2);%输出数据矩阵
thita = inv(FIA'*FIA)*FIA'*Y;%计算参数矩阵
        Jn0 = 0;
        for k = n+1:n+N
            for j = 1:n
                du(j) = Data(k-j,1);
                dy(j) = Data(k+1-j,2);
            end
            dy(n+1) = Data(1,2);
            E1(k) = [1,thita(1:n)']*dy-thita(n+1:2*n)'*du;
            Jn1 = Jn0+E1(k)^2;
            %F 检验法
            t(n) = abs((Jn0-Jn1)/Jn1*(N-2*n-2)/2);
            Jn0 = Jn1;
        end
        Jn(n) = Jn0;
        for m = 0:1:9
            for m2 = n+1:1:L-m
                r1 = r1+E1(m2)*E1(m2+m)/(L-m-n);
                r0 = r0+E1(m2)^2/(L-m-n);
            end
            rm(n,m+1) = r1/r0;
        end
end
subplot(2,1,1);
plot(1:10,Jn,'r');
title('残差平方和——Jn 曲线图');
subplot(2,1,2);
plot(1:1:10,t,'g');
title('F 检验法结果图');
figure(2);
plot(0:9,rm(1,:),'g'),hold on;
plot(0:9,rm(2,:),'b'),hold on;
plot(0:9,rm(3,:),'r');
title('残差白色定阶结果图');
hold off;
```

经过编程，所得结果如下：







由残差平方和——Jn 曲线（残差平方和越小，阶次越合理）可定系统阶次为 2 或 3。但是考虑到第一种的模型定阶方法的定阶结果。综上，系统阶次定为 2，也符合实际二阶水箱系统的阶次。

第二题 模仿 index， 构建适合使用增广最小二乘法的对象和适合使用广义最小二乘法的对象模型，用学过的所有参数辨识算法对这两个对象进行辨识，分析各种方法的辨识效果，计算量大小。

和第一题类似，首先我简单阐述一下增广最小二乘法和广义最小二乘法所使用的理论基础，为接下来的编程做好准备。

增广最小二乘法编程时所用的公式

$$z(k)=-\sum_{i=1}^{n}a_i y(k-i)+\sum_{i=1}^{n}b_i u(k-i)+v(k)+\sum_{i=1}^{n}c_i v(k-i)$$

$$\theta=[a_1,a_2,\cdots,a_n,b_1,b_2,\cdots,b_n,c_1,\cdots,c_n]^T$$

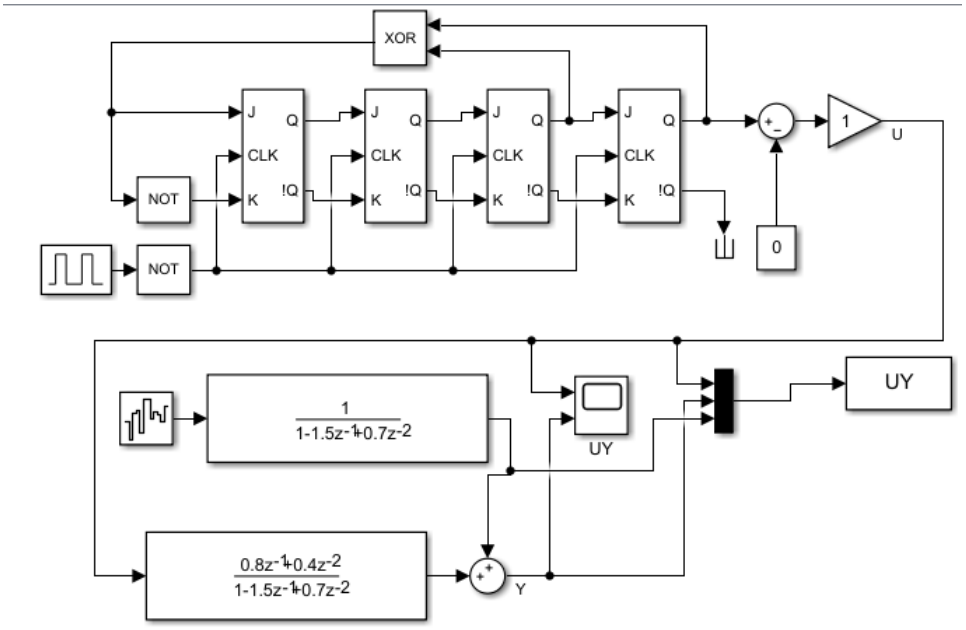
$$z(k)=h(k)\theta+v(k)$$

$$\hat{\theta}_{m+1}=\hat{\theta}_m+K_{m+1}[z(m+1)-h(m+1)\hat{\theta}_m]$$

$$P_{m+1}=P_m-P_m h^T(m+1)[w^{-1}(m+1)+h(m+1)P_m h^T(m+1)]^{-1}h(m+1)P_m$$

$$K_{m+1}=P_m h^T(m+1)[w^{-1}(m+1)+h(m+1)P_m h^T(m+1)]^{-1}$$

适用于该算法的 Simulink 对象为



广义最小二乘法编程时所用的公式

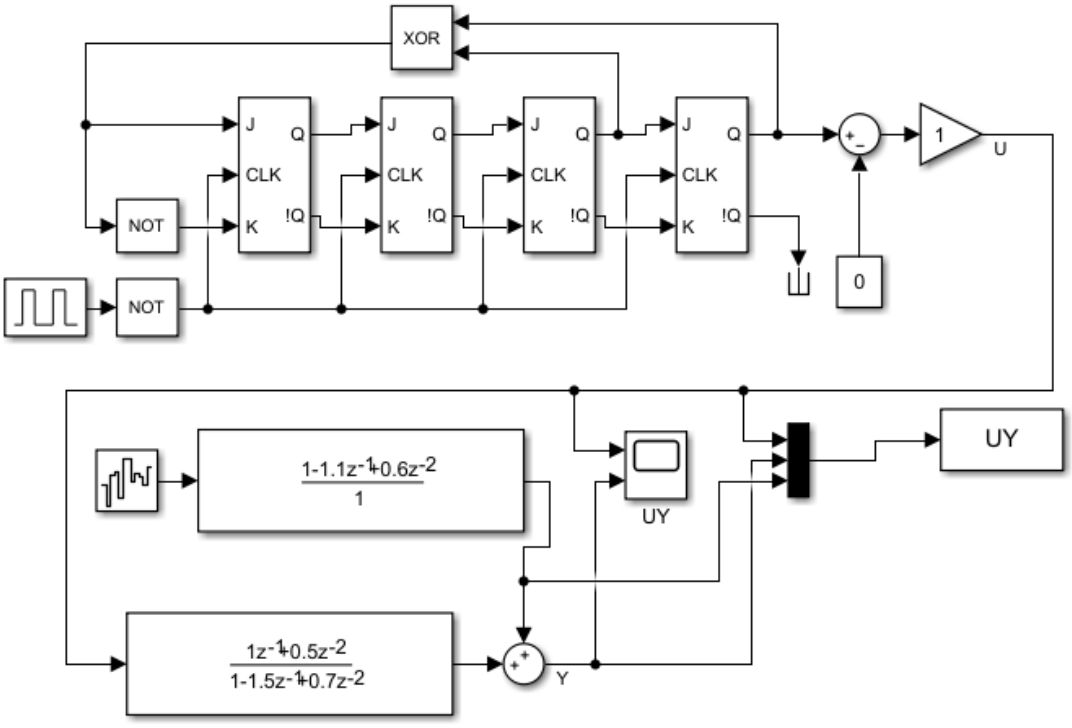
$$\begin{cases} a(z^{-1})y(k) = b(z^{-1})u(k) + \xi(k) \\ a(z^{-1}) = 1 + a_1 z^{-1} + \Lambda + a_n z^{-n} \\ b(z^{-1}) = b_0 + b_1 z^{-1} + \Lambda + b_n z^{-n} \end{cases}$$

$$\begin{cases} \xi(k) = \dfrac{1}{f(z^{-1})}\varepsilon(k) \\ f(z^{-1}) = 1 + f_1 z^{-1} + \Lambda + f_m z^{-m} \end{cases}$$

对于广义最小二乘法，传递函数参数辨识可以参照最小二乘法，噪声传递函数则用下面的公式

$$\hat{\mathbf{f}} = (\boldsymbol{\Omega}^{\mathrm{T}}\boldsymbol{\Omega})^{-1}\boldsymbol{\Omega}^{\mathrm{T}}\boldsymbol{\xi}$$

适用于该算法的 Simulink 对象为



需要注意的是，根据 ppt 上所叙述的，增广最小二乘法和广义最小二乘法辨识参数位置并不相同。此外，使用广义最小二乘法辨识时，需要在干扰信号上加一个白化滤波器，提高辨识的准确性。假设噪声传递函数为 G1(z)，则

$$\mathrm{G1(z)} = \frac{d1 + c11 * z^{-1} + c12 * z^{-2}}{d2 + c21 * z^{-1} + c22 * z^{-2}}$$

其中，广义最小二乘法辨识参数分子参数 c21，c22，此时 d2、d1=1， c11、c12 为 0；增广最小二乘法辨识分母参数 c11，c12，此时 d1、d2=1，c21，c22 为 0，所以我们需要构造不同的对象进行算法实现。

综上所述，经过实际分析和测试，在本题中使用增广最小二乘法测试对象的传递函数为

$$\mathrm{G(z)} = \frac{1 * z^{-1} + 0.5 * z^{-2}}{1 - 1.5 * z^{-1} + 0.7 * z^{-2}}$$

其噪声传递函数为

$$\mathrm{G1(z)} = \frac{1 - 1.1 * z^{-1} + 0.6 * z^{-2}}{1}$$

在本题中使用广义最小二乘法测试对象的传递函数为

$$\mathrm{G(z)} = \frac{0.8 * z^{-1} + 0.4 * z^{-2}}{1 - 1.5 * z^{-1} + 0.7 * z^{-2}}$$

其噪声传递函数为

$$\mathrm{G1(z)} = \frac{1}{1 - 1.5 * z^{-1} + 0.7 * z^{-2}}$$

增广最小二乘法的编程方法为：

```
%lipengboLS
%增广最小二乘
 N=600;
% U=(idinput(N,'prbs',[0 1])+1)*0.5;
% E=normrnd(0,0.1,N,1);ek=zeros(N,1);
% % y=zeros(N,1);
% for i=3:N
%         ek(i)=[E(i) E(i-1) E(i-2)]*[1 -1.5 0.7]';
% %         y(i)=[-y(i-1) -y(i-2) U(i-1) U(i-2)]*[-1.5 0.7 1 0.5]'+ek(i);
% end
```

```
U=UY(1:N,1);        %输入矩阵
y=UY(1:N,2);        %输出矩阵
e=UY(1:N,3);        %误差矩阵
%e=ek;
z=[y,U];
Thita=rels(z,2,2,2,e);
x=1:N;
plot(x,Thita(1,x),'r');
hold on
plot(x,Thita(2,x),'r');
plot(x,Thita(3,x),'b');
plot(x,Thita(4,x),'g');
plot(x,Thita(5,x),'y');
plot(x,Thita(6,x),'y');
Thita(:,300)

function thita=rels(z,na,nb,nc,e) %程序中所用的函数
nz=length(z(:,1));
nn=na+nb+nc;
thitak=ones(nn,1)*0.001;
thita=zeros(nn,nz);
p1=eye(nn,nn)*(1.0e4);
p2=zeros(nn,nn);
K=zeros(nn,1);
   e=zeros(nz,1);
I=eye(nn);
for i=na+1:nz
      Q=[[-z(i-1:-1:i-na,1)]',[z(i-1:-1:i-nb,2)]',[e(i-1:-1:i-nc,1)]'];
      K=p1*Q'*inv(Q*p1*Q'+1);
      p2=(I-K*Q)*p1;
      thita(:,i)=thitak+K*(z(i,1)-Q*thitak);
      p1=p2;
      thitak=thita(:,i);
      e(i)=z(i,1)-Q*thitak;
end
```

增广最小二乘法的辨识结果为

```
>> question2_2

ans =

   -1.3451
    0.5650
    1.0226
    0.6537
   -1.1408
    0.6446
```

广义最小二乘法的编程方法为：

```
%广义最小二乘法
u=UY(1:800,1)';      %输入矩阵
z=UY(1:800,2)';      %输出矩阵
e=UY(1:800,3)';
P=100*eye(4);   %估计方差
Pstore=zeros(4,400);
Pstore(:,2)=[P(1,1),P(2,2),P(3,3),P(4,4)];
Theta=zeros(4,400);  %参数的估计值，存放中间过程估值
Theta(:,2)=[2;2;2;2];
K=[1;1;1;1];  %增益
PE=100*eye(2);
ThetaE=zeros(2,400);
ThetaE(:,2)=[0.5;0.3];
KE=[10;10];
for i=3:400   %迭代辨识得到结果
    h=[-z(i-1);-z(i-2);u(i-1);u(i-2)];
    K=P*h*inv(1+h'*P*h);
    Theta(:,i)=Theta(:,i-1)+K*(z(i)-h'*Theta(:,i-1));
    P=(eye(4)-K*h')*P;
    Pstore(:,i-1)=[P(1,1),P(2,2),P(3,3),P(4,4)];
    he=[-e(i-1);-e(i-2)];
    KE=PE*he*inv(1+he'*PE*he);
    ThetaE(:,i)=ThetaE(:,i-1)+KE*(e(i)-he'*ThetaE(:,i-1));
    PE=(eye(2)-KE*he')*PE;
end
disp('参数 a1 a2 b1 b2 的估计结果：')
Theta(:,400)
disp('噪声传递系数 c1 c2 的估计结果：')
ThetaE(:,400)
% i=1:400;
% figure(1)
% plot(i,Theta(1,:),i,Theta(2,:),i,Theta(3,:),i,Theta(4,:))
% title('参数过渡过程')
% legend('a1','a2','b1','b2')
% figure(2)
% plot(i,Pstore(1,:),i,Pstore(2,:),i,Pstore(3,:),i,Pstore(4,:))
% title('估计方差过渡过程')
% figure(3)
% plot(i,ThetaE(1,:),i,ThetaE(2,:));
% title('噪声传递系数过渡过程')
% legend('e1','e2')
```

广义最小二乘法辨识结果为

```
>> question2_1
参数a1 a2 b1 b2的估计结果：

ans =

   -1.5010
    0.7134
    0.8187
    0.4237

噪声传递系数c1 c2的估计结果：

ans =

   -1.4905
    0.7157
```

观察上述的辨识结果，我发现这两种方法对于传递函数参数和噪声传递函数参数的辨识结果都非常理想，可以比较准确的辨识出相关参数。但是这两种对于模型的适配性提出了较高的要求，需要选择合适的模型参数，才可以得到合适的辨识结果。

此外，我还尝试采取了一次完成的最小二乘算法、夏式修正法、辅助变量法对于这两个对象也进行了辨识。简单介绍一下这三种方法

一次完成的最小二乘算法：利用最小二乘法可以简便地求得未知的数据，并使得这些求得的数据与实际数据之间误差的平方和为最小。该方法计算量最少，但只适用于线性时不变系统

其编程方法为：

```
%最小二一次完成算法
u=UY(1:201,1)';          %输入矩阵
z=UY(1:201,2)';          %输出矩阵
H=zeros(150,4);
for i=1:150
```

```
    H(i,1)=-z(i+1);
    H(i,2)=-z(i);
    H(i,3)=u(i+1);
    H(i,4)=u(i);
end
Theta=inv(H'*H)*H'*(z(3:152))'
```

夏式修正法：克服基本 LS 有偏估计问题，提高广义 LS 的计算效率。该方法是一种无偏的估计方法，计算量较广义 LS 的要小许多，并且计算效率高，算法的适应性强，便于推广和扩展。

其编程方法为：

```
Data = UY;%生成数据矩阵
%使用夏氏修正法，对 2 阶系统进行参数辨识
n = 2;L = length(Data);N = L-n;
U = Data(:,1);
Y = Data(:,2);
glOL =[-Y(2:L-1),-Y(1:L-2),U(2:L-1),U(1:L-2)];
Zgl1 = Data(3:L,2);
Sgl1 = glOL'*glOL;Sgl2=inv(Sgl1);Sgl3=glOL'*Zgl1;
Xsthita = Sgl2*Sgl3;%计算参数矩阵
thitab0 = 0;%设偏差项的偏差初值为 0
Fa = Sgl2*glOL';
M = eye(N)-glOL*Sgl2*glOL';
F = eye(N);
if(F>=10^-6*eye(N))
    E1 = Zgl1-glOL*Xsthita;%计算残差 E
    omiga(2:N,1) = -E1(1:N-1);
    omiga(3:N,2) = -E1(1:N-2);
    D = omiga'*M*omiga;
    Fx = inv(D)*omiga'*M*Zgl1;
    thitab = Fa*omiga*Fx;
    Xsthita = Xsthita - thitab;
    F = thitab - thitab0;
    thitab0 = thitab;
end
theta = [Xsthita(1) Xsthita(2) Xsthita(3) Xsthita(4)]'
```

辅助变量法：克服基本最小二乘估计的有偏估计问题，得到一种无偏参数估计算法。该方法的计算量与基本最小二乘估计相同，并且辨识精度高于最小 LS 估计法，但是参数估计时需构造辅助变量矩阵。

其编程方法为：

```
u=UY(1:700,1)';        %输入矩阵
z=UY(1:700,2)';        %输出矩阵
%递推求解
P=100*eye(4);     %估计方差
Pstore=zeros(4,400);
Pstore(:,1)=[P(1,1),P(2,2),P(3,3),P(4,4)];
Theta=zeros(4,400);   %参数的估计值，存放中间过程估值
Theta(:,1)=[3;3;3;3];
Theta(:,2)=[3;3;3;3];
Theta(:,3)=[0;0;0;0];
Theta(:,4)=[0;0;0;0];
%    K=zeros(4,400);   %增益矩阵
K = [10;10;10;10];
for i=5:400
      h=[-z(i-1);-z(i-2);u(i-1);u(i-2)];
      hstar=[-z(i-2-1);-z(i-2-2);u(i-1);u(i-2)];     %辅助变量
      K=P*hstar*inv(h'*P*hstar+1);
      Theta(:,i)=Theta(:,i-1)+K*(z(i)-h'*Theta(:,i-1));
      P=(eye(4)-K*h')*P;
      Pstore(:,i-1)=[P(1,1),P(2,2),P(3,3),P(4,4)];
end
theta = Theta(:,400)
i=1:400;
figure(1)
plot(i,Theta(1,:),i,Theta(2,:),i,Theta(3,:),i,Theta(4,:))
title('辨识参数过渡过程')
figure(2)
plot(i,Pstore(1,:),i,Pstore(2,:),i,Pstore(3,:),i,Pstore(4,:))
title('估计方差过渡过程')
```

由于最小二乘算法和辅助变量法只能辨识传递函数参数，不能辨识噪声参数，所以只记录比较传递函数辨识的参数。

适用于增广最小二乘法对象传递函数的辨识结果

辨识方法			
最小二乘法	夏式修正法	辅助变量法	增广最小二乘法
theta = -0.6616 -0.0443 1.1066 1.2539	theta = -1.3986 0.6105 1.0257 0.5814	theta = -1.2482 0.5186 1.1065 0.8350	theta= -1.3451 0.5650 1.0226 0.6537

适用于广义最小二乘法对象传递函数的辨识结果

辨识方法			
最小二乘法	夏式修正法	辅助变量法	广义最小二乘法
theta = -1.4832 0.7035 0.8560 0.5031	theta = -1.5105 0.7163 0.7913 0.4189	theta = -1.5220 0.7288 0.8125 0.4004	theta= -1.5010 0.7134 0.8187 0.4237

观察分析上述数据，大部分方式的辨识结果较为准确，但是适用于增广最小二乘法的对象，采用最小二乘法或辅助变量法进行辨识时，其参数辨识出现了较大的误差，其原因在于这两种方法并未考虑噪声对于输出的影响，而夏式修正法则考虑到噪声对于输出的影响。

此外，广义最小二乘法适用对象中噪声的传递函数本质上相当于一个滤波器，这降低了噪声对于输出的影响，所以最小二乘法和辅助变量法在辨识适用于广义最小二乘法对象的参数时更为准确。

综上，增广最小二乘法、广义最小二乘法、夏式法辨识效果较好，并且夏式法计算量较少，和一般最小二乘计算量相当。

第三题 采用多变量系统的最小二乘辨识方法辨识如下 MIMO 系统的参数（20 分）

$$\begin{bmatrix} y_1(k) \\ y_2(k) \end{bmatrix} + A_1 \begin{bmatrix} y_1(k-1) \\ y_2(k-1) \end{bmatrix} + A_2 \begin{bmatrix} y_1(k-2) \\ y_2(k-2) \end{bmatrix} = \begin{bmatrix} u_1(k) \\ u_2(k) \end{bmatrix} + B_1 \begin{bmatrix} u_1(k-1) \\ u_2(k-1) \end{bmatrix} + B_2 \begin{bmatrix} u_1(k-2) \\ u_2(k-2) \end{bmatrix} + \begin{bmatrix} v_1(k) \\ v_2(k) \end{bmatrix}$$

式中，$\{v_1(k)\}$和$\{v_2(k)\}$是同分布的随机噪声，且有$N(0,0.5)$；输入信号采用 4 阶M序列，其幅值为 1。模型的理想系数为

$$A_1 = \begin{bmatrix} 0.5 & -0.2 \\ -0.3 & 0.6 \end{bmatrix}; \quad A_2 = \begin{bmatrix} 1.2 & -0.6 \\ 0.1 & -0.6 \end{bmatrix}; \quad B_0 = \begin{bmatrix} 1.0 & 0.0 \\ 0.0 & 1.0 \end{bmatrix}; \quad B_1 = \begin{bmatrix} 0.5 & -0.4 \\ 0.2 & -0.3 \end{bmatrix} \quad B_2 = \begin{bmatrix} 0.4 & -0.3 \\ -0.2 & 0.1 \end{bmatrix}$$

在这个题中，不利用 simulink 中的模型对系统进行仿真和测试，利用程序根据系统的差分方程组直接获得系统的测试数据，并对系统参数进行辨识。在下面，列出编程时所需要用到的公式。此外，由于多次迭代计算较为复杂，对编程精确性、电脑计算性能要求较高，故最后采取一次编程计算求解的方式。

$$Y_j(k-i) = \begin{bmatrix} y_1(k-i) \\ y_2(k-i) \\ \vdots \\ y_m(k-i) \end{bmatrix} \quad i = 0,1,\cdots,n \qquad U(k-i) = \begin{bmatrix} u_1(k-i) \\ u_2(k-i) \\ \vdots \\ u_r(k-i) \end{bmatrix} \quad i = 1,2,\cdots,n \qquad V_j = \begin{bmatrix} V_j(1) \\ V_j(2) \\ \vdots \\ V_j(N) \end{bmatrix}$$

$$\theta_j = \begin{bmatrix} a_{1j1} & \cdots & a_{1jm} & \cdots & a_{nj1} & \cdots & a_{njm} & b_{0j1} & \cdots & b_{0jr} & \cdots & b_{nj1} & \cdots & b_{njr} \end{bmatrix}^T$$

$$H_j = \begin{bmatrix} -Y_j^T(0) & \cdots & -Y_j^T(1-n) & U^T(1) & \cdots & U^T(1-n) \\ -Y_j^T(1) & \cdots & -Y_j^T(2-n) & U^T(2) & \cdots & U^T(2-n) \\ \vdots & & \vdots & \vdots & & \vdots \\ -Y_j^T(N-1) & \cdots & -Y_j^T(N-n) & U^T(N) & \cdots & U^T(N-n) \end{bmatrix}$$

$$Y_j = H_j \theta_j + V_j$$

$$\hat{\theta}_j = (H_j^T H_j)^{-1} H_j^T Y_j$$

其编程方式为：

```
clc,clear all
L=200; % M 序列的长度
y1=1;y2=1;y3=1;y4=1;%4 个移位积存器的输出初始值
for i=1:L %开始循环，长度为 L
    x1=xor(y3,y4);
    x2=y1; x3=y2; x4=y3; y(i)=y4;
    if y(i)>0.5
        u1(i)=1; %如果 M 序列的值为"1"时,%辨识的输入信号取“-1”
    else
        u1(i)=0; %当 M 序列的值为"0"时.辨识的输入信号取“1”
    end
    y1=x1;y2=x2;y3=x3;y4=x4; %为下一次的输入信号做准备
end %大循环结束，产生输入信号 u
y1=1;y2=0;y3=1;y4=1;
for i=1:L %开始循环，长度为 L
    x1=xor(y3,y4);
    x2=y1; x3=y2; x4=y3; y(i)=y4;
    if y(i)>0.5
        u2(i)=1; %如果 M 序列的值为"1"时,%辨识的输入信号取“-1”
    else
        u2(i)=0; %当 M 序列的值为"0"时.辨识的输入信号取“1”
    end
    y1=x1;y2=x2;y3=x3;y4=x4; %为下一次的输入信号做准备
end %大循环结束，产生输入信号 u

%高斯白噪声，长度为 L
v1=0*randn(1,L);
v2=0*randn(1,L);
Y1(2)=0.1;Y1(1)=0.1; %取 Y 的前两个初始值为零
Y2(2)=0.1;Y2(1)=0.1; %取 Y 的前两个初始值为零
for k=3:L
Y1(k)=-0.5*Y1(k-1)+0.2*Y2(k-1)-1.2*Y1(k-2)+0.6*Y2(k-2)+u1(k)+0.5*u1(k-1)-0.4*u2(k-1)+0.4*
u1(k-2)-0.3*u2(k-2)+v1(k);
Y2(k)=0.3*Y1(k-1)-0.6*Y2(k-1)-0.1*Y1(k-2)+0.6*Y2(k-2)+u2(k)+0.2*u1(k-1)-0.3*u2(k-1)-0.2*u
1(k-2)+0.1*u2(k-2)+v2(k);
end
Y=[Y1;Y2];
U=[u1;u2];
N=150; n=2;
A1=[0.5 -0.2;-0.3 0.6];
A2=[1.2 -0.6;0.1 -0.6];
B0=[1.0 0.0;0.0 1.0];
B1=[0.5 -0.4;0.2 -0.3];
```

```
B2=[0.4 -0.3;-0.2 0.1];
c0=zeros(10,1); %被辨识参数矩阵的初始值
c0 =[3 3 3 3 3 3 3 3 3 3];
c0=[c0;c0];   %构造系数矩阵
H=zeros(N-n,4*n+2);
c=c0;
c=c';
for j=1:2
%重写给 H 赋值的函数
for k=1+n:N   %给每一行赋值
      for i=1:(2*n+1) %给每两列赋值
            if i<=2
            H(k-n,2*i-1:2*i)=-Y(:,k-i)';
            else
            H(k-n,2*i-1:2*i)=U(:,k-i+3)';
            end
      end
end
        tem=H'*H;
        tem1=rank(tem);
        c(:,j)=pinv(tem)*H'*Y(j,1+n:N)';
end
c=c';
y_tem=H*c';
y_tem=y_tem';
disp('矩阵 a1 结果为');
a1=c(1:2,1:2)
disp('矩阵 a2 结果为');
a2=c(1:2,3:4)
disp('矩阵 b0 结果为');
b0=c(1:2,5:6)
disp('矩阵 b1 结果为');
b1=c(1:2,7:8)
disp('矩阵 b2 结果为');
b2=c(1:2,9:10)
```

利用上式，即可求解出 Thita 中的辨识参数。但是在此多变量系统中。由于矩阵 H’*H 不满秩，所以我们只能求矩阵的伪逆，来求得 Thita，求得正确的结果。

辨识结果（有噪声）

```
矩阵a1结果为

a1 =

    0.4930   -0.3939
   -0.2972    0.6760

矩阵a2结果为

a2 =

    1.2509   -0.8208
    0.0800   -0.5134

矩阵b0结果为

b0 =

    1.0547   -0.0992
    0.0068    0.8382

矩阵b1结果为

b1 =

    0.5728   -0.6560
    0.2478   -0.2121
```

```
矩阵b2结果为

b2 =

    0.3414   -0.2691
   -0.1510    0.1108
```

辨识结果（无噪声）

```
a1 =

    0.5000   -0.2000
   -0.3000    0.6000

矩阵a2结果为

a2 =

    1.2000   -0.6000
    0.1000   -0.6000

矩阵b0结果为

b0 =

    1.0000    0.0000
    0.0000    1.0000

矩阵b1结果为

b1 =

    0.5000   -0.4000
    0.2000   -0.3000
```

```
矩阵b2结果为

b2 =

    0.4000   -0.3000
   -0.2000    0.1000
```

分析上述的辨识结果，辨识的结果大致符合模型的初始参数，造成辨识结果和原结果差距较大的主要原因在于同分布的随机噪声，多变量最小二系统辨识过程中忽略了系统噪声对于辨识结果的影响，这使得辨识结果存在误差。如果在编程时消除噪声干扰，则辨识结果几乎与初始结果符合。