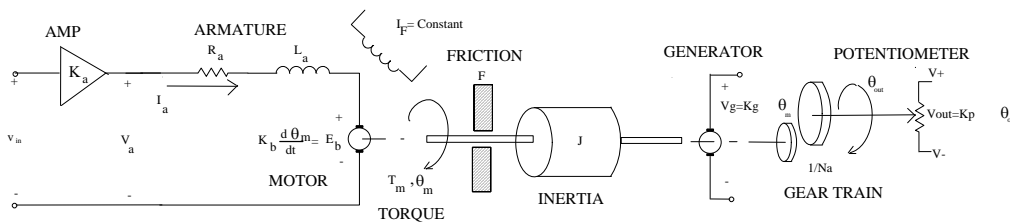


**Objective:** To study the effects of digital observers and to investigate deadbeat strategies

**Prelab:** (Counts as HW#13)

Due Monday, March 8



Again, for Lab 3 we will use the Motomatic DC servo shown above. Recall from Lab 2, we found the following discrete model of the Motomatic:

$$x_{k+1} = \begin{bmatrix} 1 & 0.0302 \\ 0 & 0.9487 \end{bmatrix} x_k + \begin{bmatrix} 0.0104 \\ 0.6627 \end{bmatrix} w_k \text{ where } x_k = \begin{bmatrix} v_{out} \\ v_g \end{bmatrix}$$

where  $V_{out}$  was the voltage off the position sensor, and  $V_g$  is the voltage from the tachometer (generator). Due to U.K.'s budget reductions, we will have to eliminate the tachometer ( $V_g$ ) from the Motomatic. Thus, the only signal we can access is  $V_{out}$ . We will have to build an observer to estimate  $V_g$ .

1. a) Using your controller design from Lab 2, design a deadbeat full-order observer for the Motomatic using  $V_{out}$  as the system output.
  - b) How much time is required until any initial observer error is driven to identically to zero?
  - b) Simulate the observer alone with an initial observer error of  $e_0 = [0.1 \ 0.1]^T$ .
  - c) Now use the result of this simulation to simulate the combined observer/controller again using an initial error of  $e_0 = [0.1 \ 0.1]^T$ . (plot  $x_k$  and  $\hat{x}_k$  vs.  $k$ )

**SEE FOLLOWING PAGE FOR SIMULATION OF OBSERVERS IN MATLAB (or use Simulink and the block diagrams from class).**

  - d) Design an improved deadbeat observer for the Motomatic and repeat parts b) and c).
  - e) Design a reduced-order deadbeat observer for the Motomatic and repeat parts b) and c) (use an initial error of  $z_0 = [0.1]$ ).

(If the deadbeat observers do not yield a satisfactory response, be prepared to redesign!)
2. a) Write a C program to implement your full-order observer design of part 1a) (use in(), out(), and sleep())
  - b) Write a C program to implement your improved observer design of part 1a) (use in(), out(), and sleep())
  - c) Write a C program to implement your reduced-order observer design of part 1a) (use in(), out(), and sleep()).

**In the Lab:** (Counts as HW#14)

Due Wednesday, March 10

1. Test your **full-order observer/controller** design in the Lab. Obtain a plot and compare to Prelab results.
2. Test your **improved observer/controller** design in the Lab. Compare to the Prelab results.
3. Test your **reduced order observer/controller** in the Lab. Compare to the Prelab results.

Matlab simulation of a full order observer using dlsim:

You must define these quantities before executing these commands:

Variables: Ko - observer gain  
Kc - Feedback regulation gain  
Nx, Nu - controller gains  
e0 - initial observer error

```
yref=5*ones(1,101);  
[ye,e]=dlsim(ahat-Ko*chat,[0;0],[0 0],0,ones(1,101),e0);  
wfix=Nu*yref+Kc*(Nx*yref+e);  
[y,x]=dlsim(ahat-bhat*Kc,bhat,chat,0,wfix);  
t=.01*[1:101];  
plot(t,[x x-e])
```

Matlab simulation of an improved observer using dlsim:

You must define these quantities before executing these commands:

Variables: Ko - observer gain  
Kc - Feedback regulation gain  
Nx, Nu - controller gains  
e0 - initial observer error  
Tro = inv([chat;r]) = [Q1 Q2]

```
yref=5*ones(1,101);  
[ye,e]=dlsim(ahat-Ko*chat,[0;0],[0 0],0,ones(1,101),e0);  
wfix=Nu*yref+Kc*(Nx*yref+Q2*r*e);  
[y,x]=dlsim(ahat-bhat*Kc*(Q1*chat+Q2*r),bhat,chat,0,wfix);  
plot(t,[x (Q1*chat*x'+Q2*r*(x-e))])
```

Matlab simulation of a reduced order observer using dlsim:

You must define these quantities before executing these commands:

Variables: Kro - reduced-rder observer gain  
Kc - Feedback regulation gain  
Nx, Nu - controller gains  
Tro = inv([chat;r]) = [Q1 Q2]

```
Aro=a22-kro*a12  
Bro=B2-kro*B1  
Cro=kro*Aro-kro*a11+a21  
q=0.1  
x=[0;0]  
for i=1:101  
z(:,i)=q+kro*chat*x(:,i);  
w=Kc*Nx*5-Kc*(Q1*chat*x(:,i)+Q2*z(:,i))+Nu*5;  
x(:,i+1)=ahat*x(:,i)+bhat*w;  
q=Aro*q+Bro*w+Cro*chat*x(:,i);  
end  
x=x(:,1:101);  
plot(t,[x' (Q1*chat*x+Q2*z)'])
```