# Kalman Filters

## NDSU ECE 463/663

### Lecture #30

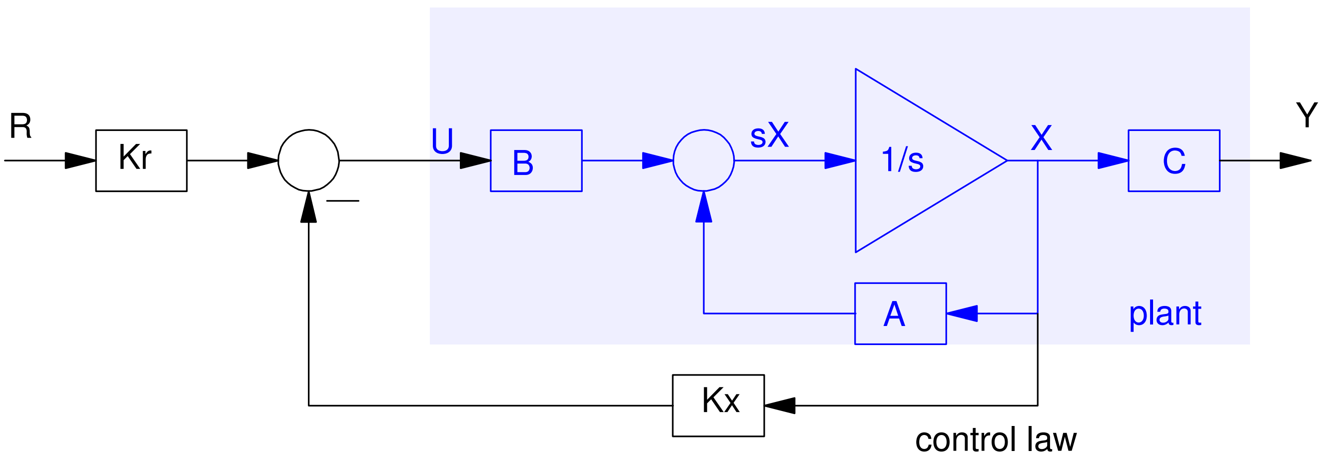
### Inst: Jake Glower

Please visit Bison Academy for corresponding lecture notes, homework sets, and solutions

# Recap:

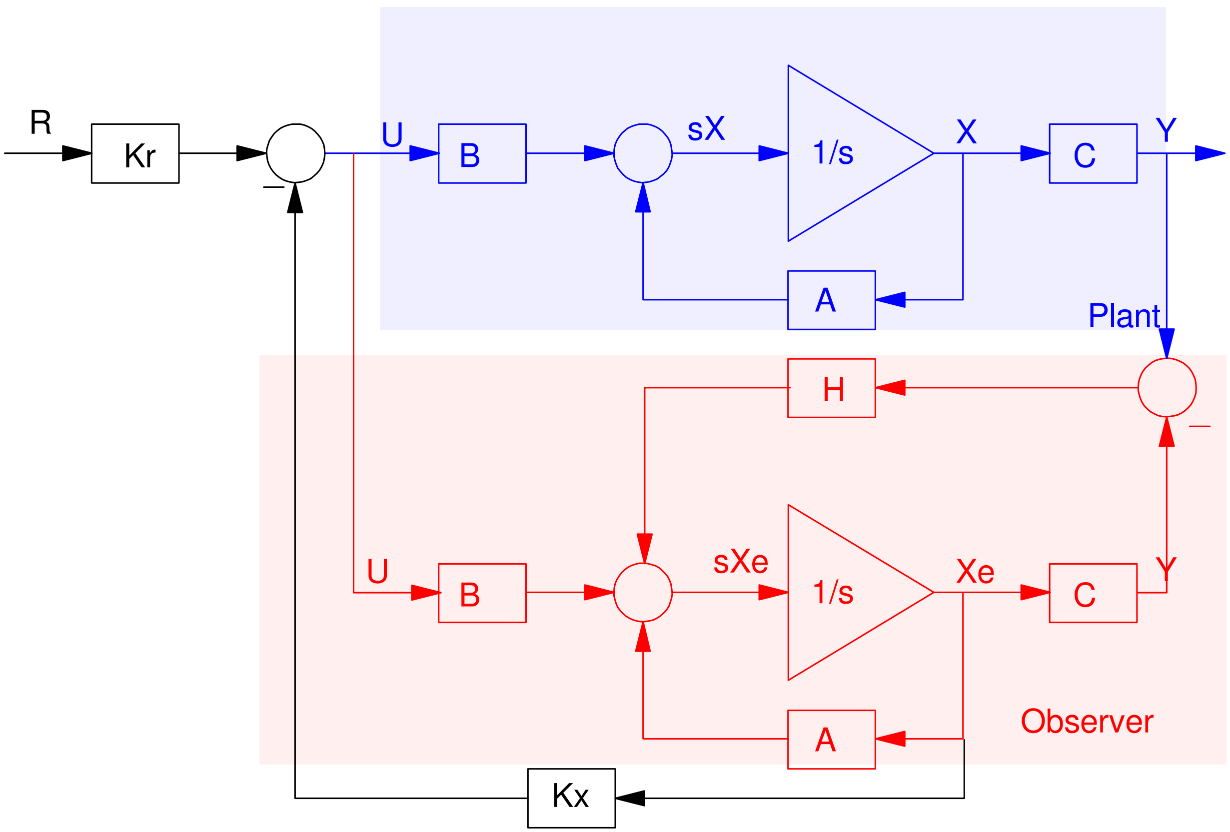
If all of the states are measured, you can use full-state feedback to control a dynamic system.

- Bass-Gura (pole placement) can be used to find the feedback gains
- LQR techniques can be used to find the feedback gains



If the states are not measured, they can be estimated with a full-order observer

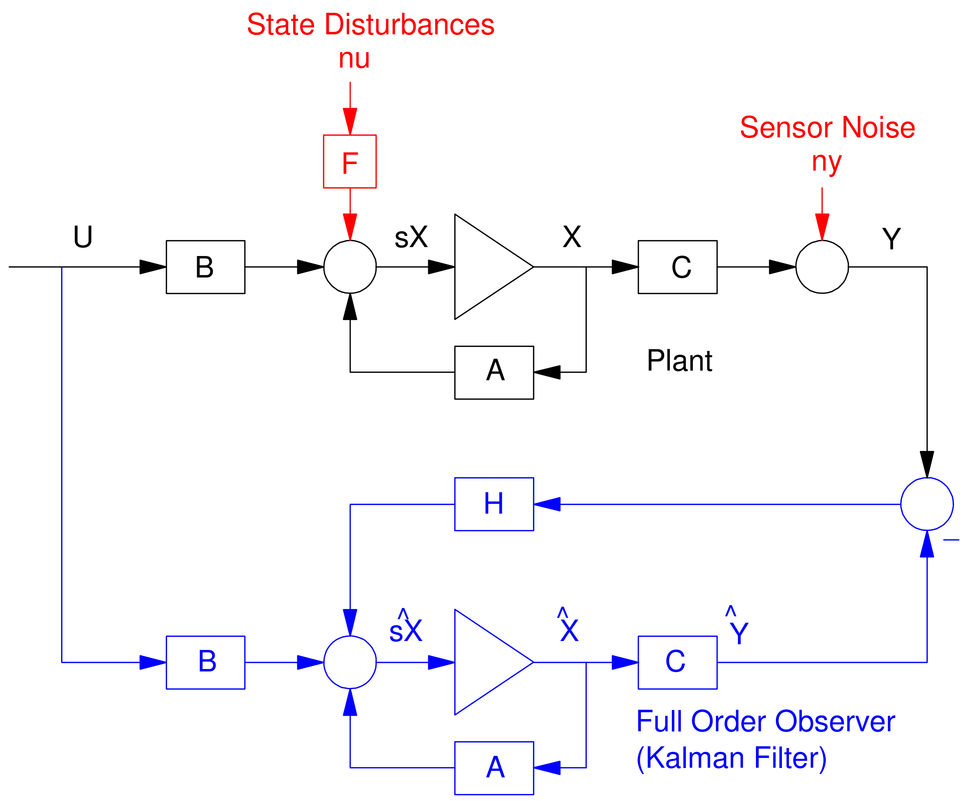
- Bass-Gura can be used to find H
- LQR techniques can be used to find H



# What if there is noise?

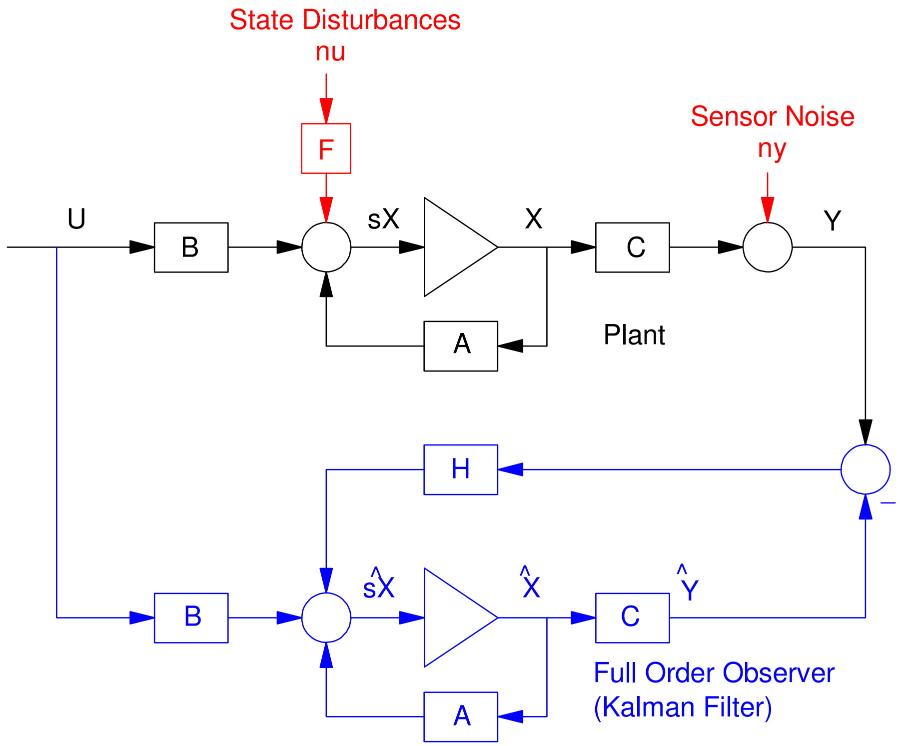
$$sX = AX + BU + Fv \qquad v = \eta(0, V^2)$$

$$Y = CX + w \qquad w = \eta(0, W^2)$$



H should reflect the amount of noise on the system:

- If the sensor noise is zero, H should be is large
- If the state-disturbances are zero, H should be small
- What is the "best" H if you have both?



## Kalman Filter

If "best" is minimizing the variance of the state error

$$E\left((X - X_e)^2\right)$$

the solution is a LQR observer where

$$Q = (FV)(FV)^T = F \cdot V^2 \cdot F^T$$
$$R = WW^T = W^2$$

An LQR observer designed with this value of Q and R is termed a *Kalman Filter*. (It's just a full-order observer with a specific feedback gain, H.)

## Example: 4th Order Heat Equation

Suppose you have a 4-order heat equation with a large disturbance at the input.

$$Fv = Bv = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} v \qquad\qquad v \sim N(0, 1^2)$$

Determine the optimal observer with sensor noise:

$w \sim N(0, W^2)$ sensor noise

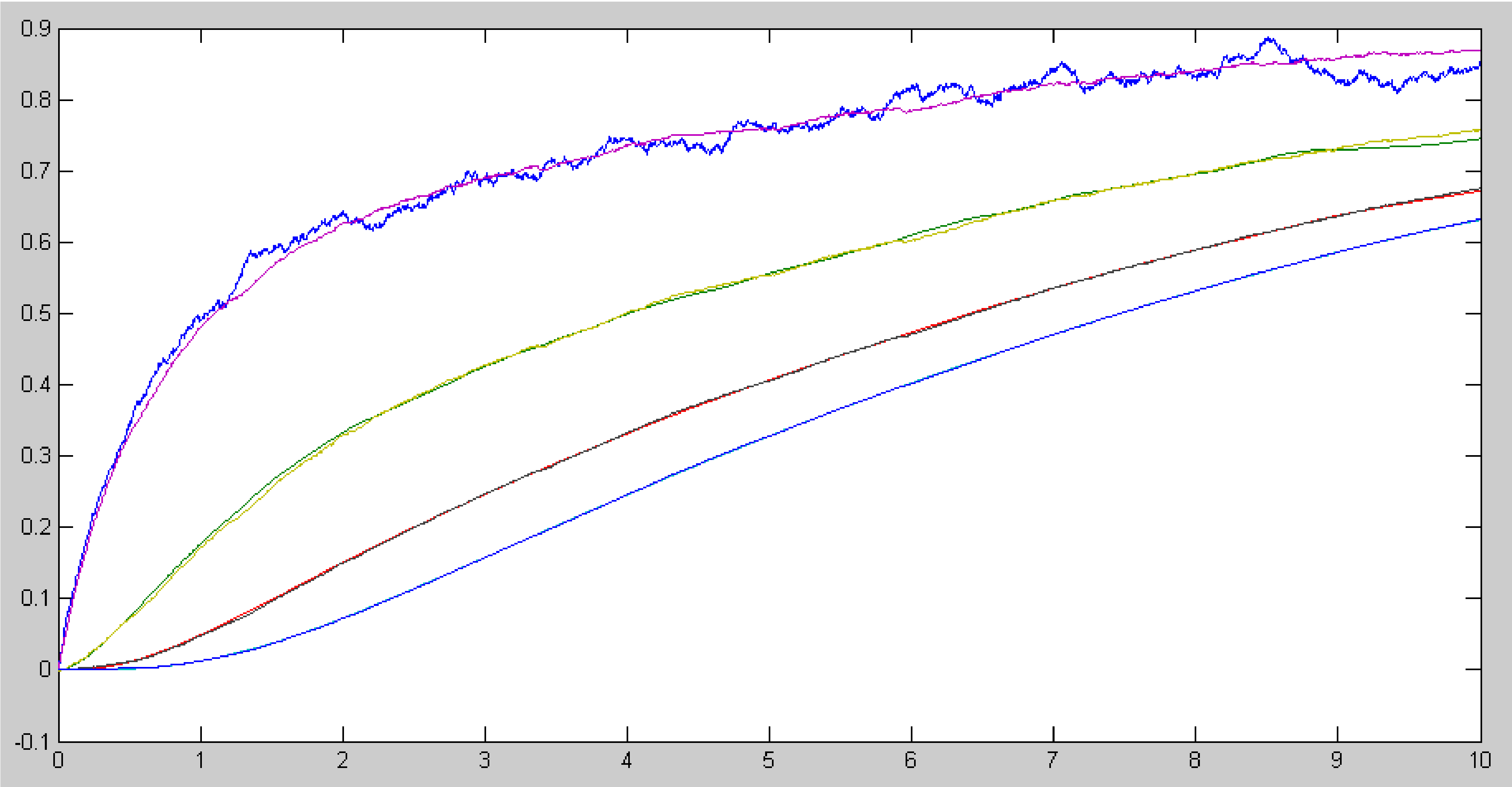
- $W = 0.01$ Good sensors: noise 100x less than disturabances
- $W = 0.1$
- $W = 1$ Noisy Sensors

# Case 1: Good Sensors (small noise)

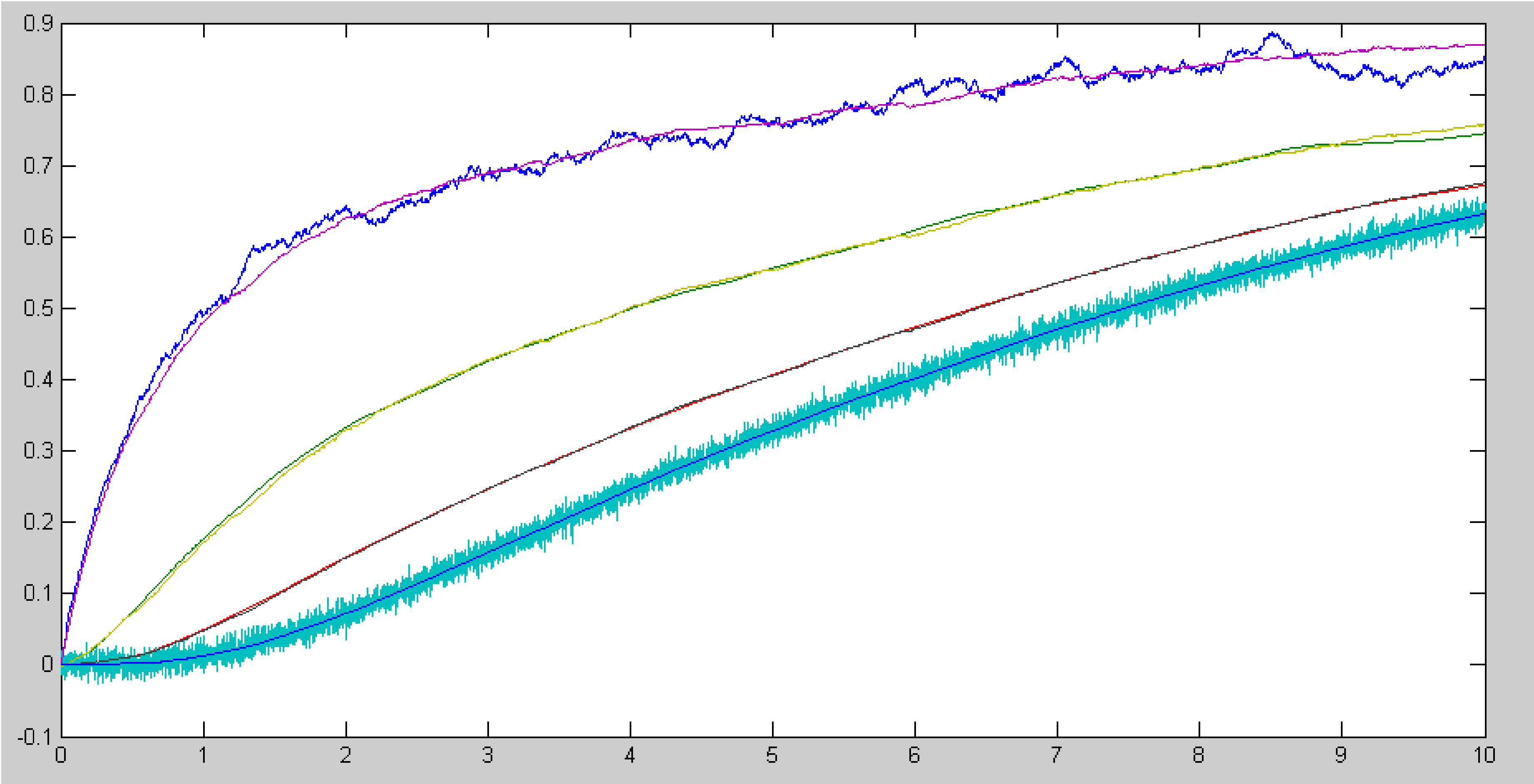
- W = 0.01
- V = 1

```
W = 0.01;       % sensor noise
V = 1;          % state disturbance (on U)
Q = F*V*V*F';
R = W*W;
H = lqr(A',  C',  Q,  R)'
A8 = [A, zeros(4,4) ;   H*C,  A - H*C ];
B8 = [B  F  zeros(4,1) ;  B   zeros(4,1)   H];

     Ref         Nx          Ny
    1.0000    1.0000          0
         0         0          0
         0         0          0
         0         0          0
    1.0000         0    25.5559
         0         0    18.4321   H shows up here
         0         0     8.0233
         0         0     3.1287
```




States and State Estimates with State and Sensor Noise: $W \sim N(0, 0.01^2)$, $V \sim N(0, 1^2)$




States and State Estimates.

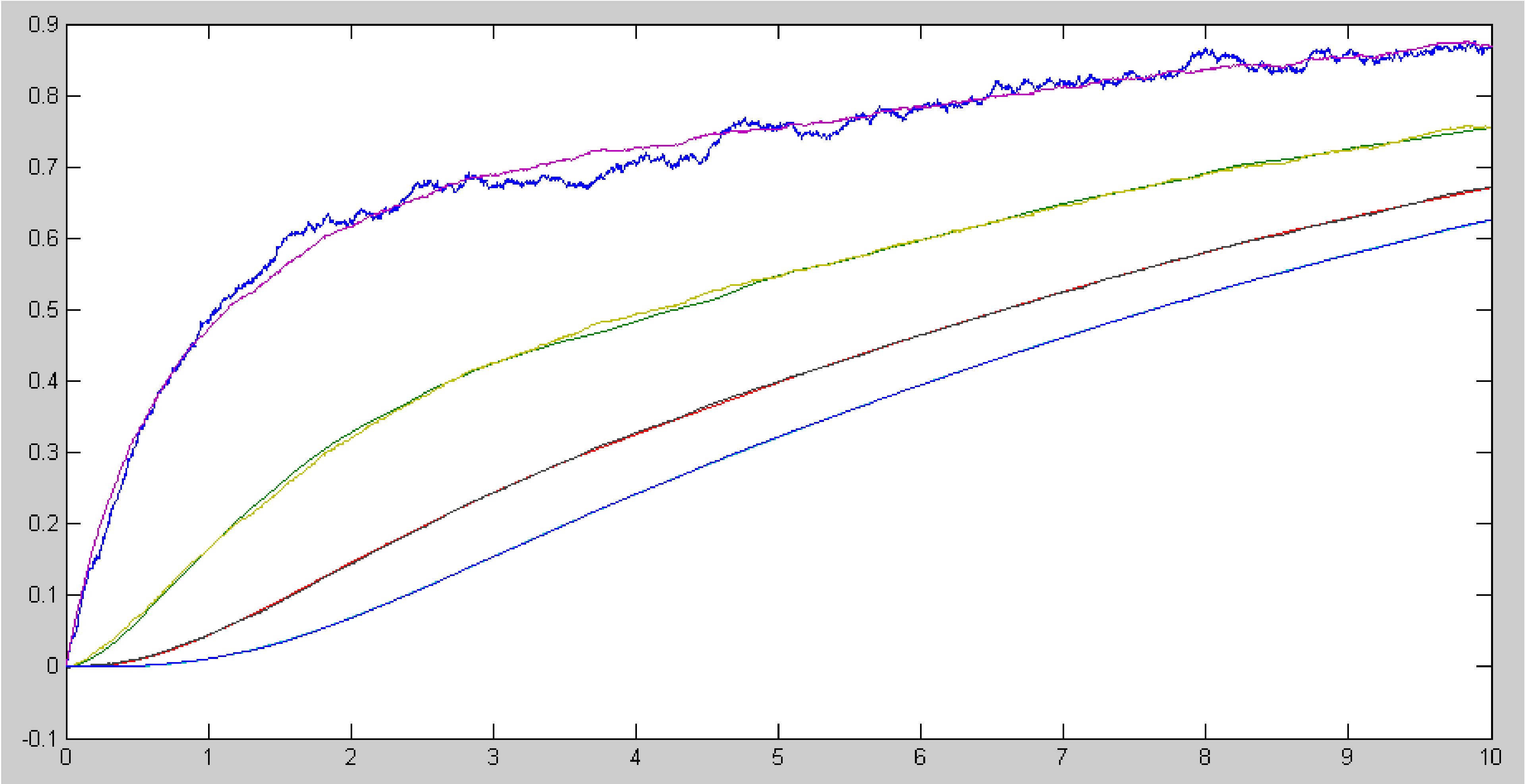
## Case 2: Not so Good Sensors (small noise)

- W = 0.1
- V = 1

```
W = 0.01;        % sensor noise
V = 1;           % state disturbance (on U)
Q = F*V*V*F';
R = W*W;
H = lqr(A',  C',  Q,  R)'

    1.1169
    1.2204
    0.9184
    0.6843
```

Note that the larger sensor noise has resulted in the observer gains being reduced.



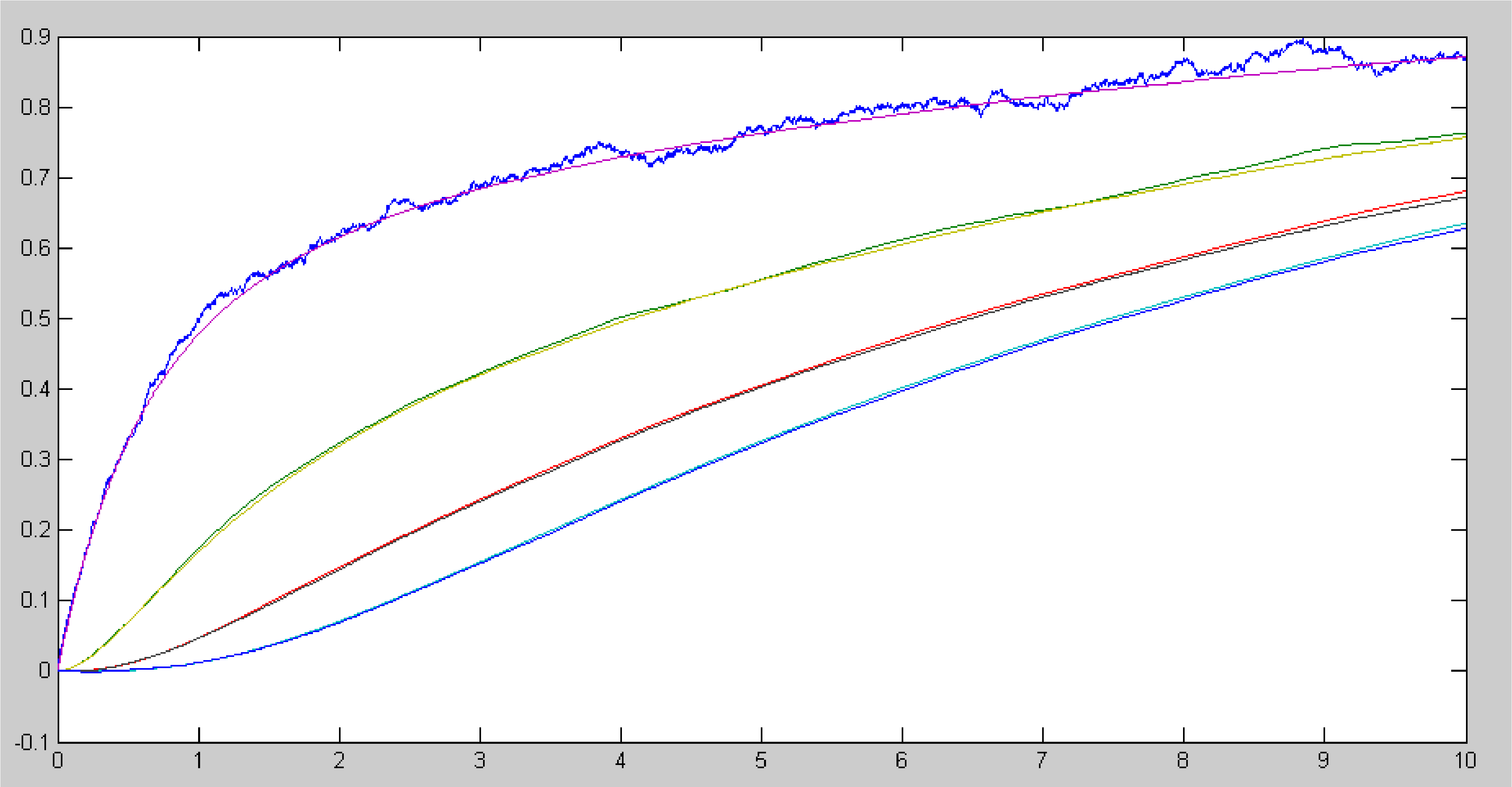

States and State Estimates with State and Sensor Noise:  Sensor Noise 10x larger

# Case 3: Noisy Sensors

- W = 1
- V = 1

```
W = 1;        % sensor noise
V = 1;           % state disturbance (on U)
Q = F*V*V*F';
R = W*W;
H = lqr(A',  C',  Q,  R)'

    0.0293
    0.0417
    0.0436
    0.0427
```



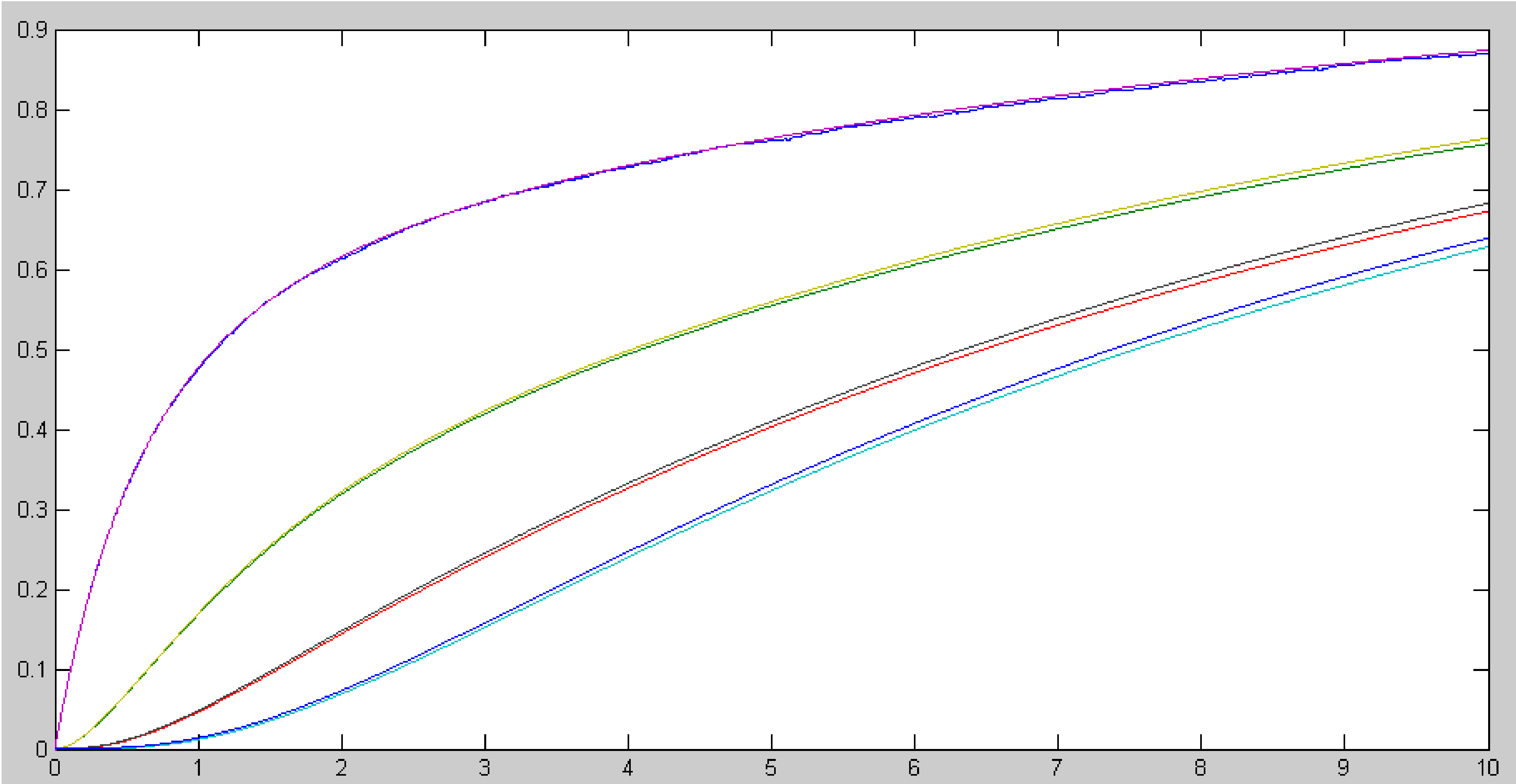

States and State Estimates with State and Sensor Noise:  Sensor Noise 100x larger

Note: Only the ratio of Q and R matter. If both disturnances are 10x smaller, you get the same observer gains (same Kalman filter gains)

```
W = 0.1;          % sensor noise
V = 0.1;          % state disturbance (on U)
Q = F*V*V*F';
R = W*W;
H = lqr(A',  C',  Q,  R)'

H =

    0.0293
    0.0417
    0.0436
    0.0427
```




States and State Estimates with State and Sensor Noise: Both sensor and input noise scaled down 10x

# Main Calling Script in Matlab

```
W = 0.01;        % sensor noise
V = 1;           % state disturbance (on U)
Q = F*V*V*F';
R = W*W;
H = lqr(A',  C',  Q,  R)'
A8 = [A, zeros(4,4) ;   H*C,  A - H*C ];
B8 = [B  F  zeros(4,1) ;  B   zeros(4,1)   H];

t = [0:0.001:10]';
N = size(t);
X0 = zeros(8,1);
U = [ ones(N),  randn(N)*V,  randn(N)*W ];
C8 = eye(8,8);
D8 = zeros(8,3);
y = step3(A8, B8, C8, D8, t, X0, U );
plot(t,y)
```

```
function [ y ] = step3( A, B, C, D, t, X0, U )

T = t(2) - t(1);
[m, n] = size(C);

npt = length(t);

Az = expm(A*T);
Bz = B*T;

X = X0;

y = zeros(npt, m);

y(1,:) = (C*X + D * ( U(1,:)' ) )';

for i=2:npt
    X = Az*X + Bz*( U(i,:)' );
    Y = C*X + D * ( U(i,:)' );
    y(i,:) = Y';
   end
end
```