Department of Engineering Cybernetics

TTK4215 System Identification and Adaptive Control
Solution 3
Instrumental Variables Method

Problem 1 (Instrumental Variables Method)

a) From

$$\frac{1}{N}\sum_{t=1}^{N}\xi(t)\left[-\varphi^{T}(t)\,\tilde{\theta}_{N}^{IV}+v_{0}(t)\right]=0, \tag{1}$$

we get

$$\tilde{\theta}_{N}^{IV}=\left(\frac{1}{N}\sum_{t=1}^{N}\xi(t)\,\varphi^{T}(t)\right)^{-1}\frac{1}{N}\sum_{t=1}^{N}\xi(t)\,v_{0}(t)\,. \tag{2}$$

Letting $N\to\infty$, we have

$$\lim_{N\to\infty}\tilde{\theta}_{N}^{IV}=\left(\lim_{N\to\infty}\frac{1}{N}\sum_{t=1}^{N}\xi(t)\,\varphi^{T}(t)\right)^{-1}\left(\lim_{N\to\infty}\frac{1}{N}\sum_{t=1}^{N}\xi(t)\,v_{0}(t)\right). \tag{3}$$

As usual we are assuming quasi-stationary processes, so the limits above can be replaced by the expectation operator to obtain

$$\lim_{N\to\infty}\tilde{\theta}_{N}^{IV}=\left(\bar{E}\xi(t)\,\varphi^{T}(t)\right)^{-1}\bar{E}\xi(t)\,v_{0}(t)\,. \tag{4}$$

b) Starting from the equation that gives the least-squares estimate

$$\frac{1}{N}\sum_{t=1}^{N}\varphi(t)\left[y(t)-\varphi^{T}(t)\,\hat{\theta}_{N}^{LS}\right]=0, \tag{5}$$

the IV-method simply replaces the first instance of the regression vector with the vector of instrumental variables, to obtain

$$\frac{1}{N}\sum_{t=1}^{N}\xi(t)\left[y(t)-\varphi^{T}(t)\,\hat{\theta}_{N}^{IV}\right]=0. \tag{6}$$

The result is obtained by solving (6) for the IV-estimate $\hat{\theta}_{N}^{IV}$.

c) See solution to Assignment 2.

d) The following MATLAB code generates the data, and solves this assignment (you were not supposed to generate data, of course).

```
% Set true parameters
a=[0.8;0.15];
b=[0.5;-0.3;0.2];
c=[0.1];

% Generate noise, and input signal.
e=random('Normal',0,1,10000,1);
U=random('Normal',0,1,10000,1);

% Set previous data (u and y).
prevu=[0;0;0];
prevy=[0;0];
preve=[0];
N=length(e);
Y=zeros(N,1);

% Generate output.
for t=1:N,
% Compute current y
   y=-a'*prevy+b'*prevu+e(t)+c'*preve;
% Store it.
   Y(t)=y;
% Set previous data:
   prevu=[U(t);prevu(1:2)];
   prevy=[y;prevy(1)];
   preve=[e(t)];
end

% Store data
save Data_Assignment3 U Y;

%Compute LS-estimate from U and Y.

% Initialize the sums of the LS equation
SUM1=zeros(5,5);
SUM2=zeros(5,1);

% Initilize storage
THETALS=[];
THETAIV=[];

% Time loop.
for t=1:N,
    % Regressor
    if (t==1)
       phi=[0;0;0;0;0];
```

```
    elseif(t==2)
       phi=[-Y(1);0;U(1);0;0];
    elseif(t==3)
       phi=[-Y(2);-Y(1);U(2);U(1);0];
    else
       phi=[-Y(t-1);-Y(t-2);U(t-1);U(t-2);U(t-3)];
    end

    % Compute sums
    SUM1=SUM1+phi*phi';
    SUM2=SUM2+phi*Y(t);

    % If j is larger or equal to 10, compute and store estimate.
    if (t>=10)
       theta=inv(SUM1)*SUM2;
       THETALS=[THETALS,theta];
    end
end

% Plot estimate as function of N.
plot(THETALS');
legend('a1','a2','b1','b2','b3');
grid;
% Display the final estimate
theta
pause;

% Repeat for IV
% First produce x sequence
X=zeros(N,1);

for t=1:N,
% Compute current x
    if (t==1)
       phi=[0;0;0;0;0];
    elseif(t==2)
       phi=[-X(1);0;U(1);0;0];
    elseif(t==3)
       phi=[-X(2);-X(1);U(2);U(1);0];
    else
       phi=[-X(t-1);-X(t-2);U(t-1);U(t-2);U(t-3)];
    end

    if (t>=10)
       x=phi'*THETALS(:,t-9);
    else
```

```
        x=0;
    end
    % Store it.
    X(t)=x;
end

% Time loop.
for t=1:N,
    % Regressor and instrumental variables.
    if (t==1)
       phi=[0;0;0;0;0];
       zeta=[0;0;0;0;0];
    elseif(t==2)
       phi=[-Y(1);0;U(1);0;0];
       zeta=[-X(1);0;U(1);0;0];
    elseif(t==3)
       phi=[-Y(2);-Y(1);U(2);U(1);0];
       zeta=[-X(2);X(1);U(2);U(1);0];
    else
       phi=[-Y(t-1);-Y(t-2);U(t-1);U(t-2);U(t-3)];
       zeta=[-X(t-1);-X(t-2);U(t-1);U(t-2);U(t-3)];
    end

    % Compute sums
    SUM1=SUM1+zeta*phi';
    SUM2=SUM2+zeta*Y(t);

    % If j is larger or equal to 10, compute and store estimate.
    if (t>=10)
       theta=inv(SUM1)*SUM2;
       THETAIV=[THETAIV,theta];
    end
end

% Plot estimate as function of N.
plot(THETAIV');
legend('a1','a2','b1','b2','b3');
grid;
% Display the final estimate
theta
```

Problem 2 (Extremum Seeking)

The following MATLAB code implements the peak seeking algorithm.

```
function [thetaout,etaout,xiout]=ext_seek(y,thetastart,dt)

persistent t theta eta xi omega omegal omegah k a

if isempty(t)
  % Initialize time
  t=0;

  % Initialize states of extremum seeking controller
  theta=thetastart;
  eta=y;
  xi=0;

  % Define perturbation frequency and amplitude
  omega=0.5;
  a=0.02;

  % Define cutoff frequencies for filters:
  omegal=0.05;
  omegah=0.05;

  % Define gain for update of theta.
  k=0.5;
end

% Extremum seeking:
theta_dot=k*xi;
xi_dot=-omegal*xi+omegal*(y-eta)*a*sin(omega*t);
eta_dot=-omegah*eta+omegah*y;

% Euler integration
theta=theta+dt*theta_dot;
xi=xi+dt*xi_dot;
eta=eta+dt*eta_dot;

% Saturation, if reasonable
%theta=max(theta,0);
%theta=min(theta,1);

% Return variables
thetaout=theta+a*sin(omega*t);
etaout=eta;
xiout=xi;

% Update time variable for next call to ext_seek:
t=t+dt;
```