ELEC 574 Project

Jacob Morrison

April 24th, 2019

General Problem

Start by reading the recently published paper ”Adaptive Air-Fuel Ratio Regulation for Port-Injected Spark-Ignited Engines Based on a Generalized Predictive Control Method” by L. Meng et al.

This project is somewhat inspired by that paper. Here we will only consider the design of an adaptive controller for Φ_{exh}, the engine fuel/air equivalent ratio using the fuel injection mass flowrate as the manipulated variable. We will assume that the system can be described by a CARIMA model:

$$A(q^{-1})y(k) = B(q^{-1})u(k-1) + C(q^{-1})e(k)/\Delta$$

where y is Φ_{exh}, $u \geq 0$ is the fuel injection flow rate, and e is zero-mean white noise with variance $\sigma_e^2 = 0.02^*$, $\Delta = 1 - q^{-1}$ and

$$\begin{aligned}
A(q^{-1}) &= 1 - (e^{-T/\tau_{exh}} + e^{-T/\tau_f})q^{-1} + e^{-(T/\tau_{exh} + T/\tau_f)}q^{-2} \\
B(q^{-1}) &= q^{-c}\left(\frac{1-X}{\dot{m}_{ap}} + \frac{X - e^{-T/\tau_f}}{\dot{m}_{ap}}q^{-1}\right) \\
C(q^{-1}) &= 1
\end{aligned}$$

where
$\dot{m}_{ap}$ = air mass flow into cylinder, kg/s
X = fraction of fuel flow into film
τ_f = fuel film evaporation time constant
τ_{exh} = time constantof gas exhaust
T = sampling time = $0.05s$
$c = \tau_d/T$ where τ_d is the AFR time delay with $3 \leq c \leq 5$

The control objective is to maintain the engine equivalent ratio $\Phi = 1$ during simulations of 1000 samples.

***Note:** Based on my interpretation of the results of the paper ”Adaptive Air-Fuel Ratio Regulation for Port-Injected Spark-Ignited Engines Based on a Generalized Predictive Control Method” as well as my own testing, I believe the paper misreported the standard deviation of the noise used as the variance. A variance of $\sigma_e^2 = 0.02$ means a standard deviation of $\sigma_e \approx 0.14$ which is very large given the situation. With that in mind, I use a standard deviation of $\sigma_e = 0.02$ when answering the questions involved in this project, as I believe that is the noise level that was actually used in the paper, and a standard deviation of $\sigma_e \approx 0.14$ is too large to see any meanigful results.

1 Subproblem One

We will assume that the following parameters are constant: $\dot{m}_{ap} = 15g/s$, $\tau_{exh} = 0.15s$, $\tau_f = 2s$, $X = 0.7$, $c = 4$. Design and simulate an adaptive generalized predictive controller that works well for this nominal set of parameters.

1.1 Problem Set Up

First, the actual values for $A(q^{-1})$ and $B(q^{-1})$ can be solved by plugging in the assumed constants, leading to a time invariant sytem.

$$\begin{aligned}
A(q^{-1}) &= 1 - (e^{-T/\tau_{exh}} + e^{-T/\tau_f})q^{-1} + e^{-(T/\tau_{exh} + T/\tau_f)}q^{-2} \\
&= 1 - (e^{-0.05/0.15} + e^{-0.05/2})q^{-1} + e^{-(0.05/0.15+0.05/2)}q^{-2} \\
&\approx 1 - 1.692q^{-1} + 0.699q^{-2} \\
B(q^{-1}) &= q^{-c}\left(\frac{1-X}{\dot{m}_{ap}} + \frac{X - e^{-T/\tau_f}}{\dot{m}_{ap}}q^{-1}\right) \\
&= q^{-4}\left(\frac{1-0.7}{15} + \frac{0.7 - e^{-0.05/2}}{15}q^{-1}\right) \\
&\approx q^{-4}(0.02 - 0.0184q^{-1})
\end{aligned}$$

Note that approximate values for the coefficients are shown above, but exact values will be used in all simulations. Further, when it comes to designing a generalized predictive controller (GPC) it is necessary to select values for N_1, N_2, and N_u, which are the minimum prediction horizon, maximum prediction horizon, and control horizon respectively. In this case the minimum prediction horizon value will be chosen as $N_1 = 4$ to match the time delay, while N_2 and N_u will be left as tuning parameters. Also note that the controller design here will be similiar to the controller design seen in the 3rd course assignment as both cases required the design of a GPC controller.

1.2 Controller Design

In general a GPC is developed by minimizing the following cost function:

$$J(N_1, N_2, N_u) = E\left\{\sum_{j=N_1}^{N_2} (y(k+j) - w(k+j))^2 + \sum_{j=1}^{N_u} \rho\Delta u(k+j-1)^2\right\} \tag{1}$$

where w is the reference signal and ρ is a tuning parameter that regulates large inputs. First though, in order to design a GPC the Diophantine equation must be introduced. In this case it takes the following form:

$$1 = A\Delta F_j(q^{-1}) + q^{-j}G_j(q^{-1}) \tag{2}$$

where j is the number of time steps ahead used for prediction, $\Delta = 1 - q^{-1}$, $\deg(F_j) = j - 1$, and $\deg(G_j) = n - 1$. This equation can be solved recursively using:

$$\begin{aligned}
f_j &= g_0^j \\
g_i^{j+1} &= g_{i+1}^j - \tilde{a}_{i+1}f_j \\
F_{j+1}(q^{-1}) &= F_j(q^{-1}) + f_j q^{-j} \\
F_1 &= 1 \\
G_1 &= q(1 - \tilde{A})
\end{aligned} \tag{3}$$

where f_j is the j^{th} term of F_j, g_i^j is the i^{th} term of G_j, $\tilde{A} = A\Delta$, and $\tilde{a}_i$ is the i^{th} coefficient of $\tilde{A}$. Now, with the ability to solve F_j and G_j, an equation for the response estimate at time $t+j$ can be developed by combining the given CARIMA model with equation (2) to give:

$$\hat{y}(k+j \mid k) = G_j y(k) + R_j \Delta u(k+j-d) \tag{4}$$

where $R_j = BF_j$ and the coefficients of R_j are denoted r_i. Further, the prediction for $y(t+j)$ under the assumption that future control signals are 0 is defined as:

$$\bar{y}(k+j) = G_j y(k) \tag{5}$$

This allows the cost function to be written as:

$$\begin{aligned} J(N_1, N_2, N_u) &= E\left\{(\mathbf{y}-\mathbf{w})^\top(\mathbf{y}-\mathbf{w}) + \rho\mathbf{\Delta u}^\top\mathbf{\Delta u}\right\} \\ &= (\mathbf{R}\mathbf{\Delta u} + \bar{\mathbf{y}} - \mathbf{w})^\top(\mathbf{R}\mathbf{\Delta u} + \bar{\mathbf{y}} - \mathbf{w}) + \rho\mathbf{\Delta u}^\top\mathbf{\Delta u} \end{aligned} \tag{6}$$

where

$$\begin{aligned} \mathbf{y} &= [\hat{y}(k+N_1 \mid k), ..., \hat{y}(k+N_2 \mid k)]^\top \\ \mathbf{\Delta u} &= [\Delta u(k+N_1-d), ..., \hat{y}(k+N_2-d)]^\top \\ \bar{\mathbf{y}} &= [\bar{y}(k+N_1), ..., \bar{y}(k+N_2)]^\top \\ \mathbf{w} &= [w(k+N_1), ..., w(k+N_2)]^\top \end{aligned}$$

and $\mathbf{R}$ is the dynamic matrix of the system which consists of the coefficients of R_j, has dimensions $(N_2 - N_1)$xN_u, and is defined in this case as:

$$\mathbf{R} = \begin{bmatrix} r_{N_1-1} & r_{N_1-2} & \dots & r_0 & 0 & \dots & \dots & 0 \\ r_{N_1} & r_{N_1-1} & \dots & \dots & r_0 & 0 & \dots & 0 \\ r_{N_1+1} & \vdots & \dots & \dots & \dots & r_0 & 0 & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & r_0 \\ r_{N_2-1} & r_{N_2-2} & \dots & \dots & \dots & \dots & \dots & r_{N_2-N_u} \end{bmatrix}$$

Finally, by minimizing equation (6) with respect to $\mathbf{\Delta u}$ the optimal change in control input is determined to be:

$$\mathbf{\Delta u} = [\mathbf{R}^\top\mathbf{R} + \rho]^{-1}\mathbf{R}^\top(\mathbf{w} - \bar{\mathbf{y}}) \tag{7}$$

Which leads to the following control update:

$$u(k) = u(k-1) + \Delta u(k+N_1-d) \tag{8}$$

Note that the system input cannot be negative ($u \geq 0$) so any values calculated as negative will simply be set to 0.

1.3 Recursive Parameter Estimation

In order to make the GPC adaptive, it is necessary to estimate the model parameters at each time step. Since the system is time invariant and $C(q^{-1}) = 1$, this will be done using a standard recursive least squares method. The estimated parameters will then be used at each time step to specifiy the control input, while the given model parameters will only be used to simulate the plant.

1.4 Simulation Results

The performance of the controller was explored by putting it through the simulation outlined in the problem. Numerous variations of tuning parameters N_u, N_2, and ρ were examined, with the best result displayed below. This result featured the values $N_u = 3$, $N_2 = 8$, and $\rho = 0.1$. Note that while this case led to the best control, it is less computationally efficient than the case when $N_u = 1$. Overall, the controller is able to maintain the system output (Φ_{exh}) at approximately the reference value of 1 quite well. Further, parameter estimates converged to approximately the actual parameter values after about 100 samples. The Matlab code for this simulation is displayed in section 5.1.

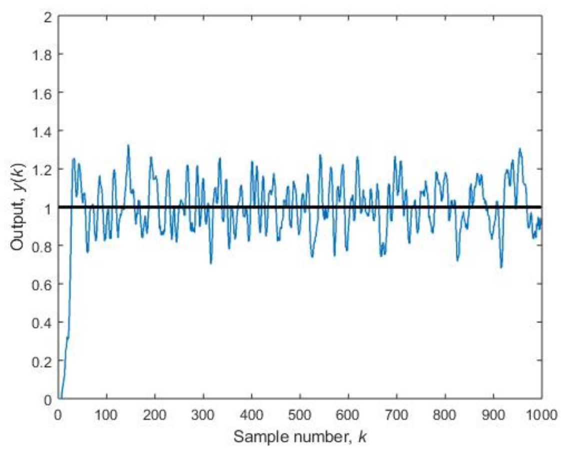

Figure 1: System output over time compared to the reference signal. The output is the engine fuel/air equivalent ratio Φ of the system

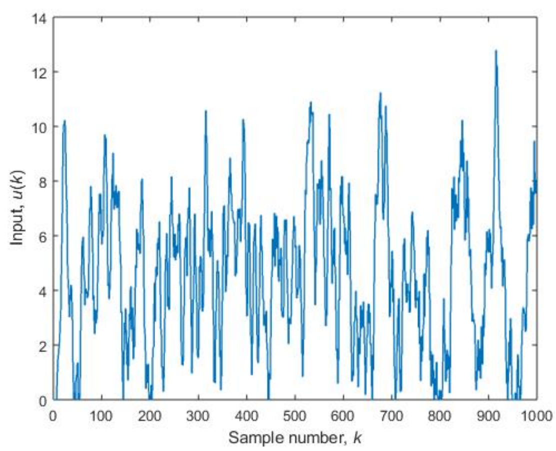

Figure 2: Input to the sytem over time. The input is the fuel injection mass flowrate in g/s

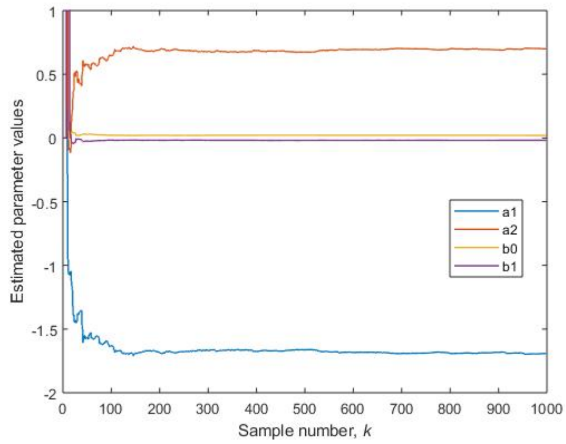

Figure 3: Model parameter convergence over time

2 Subproblem Two

Simulate the nominal adaptive controller designed above for the situations $c = 3$ and $c = 5$. Make any change in the design parameters that you deem necessary to ensure robustness.

2.1 Problem Set Up

The problem set up and controller design in this case are essentially the same as in the previous subproblem. There are only 2 slight changes that will be made in order to ensure robustness. Firstly, N_1 will be set to 3 instead of 4 to account for the minimum possible input delay. Secondly, the order of the recursive estimate for $B(q^{-1})$ will be increased from 2 to 4 so as to include all possible time delay values in the system ($3 \leq c \leq 5$), which is a common practice when the delay is unknown.

2.2 Simulation Results

The robustness of the adaptive controller was explored by putting it through two simulations relating to the situations $c = 3$ and $c = 5$. Here the values for N_u, N_2, and ρ were simply taken from the previous subproblem, with the results displayed below. As a whole, the controller appears quite robust to the various input delays and is able to maintain the system output (Φ_{exh}) at approximately the reference value of 1, both when $c = 3$ and $c = 5$. It does perform better when $c = 3$, but this makes sense as lesser time delays are easier to control. Additionally, the parameter estimates for the coefficients of $A(q^{-1})$ converged to approximately their actual values rather quickly in both cases, while the estimates for the coefficients of $B(q^{-1})$ converged quite fast but are different than the actual values due to the increased order of $B(q^{-1})$ in the recursive estimation. The Matlab code for these simulations is identical to the code displayed in section 5.1 except $N_1 = 3$, the order of the estimate for $B(q^{-1})$ was set to 4 ($nb = 4$), c was set to 3 for one simulation and 5 for the other, and line 107 was commented out while lines 98-104 were uncommented.

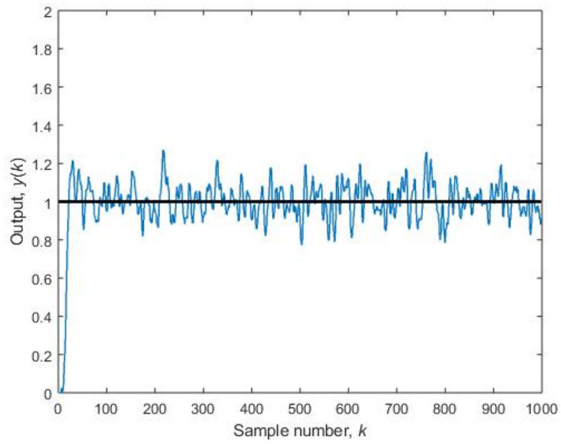

Figure 4: System output over time compared to the reference signal when $c = 3$. The output is the engine fuel/air equivalent ratio Φ of the system

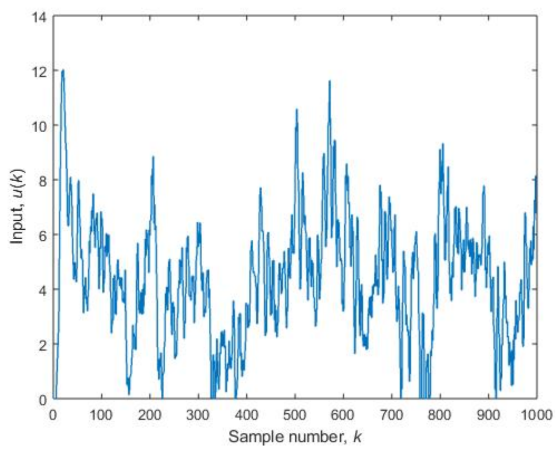

Figure 5: Input to the sytem over time when $c = 3$. The input is the fuel injection mass flowrate in g/s

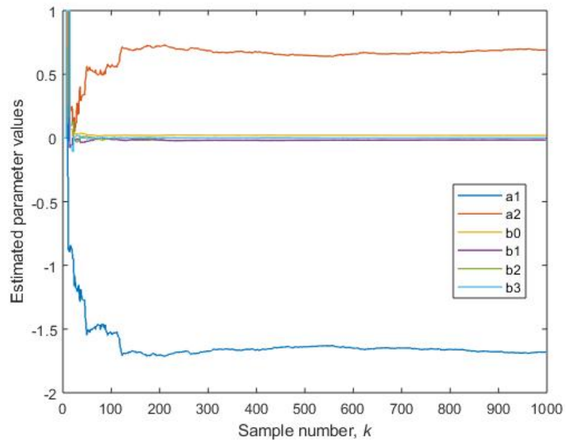

Figure 6: Model parameter estimates over time when $c = 3$

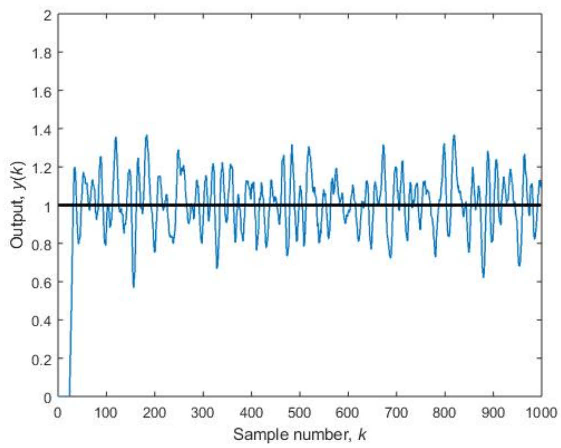

Figure 7: System output over time compared to the reference signal when $c = 5$. The output is the engine fuel/air equivalent ratio Φ of the system

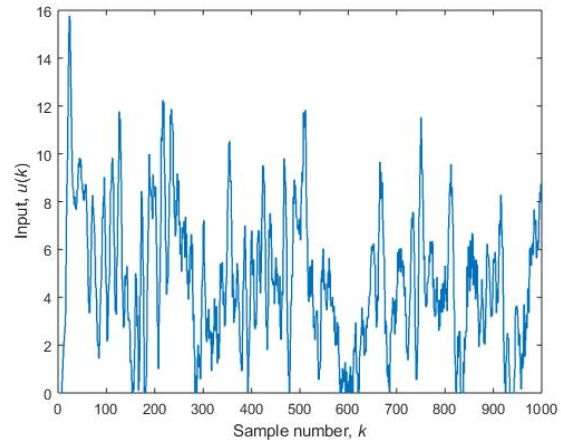

Figure 8: Input to the sytem over time when $c = 5$. The input is the fuel injection mass flowrate in g/s

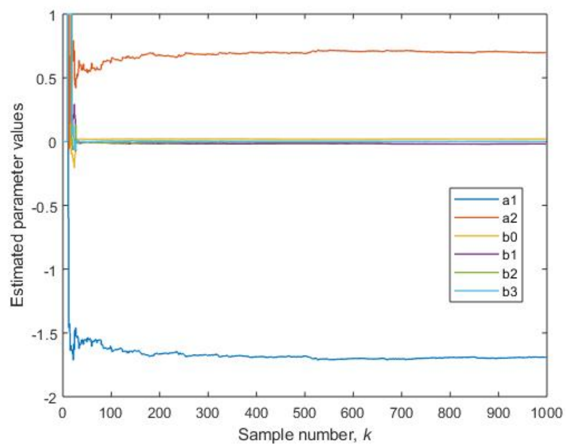

Figure 9: Model parameter estimates over time when $c = 5$

3 Subproblem Three

We will now assume that X is time-varying and can be described as a sine wave of peak-to-peak amplitude of 0.2 and period of 200 samples around its nominal value of 0.7. We will also assume that $m_{ap} = 15g/s$ except for: samples 200-220: $m_{ap} = 20g/s$; samples 400-420: $m_{ap} = 30g/s$; samples 600-620: $m_{ap} = 40g/s$; Simulate your adaptive controller for this situation when $c = 4$.

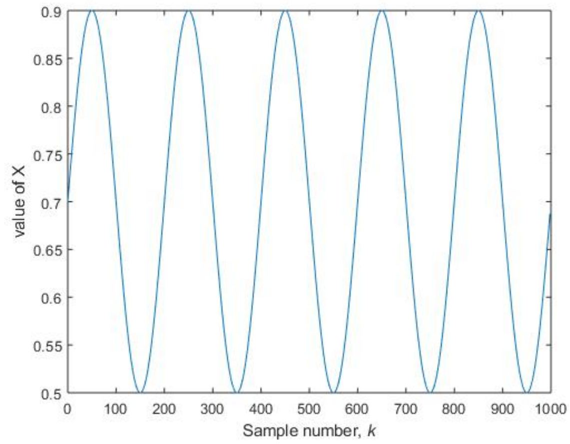

Figure 10: variation in X value over time

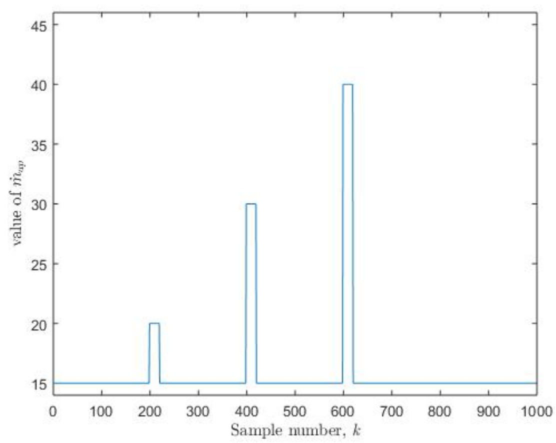

Figure 11: variation in $\dot{m}_{ap}$ value over time

3.1 Problem Set Up

The controller design in this case is the same as in subproblem one. The set up of $B(Q^{-1})$ changes slightly here though, as $B(Q^{-1})$ is a function of X and $\dot{m}_{ap}$, which now both change over time. This means that the system is now time varying, so in order to make the adaptive GPC effective a new recursive estimation technique will be required.

3.2 Recursive Parameter Estimation

Due to the time varying nature of the system, the recursive parameter estimation will be accomplished using the Exponential Forgetting and Resetting Algorithm (EFRA). The EFRA allows the estimation to focus on recent data, without considering older data that is no longer representative of the process. There is a version of RLS that includes a forgetting factor that also focuses on recent data, though EFRA is more effective as the convariance matrix P will never grow exponentially as it is bounded on both sides at all times. The estimate update process with EFRA works similiarly to that of RLS, except with a few extra tuning parameters that are detailed below.

$$
\begin{aligned}
K(k+1) &= \frac{P(t)x(k+1)}{\lambda + x^\top p(k+1)P(k)x(k+1)} \\
P(k+1) &= \frac{1}{\lambda}\left(P(k) - \frac{P(k)x(k+1)x^\top(k+1)P(k)}{\lambda + x^\top(k+1)P(k)x(k+1)}\right) + \beta I - \gamma P(k)^2 \\
\hat{\theta}(k+1) &= \hat{\theta}(k) + \alpha K(k+1)\left(y(k+1) - x^\top(k+1)\hat{\theta}(k)\right)
\end{aligned}
$$

Here x is the same as in RLS, λ is the forgetting factor (also featured in the forgetting factor version of RLS) , and α, β, and γ are additional parameters that can be tuned. As before, the estimated parameters will be used at each time step to specifiy the control input, while the given model parameters will ony be used to simulate the plant.

3.3 Simulation Results

The performance of the controller with the updated parameter estimation was explored by putting it through the same simulation as in subproblem one, except this time with the time varying values for X and $\dot{m}_{ap}$. Numerous variations of tuning parameters N_u, N_2, and ρ were again examined, with the best result displayed below. In this case the best result featured the values $N_u = 3$, $N_2 = 12$, and $\rho = 0.15$. Additionally, values of $\lambda = 0.95$, $\alpha = 0.9$, and $\beta = \gamma = 0.001$ were selected through trial and error for the recursive parameter estimation. The controller seems to perform reasonably well, though not quite as well as the controller in subproblem one. This makes sense seeing as time varying systems are typically harder to control. Further, parameter estimates vary a lot more than in the previous subproblems due to the EFRA estimation technique. It is intersting to note that in reality only $B(q^{-1})$ is time varying, but the EFRA leads to increased variation in the estimates for the coefficients of $A(q^{-1})$ as well. Nonetheless, this does not seem to hinder controller performance. The Matlab code for this simulation is displayed in section 5.2.

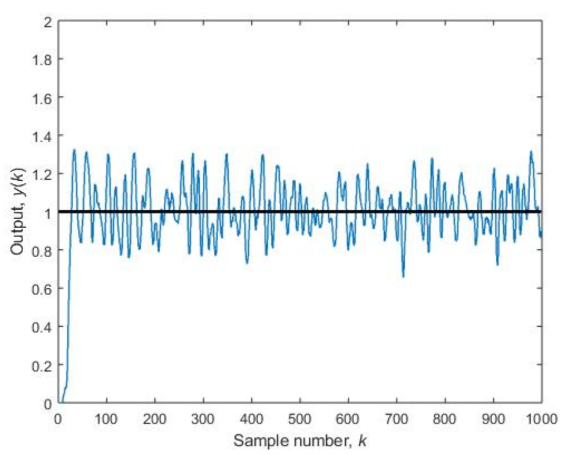

Figure 12: System output over time compared to the reference signal. The output is the engine fuel/air equivalent ratio Φ of the system

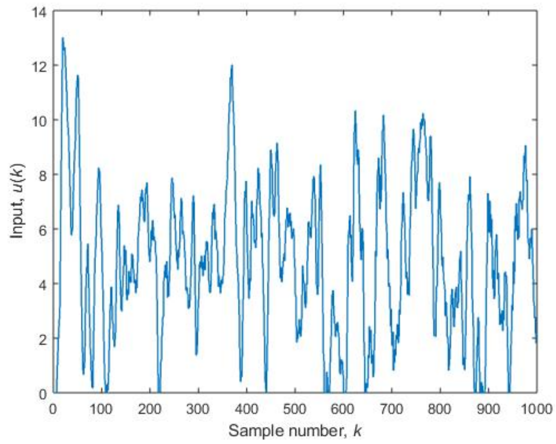

Figure 13: Input to the sytem over time. The input is the fuel injection mass flowrate in g/s

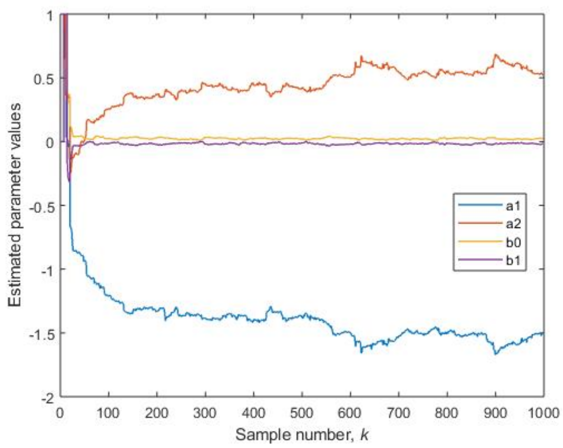

Figure 14: Model parameter convergence over time

4 Subproblem Four

Feeling brave, we will now simulate the above situation but adding variations for the delay c: samples 1-200: $c = 4$; samples 200-500: $c = 3$; samples 500-800: $c = 5$; samples 800-1000: $c = 4$;

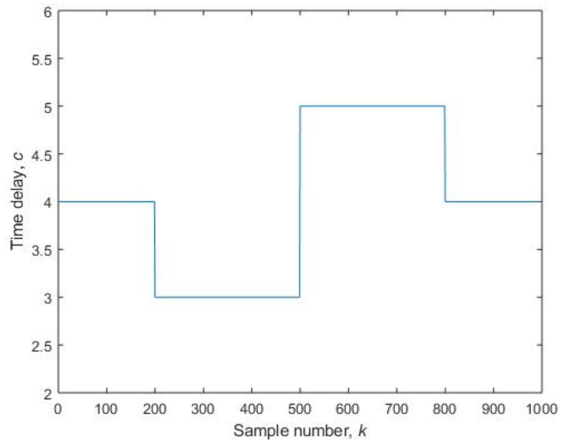

Figure 15: variation in time delay c over time

4.1 Problem Set Up

The problem set up and controller design in this case are very similar to the previous subproblem. The differences here are the same as the differences between subproblem one and two. N_1 will again be set as 3 to account for the minimum possible input delay while the order of the recursive estimate for $B(q^{-1})$ will be set to 4 so as to include all possible time delay values in the system ($3 \leq c \leq 5$).

4.2 Simulation Results

The performance of the previously designed GPC was examined by carrying out the simulation described in this subproblem. Here the values for N_u, N_2, and ρ were again taken from the previous subproblem, with the results displayed below. As a whole, the controller appears quite robust to the variations in c, never allowing the system output to stray too far from the reference. This leads to the positive conclusion that adaptive GPC with EFRA parameter estimation represents a solid control option for time varying single-input single-output systems, even when the time delay is both unknown and variable. The Matlab code for this simulation is displayed in section 5.3.

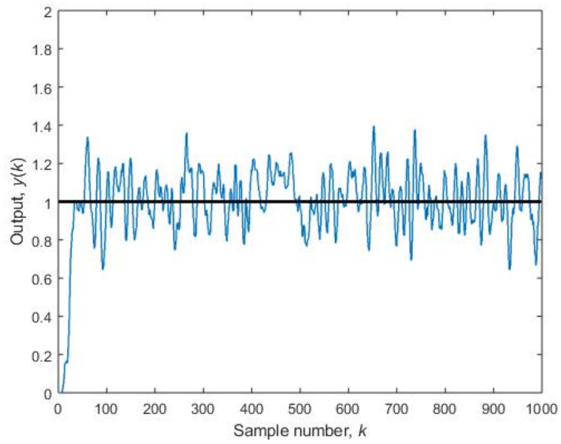

Figure 16: System output over time compared to the reference signal when the time delay is varying. The output is the engine fuel/air equivalent ratio Φ of the system

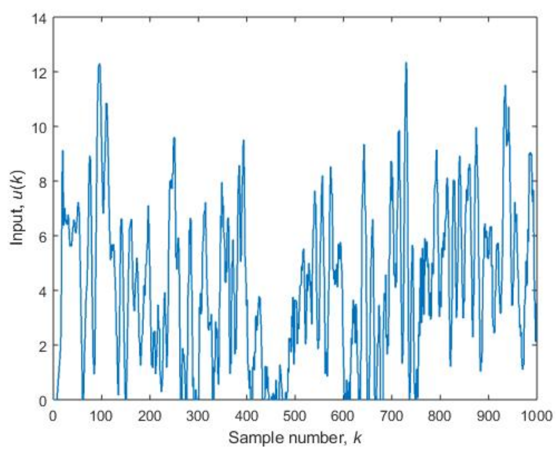

Figure 17: Input to the sytem over time when the time delay is varying. The input is the fuel injection mass flowrate in g/s

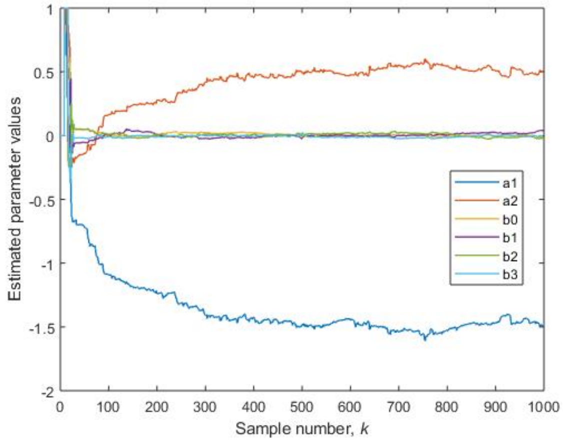

Figure 18: Model parameter estimates over time when the time delay is varying

5 Matlab Code

5.1 Code for Subproblem One and Two

The code displayed below was used for subproblem one. Note that the code for subproblem two is identical, save for the minor differences outlined in section 2.2.

```
clc;
clear;
close all;

%set up vars
map = 15; %g/s
texh = 0.15; %s
tf = 2; %s
X = 0.7;
T = 0.05; %sampling time in seconds

%set up reference
t= (0:1200)';
N = length(t);
yr = zeros(N,1);
yr(1:N) = 1;

%set up actual model, estimated model, control params
%actual model
A = [ (exp( T/texh)+exp( T/tf)), exp( (T/texh+T/tf))]';
B = [(1 X)/(map), (X exp( T/tf))/map]';
c = 4;
e = random('norm', 0, 0.02, 1 , N);

%initialize estimated model
na = 2;
nb = 2;
```

```
nk = c; %input delay
nparams = na+nb; %6 parameters to be estimated (a1,a2,a3,b0,b1,b2)

ts = max([na nb+nk])+1; %sample number to start iterations at

u = zeros(N,1);
y = zeros(N,1);

%set up RLS components
theta = [ ones(na,1);ones(nb,1)];
thetavec = zeros(N,nparams);
alpha = 1000;
P = alpha*eye(nparams);
x = zeros(nparams,1);

%control params
N1  = 4;
N2  = 8;
Nu  = 3;
rho = 0.1;
%alpha2 = 0;
te = N N2; %sample number to stop iterations at
ybar = zeros(N2 N1+1,1);
Adelta = zeros(na+1,1);
%w = zeros(N2 N1+1,1);
G = zeros(N2 N1+1,na+1);
f = zeros(N2,1);
r = zeros(length(f)+1,1);
R_tmp = zeros(length(r)+Nu 1 ,Nu);
R = zeros(N2 N1+1,Nu);

%simulation and recursive estimation
%comment out the following line to leave noise in the system
%e = zeros(1,length(e));

for i = ts:te
    %test first without estimating params
    %Aest = A;
    %Best = B;
    %get estimated parameters from theta:
    Aest =  theta(1:na);
    Best = theta(na+1:nparams);

    %get Adelta from Aest
    for j = 1:na+1
        if j == 1
            Adelta(j) = Aest(j) 1;
        elseif j == na+1
            Adelta(j) =  Aest(j 1);
        else
            Adelta(j) = Aest(j) Aest(j 1);
```

```
        end
    end

    %solve diophantine equation
    G(1,:) =  Adelta';
    f(1)    = G(1,1);
    r(1)   = Best(1);
    r(2)    = Best(2)+Best(1)*f(1);
    for j = 1:N2 1
        for k = 1:na+1
            if k == na+1
                G(j+1,k) =  Adelta(k)*f(j);
            else
                G(j+1,k) = G(j,k+1) Adelta(k)*f(j);
            end
        end
        f(j+1) = G(j+1,1);
        %set up R matrix coefficients
        %if order(B) = 4
%         if j == 1
%             r(j+2) = Best(3)+Best(2)*f(j)+Best(1)*f(j+1);
%         elseif j==2
%             r(j+2) = Best(4)+Best(3)*f(j 1)+Best(2)*f(j)+Best(1)*f(j+1);
%         else
%             r(j+2) = Best(4)*f(j 2)+Best(3)*f(j 1)+Best(2)*f(j)+Best(1)*f(j+1);
%         end

        %if order(B) = 2
        r(j+2) = Best(2)*f(j)+Best(1)*f(j+1);
    end
    F = [1;f];

    %set up dynamic matrix R
    for j = 1:Nu
        R_tmp(j:j+length(r) 1,j) = r;
    end
    R = R_tmp(N1:N2,:);

    %simulate plant output
    y(i) =  A(1)*y(i 1) A(2)*y(i 2)+ B(1)*u(i c 1) ...
        +B(2)*u(i c 2)+e(i);

    %investigate altered reference values from paper
    %for j = N1:N2
        %w(j N1+1) = alpha2*y(i)+(1 alpha2)*yr(i);
    %end
    %get simple future reference values
    w = yr((i+N1):(i+N2));

    %predict future outputs
```

```
    for j = N1:N2
        ybar(j N1+1) = G(j,:)*y(i: 1:i 2);
    end

    %update control action
    Rmat = R'*R+rho*eye(min(size(R)));
    delta_u = Rmat\(R')*(w ybar);
    u(i) = u(i 1)+delta_u(1);
    if u(i)<0
        u(i)=0;
    end

    %update parameter estimates
    for j = 1:na
        x(j) = y(i j);
    end
    for j = 1:nb
        x(na+j) = u(i nk j);
    end
    K = (P*x)/(1+x'*P*x);
    P = P ((P*(x*x')*P)/(1+x'*P*x));
    theta = theta+K*(y(i) x'*theta);
    thetavec(i,:) = [Aest' Best'];
end

%Plot results
val=999;
figure(2)
stairs(t(1:val),u(1:val),'LineWidth',1);
ylabel({'Input, \itu\rm(\itk\rm)'});
xlabel({'Sample number, \itk'});
figure(3);
stairs1 = stairs(t(1:val),thetavec(1:val,:),'LineWidth',1);
set(stairs1(1),'DisplayName','a1');
set(stairs1(2),'DisplayName','a2');
set(stairs1(3),'DisplayName','b0');
set(stairs1(4),'DisplayName','b1');
%set(stairs1(5),'DisplayName','b2');
%set(stairs1(6),'DisplayName','b3');
ylabel({'Estimated parameter values'});
xlabel({'Sample number, \itk'});
legend('show');
figure(1);
stairs2 = stairs(t(1:val),[y(1:val),yr(1:val)]);
set(stairs2(1),'LineWidth',1);
set(stairs2(2),'LineWidth',2,'Color',[0 0 0]);
ylabel({'Output, \ity\rm(\itk\rm)'});
xlabel({'Sample number, \itk'});
ylim([0 2]);
```

5.2 Code for Subproblem Three

```
clc;
clear;
close all;

%set up reference
t= (0:1200)';
N = length(t);
yr = zeros(N,1);
yr(1:N) = 1;

%set up vars
map = 15*ones(N,1); %g/s
map(200:220) = 20*ones(21,1);
map(400:420) = 30*ones(21,1);
map(600:620) = 40*ones(21,1);
texh = 0.15; %s
tf = 2; %s
X = 0.7+0.2*sin(t*pi/100);
T = 0.05; %sampling time in seconds

%set up actual model, estimated model, control params
%actual model
A = [ (exp( T/texh)+exp( T/tf)), exp( (T/texh+T/tf))]';
B = [(1 X)./(map), (X exp( T/tf))./map]';
c = 4;
e = random('norm', 0, 0.02, 1 , N);

%initialize estimated model
na = 2;
nb = 2;
nk = c; %input delay
nparams = na+nb; %6 parameters to be estimated (a1,a2,a3,b0,b1,b2)

%set up parameter estimation constants
lambda = 0.95; %forgetting factor
alpha3 = 0.9;
beta = 0.001;
gamma = 0.001;

ts = max([na nb+nk])+2; %sample number to start iterations at

u = zeros(N,1);
y = zeros(N,1);

%set up RLS components
theta = [ ones(na,1);ones(nb,1)];
thetavec = zeros(N,nparams);
alpha = 1000;
P = alpha*eye(nparams);
x = zeros(nparams,1);
```

```
%control params
%3, 8, 2, 0.1
%4, 12, 3, 0.4, 0.2
N1  = 4;
N2  = 12;
Nu  = 3;
rho = 0.15;
%alpha2 = 0;
te = N N2; %sample number to stop iterations at
ybar = zeros(N2 N1+1,1);
Adelta = zeros(na+1,1);
%w = zeros(N2 N1+1,1);
G = zeros(N2 N1+1,na+1);
f = zeros(N2,1);
r = zeros(length(f)+1,1);
R_tmp = zeros(length(r)+Nu 1 ,Nu);
R = zeros(N2 N1+1,Nu);

%simulation and recursive estimation
%comment out the following line to leave noise in the system
%e = zeros(1,length(e));

for i = ts:te
    %test first without estimating params
    %Aest = A;
    %Best = B(:,i);
    %get estimated parameters from theta:
    Aest =  theta(1:na);
    Best = theta(na+1:nparams);

    %get Adelta from Aest
    for j = 1:na+1
        if j == 1
            Adelta(j) = Aest(j) 1;
        elseif j == na+1
            Adelta(j) =  Aest(j 1);
        else
            Adelta(j) = Aest(j) Aest(j 1);
        end
    end

    %solve diophantine equation
    G(1,:) =  Adelta';
    f(1)   = G(1,1);
    r(1)  = Best(1);
    r(2)   = Best(2)+Best(1)*f(1);
    for j = 1:N2 1
        for k = 1:na+1
            if k == na+1
                G(j+1,k) =  Adelta(k)*f(j);
            else
```

```
                G(j+1,k) = G(j,k+1) Adelta(k)*f(j);
            end
        end
        f(j+1) = G(j+1,1);
        %set up R matrix coefficients
        r(j+2) = Best(2)*f(j)+Best(1)*f(j+1);
    end
    F = [1;f];

    %set up dynamic matrix R
    for j = 1:Nu
        R_tmp(j:j+length(r) 1,j) = r;
    end
    R = R_tmp(N1:N2,:);

    %simulate plant output
    y(i) =  A(1)*y(i 1) A(2)*y(i 2)+ B(1)*u(i c 1)...
        +B(2)*u(i c 2)+e(i);

    %investigate altered reference values from paper
    %for j = N1:N2
        %w(j N1+1) = alpha2*y(i)+(1 alpha2)*yr(i);
    %end
    %get simple future reference values
    w = yr((i+N1):(i+N2));

    %predict future outputs
    for j = N1:N2
        ybar(j N1+1) = G(j,:)*y(i: 1:i 2);
    end

    %update control action
    Rmat = R'*R+rho*eye(min(size(R)));
    delta_u = Rmat\(R')*(w ybar);
    u(i) = u(i 1)+delta_u(1);
    if u(i)<0
        u(i)=0;
    end

    %update parameter estimates
    for j = 1:na
        x(j) = y(i j);
    end
    for j = 1:nb
        x(na+j) = u(i nk j);
    end
    %RLS
%     K = (P*x)/(lambda+x'*P*x);
%     P = (P ((P*(x*x')*P)/(lambda+x'*P*x)))*(1/lambda);
%     theta = theta+K*(y(i) x'*theta);
%     thetavec(i,:) = [Aest' Best'];
```

```

    %EFRA
    K = (P*x)/(lambda+x'*P*x);
    P = (P ((P*(x*x')*P)/(lambda+x'*P*x)))*(1/lambda)+...
        beta*eye(nparams) gamma*P*P;
    theta = theta+alpha3*K*(y(i) x'*theta);
    thetavec(i,:) = [Aest' Best'];
end

%Plot results
val=999;
figure(2)
stairs(t(1:val),u(1:val),'LineWidth',1);
ylabel({'Input, \itu\rm(\itk\rm)'});
xlabel({'Sample number, \itk'});
figure(3);
stairs1 = stairs(t(1:val),thetavec(1:val,:),'LineWidth',1);
set(stairs1(1),'DisplayName','a1');
set(stairs1(2),'DisplayName','a2');
set(stairs1(3),'DisplayName','b0');
set(stairs1(4),'DisplayName','b1');
ylabel({'Estimated parameter values'});
xlabel({'Sample number, \itk'});
legend('show');
figure(1);
stairs2 = stairs(t(1:val),[y(1:val),yr(1:val)]);
set(stairs2(1),'LineWidth',1);
set(stairs2(2),'LineWidth',2,'Color',[0 0 0]);
ylabel({'Output, \ity\rm(\itk\rm)'});
xlabel({'Sample number, \itk'});
ylim([0 2]);
```

5.3 Code for Subproblem Four

```
clc;
clear;
close all;

%set up reference
t= (0:1200)';
N = length(t);
yr = zeros(N,1);
yr(1:N) = 1;

%set up vars
map = 15*ones(N,1); %g/s
map(200:220) = 20*ones(21,1);
map(400:420) = 30*ones(21,1);
map(600:620) = 40*ones(21,1);
texh = 0.15; %s
tf = 2; %s
X = 0.7+0.2*sin(t*pi/100);
```

```
T = 0.05; %sampling time in seconds

%set up actual model, estimated model, control params
%actual model
A = [ (exp( T/texh)+exp( T/tf)), exp( (T/texh+T/tf))]';
B = [(1 X)./(map), (X exp( T/tf))./map]';
c = 4*ones(N,1);
c(201:500) = 3*ones(300,1);
c(501:800) = 5*ones(300,1);
e = random('norm', 0, 0.02, 1 , N);

%initialize estimated model
na = 2;
nb = 4;
nk = 3; %input delay
nparams = na+nb; %6 parameters to be estimated (a1,a2,a3,b0,b1,b2)

%set up parameter estimation constants
lambda = 0.95; %forgetting factor
alpha3 = 0.9;
beta = 0.001;
gamma = 0.001;

ts = max([na nb+nk])+2; %sample number to start iterations at

u = zeros(N,1);
y = zeros(N,1);

%set up RLS components
theta = [ ones(na,1);ones(nb,1)];
thetavec = zeros(N,nparams);
alpha = 1000;
P = alpha*eye(nparams);
x = zeros(nparams,1);

%control params
%3, 8, 2, 0.1
%4, 12, 3, 0.4, 0.2
N1  = 4;
N2  = 12;
Nu  = 3;
rho = 0.15;
%alpha2 = 0;
te = N N2; %sample number to stop iterations at
ybar = zeros(N2 N1+1,1);
Adelta = zeros(na+1,1);
%w = zeros(N2 N1+1,1);
G = zeros(N2 N1+1,na+1);
f = zeros(N2,1);
r = zeros(length(f)+1,1);
R_tmp = zeros(length(r)+Nu 1 ,Nu);
```

```
R = zeros(N2 N1+1,Nu);

%simulation and recursive estimation
%comment out the following line to leave noise in the system
%e = zeros(1,length(e));

for i = ts:te
    %test first without estimating params
    %Aest = A;
    %Best = B(:,i);
    %get estimated parameters from theta:
    Aest =  theta(1:na);
    Best = theta(na+1:nparams);

    %get Adelta from Aest
    for j = 1:na+1
        if j == 1
            Adelta(j) = Aest(j) 1;
        elseif j == na+1
            Adelta(j) =  Aest(j 1);
        else
            Adelta(j) = Aest(j) Aest(j 1);
        end
    end

    %solve diophantine equation
    G(1,:) =  Adelta';
    f(1)    = G(1,1);
    r(1)   = Best(1);
    r(2)    = Best(2)+Best(1)*f(1);
    for j = 1:N2 1
        for k = 1:na+1
            if k == na+1
                G(j+1,k) =  Adelta(k)*f(j);
            else
                G(j+1,k) = G(j,k+1) Adelta(k)*f(j);
            end
        end
        f(j+1) = G(j+1,1);
        %set up R matrix coefficients
        if j == 1
            r(j+2) = Best(3)+Best(2)*f(j)+Best(1)*f(j+1);
        elseif j==2
            r(j+2) = Best(4)+Best(3)*f(j 1)+Best(2)*f(j)+Best(1)*f(j+1);
        else
            r(j+2) = Best(4)*f(j 2)+Best(3)*f(j 1)+Best(2)*f(j)+Best(1)*f(j+1)
                ;
        end
    end
    F = [1;f];

```

```
    %set up dynamic matrix R
    for j = 1:Nu
        R_tmp(j:j+length(r) 1,j) = r;
    end
    R = R_tmp(N1:N2,:);

    %simulate plant output
    y(i) =  A(1)*y(i 1) A(2)*y(i 2)+ B(1)*u(i c(i) 1)...
        +B(2)*u(i c(i) 2)+e(i);

    %investigate altered reference values from paper
    %for j = N1:N2
        %w(j N1+1) = alpha2*y(i)+(1 alpha2)*yr(i);
    %end
    %get simple future reference values
    w = yr((i+N1):(i+N2));

    %predict future outputs
    for j = N1:N2
        ybar(j N1+1) = G(j,:)*y(i: 1:i 2);
    end

    %update control action
    Rmat = R'*R+rho*eye(min(size(R)));
    delta_u = Rmat\(R')*(w ybar);
    u(i) = u(i 1)+delta_u(1);
    if u(i)<0
        u(i)=0;
    end

    %update parameter estimates
    for j = 1:na
        x(j) = y(i j);
    end
    for j = 1:nb
        x(na+j) = u(i nk j);
    end
    %RLS with forgetting factor
%       K = (P*x)/(lambda+x'*P*x);
%       P = (P ((P*(x*x')*P)/(lambda+x'*P*x)))*(1/lambda);
%       theta = theta+K*(y(i) x'*theta);
%       thetavec(i,:) = [Aest' Best'];

    %EFRA
    K = (P*x)/(lambda+x'*P*x);
    P = (P ((P*(x*x')*P)/(lambda+x'*P*x)))*(1/lambda)+...
        beta*eye(nparams) gamma*P*P;
    theta = theta+alpha3*K*(y(i) x'*theta);
    thetavec(i,:) = [Aest' Best'];
end
```

```
%Plot results
val=999;
figure(2)
stairs(t(1:val),u(1:val),'LineWidth',1);
ylabel({'Input, \itu\rm(\itk\rm)'});
xlabel({'Sample number, \itk'});
figure(3);
stairs1 = stairs(t(1:val),thetavec(1:val,:),'LineWidth',1);
set(stairs1(1),'DisplayName','a1');
set(stairs1(2),'DisplayName','a2');
set(stairs1(3),'DisplayName','b0');
set(stairs1(4),'DisplayName','b1');
set(stairs1(5),'DisplayName','b2');
set(stairs1(6),'DisplayName','b3');
ylabel({'Estimated parameter values'});
xlabel({'Sample number, \itk'});
legend('show');
figure(1);
stairs2 = stairs(t(1:val),[y(1:val),yr(1:val)]);
set(stairs2(1),'LineWidth',1);
set(stairs2(2),'LineWidth',2,'Color',[0 0 0]);
ylabel({'Output, \ity\rm(\itk\rm)'});
xlabel({'Sample number, \itk'});
ylim([0 2]);
```