Lecture 14

Kalman filtering (Optimal State Estimation)

- **Review of LQR (Optimal State Feedback)**
  - **Q and R : had quantified penalties on states (Q) and u (R)**
- **Kalman filtering**
  - **Q and R : now quantify process (Q) vs sensor (R) noise**

Recall the framework for LQR…

LQR = Linear Quadratic Regulator

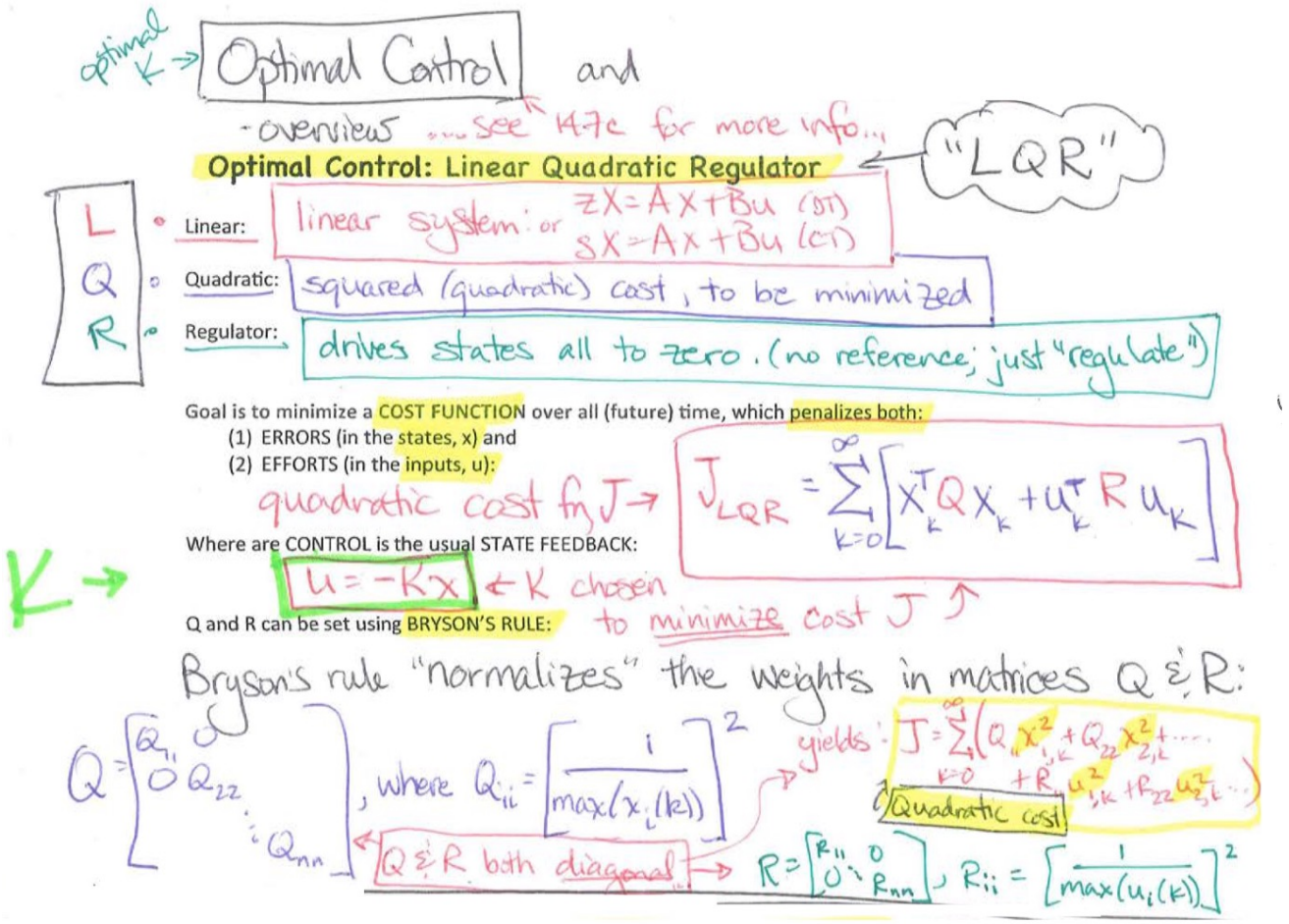
Recall that:

- **Q scales penalties on state(s), in x**
- **R scales penalties on input(s), in u**

Typically, both Q and R are diagonal matrices.

We can use Bryson's rule to set terms in Q and R, approximately.

(In good control design, small tweaks to Q and R often do not often yield big changes in closed-loop dynamics…)



IMPORTANT :

Q and R are also variables that are traditionally used in Kalman filtering…

But Q and R will now, instead, be:

Covariance Matrices

Recall the Control and Estimation Problems:

- **Control problem:** Design gain matrix K.

  $x_{k+1} = A_d x_k + B_d u_k$ with $u_k = -K(x_k - x_{ref,k})$

  - Closed-loop poles are eig(A-B*K)
  - **Trade-off?** Large K = fast response, but large actuation

- **Estimation problem:** Design gain matrix L.

  $\hat{x}_{k+1} = A_d \hat{x}_k + B_d u_k + L(y_k - C\hat{x}_k)$ where $y_k = Cx_k$

  - Closed-loop poles are eig(A-L*C)
  - **Trade-off:** Large L = fast error decay, but noise sensitivity

Recall the Separation Principle :

When x is an $n \times 1$ vector (i.e., **with *n* states**):

- The *n* poles from the **Control Problem** will be:
  eig(A-BK)

- The *n* poles from the **Estimation Problem** will be:
  eig(A-L_pC)

The "error dynamics" have ONLY: eig(A-L_pC).

The "state dynamics" have **a total of 2*n* poles**, i.e. BOTH:
eig(A-BK) and eig(A-L_pC)

Note "error" is: $e_x = x - \hat{x}$, so $\hat{x} = x - e_x$.
If $u = -K\hat{x}$, then u is "driven by" the error signal:

$$u = -Kx + Ke_x$$

Optimal Estimation : minimize expected error!

Optimal Estimation: Linear Quadratic Estimation

This optimal estimator is also called a "Kalman filter".

Q & R variables have a different meaning (than for LQR)...

Goal is to minimize VARIANCE in the estimated states.

We now assume there is both:

(1) PROCESS noise, in the state dynamics: $x(k+1) = A_d x(k) + B_d u(k) + G w(k)$

(2) MEASUREMENT noise, affecting y(k): $y(k) = C_d x(k) + D_d u(k) + v(k)$

Cov (Q) → process noise

measurement noise → Covariance R

We will assume both w(k) and v(k) are Gaussian, white noise sources.

a) Gaussian: Just need to know mean and covariance to describe noise. A Gaussian input to a lin. sys. yield a Gaussian output.

b) white noise: no time correlation

Optimal solution for estimator gains, L, depends of total current variance of the state estimates, before a measurement (P^-), compared with measurement variance (R):

Optimal gain matrix, L, divides "trust" between noisy model estimate, $\hat{y}$, & noisy measurement, y.

L →

$$\hat{x}_{k+1} = A\hat{x}_k + Bu_k + L(y - \hat{y})$$

Same $L=L_p$ is used, but now we pick L_p to minimize the variance in our estimate.

What is a “covariance matrix”?

From Wikipedia…

> In probability theory and statistics, a **covariance matrix** (also known as **auto-covariance matrix**, **dispersion matrix**, **variance matrix**, or **variance–covariance matrix**) is a square matrix giving the covariance between each pair of elements of a given random vector.
>
> Intuitively, the covariance matrix generalizes the notion of variance to multiple dimensions. As an example, the variation in a collection of random points in two-dimensional space cannot be characterized fully by a single number, nor would the variances in the x and y directions contain all of the necessary information; a 2×2 matrix would be necessary to fully characterize the two-dimensional variation.
>
> Any covariance matrix is symmetric and positive semi-definite and its main diagonal contains variances (i.e., the covariance of each element with itself).

How do convariance matrices enter control?

Instead of only u, real-world systems are also driven by PROCESS NOISE.

- Q characterizes this process noise.

Instead of only y=Cx+Du, real measurements tend to have SENSOR NOISE.

- R characterizes this sensor noise.

From Wikipedia… *(look up "covariance matrix")*

Definition [edit]

Throughout this article, boldfaced unsubscripted $\mathbf{X}$ and $\mathbf{Y}$ are used to refer to random vectors, and Roman subscripted X_i and Y_i are used to refer to scalar random variables.

If the entries in the column vector

$$\mathbf{X} = (X_1, X_2, \ldots, X_n)^{\mathrm{T}}$$

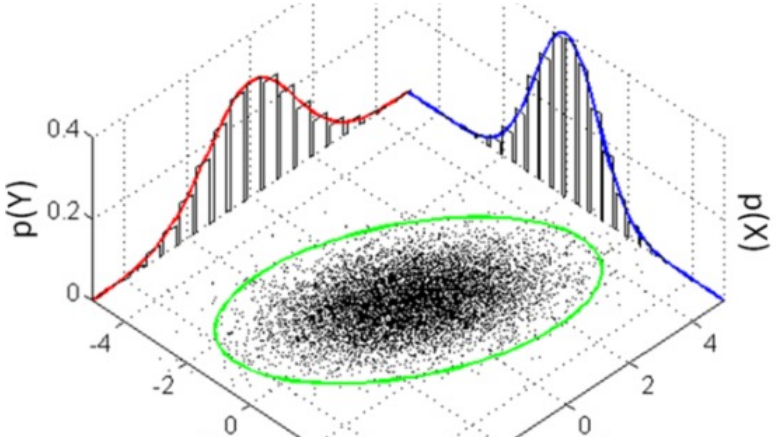
are random variables, each with finite variance and expected value, then the covariance matrix $\mathrm{K}_{\mathbf{XX}}$ is the matrix whose (i, j) entry is the covariance[1]:177

$$\mathrm{K}_{X_i X_j} = \operatorname{cov}[X_i, X_j] = \mathrm{E}[(X_i - \mathrm{E}[X_i])(X_j - \mathrm{E}[X_j])]$$

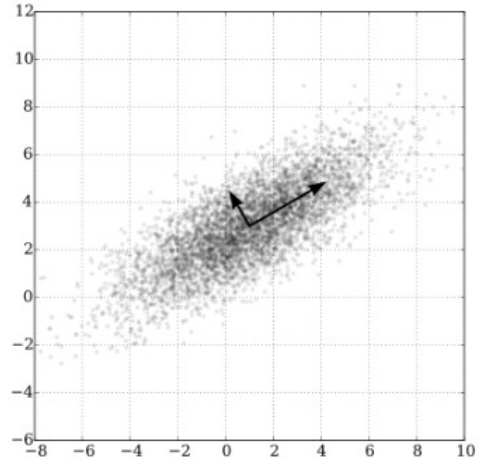
where the operator E denotes the expected value (mean) of its argument.

What is a “covariance matrix”?

From Wikipedia…



In probability theory and statistics, the **multivariate normal distribution**, **multivariate Gaussian distribution**, or **joint normal distribution** is a generalization of the one-dimensional (univariate) normal distribution to higher dimensions. One definition is that a random vector is said to be *k*-variate normally distributed if every linear combination of its *k* components has a univariate normal distribution. Its importance derives mainly from the multivariate central limit theorem. The multivariate normal distribution is often used to describe, at least approximately, any set of (possibly) correlated real-valued random variables, each of which clusters around a mean value.



Sample points from a bivariate Gaussian distribution with a standard deviation of 3 in roughly the lower left–upper right direction and of 1 in the orthogonal direction. Because the *x* and *y* components co-vary, the variances of x and y do not fully describe the distribution. A 2×2 covariance matrix is needed; the directions of the arrows correspond to the eigenvectors of this covariance matrix and their lengths to the square roots of the eigenvalues.

Toy Example

More on covariance matrices Q and R...

Both the optimal Control and optimal Estimation assume noise is Gaussian and white.

For Both problems:

Q (nxn) is a covariance matrix for STATES, x

R ($\ell x \ell$) is a covariance matrix for OUTPUTS, y

Q and R must therefore be symmetric, positive semi-definite matrices.

(You can't have a "negative" standard deviation or variance for a Gaussian!)

st. dev. of "noise"

$\sigma_1 = 2$

$\sigma_2 = 1$

MATLAB examples.

```
mu1 = 2; mu2 = 1; N=1e6;
x1 = normrnd(0,mu1,1,N);
x2 = normrnd(0,mu2,1,N);
figure(1); clf; subplot(231)
plot(x1,x2,'b.'); axis equal

A = 30*pi/180;
x1a = cos(A)*x1 - sin(A)*x2;
x2a = sin(A)*x1 + cos(A)*x2;
subplot(232);
plot(x1a,x2a,'b.'); axis equal

x1b = 4*x2;
x2b = 3*x2;
subplot(233); plot(x1b,x2b,'b.'); axis equal
```

random (noisy) variables.

New variables are a linear combination of x_1 & x_2.

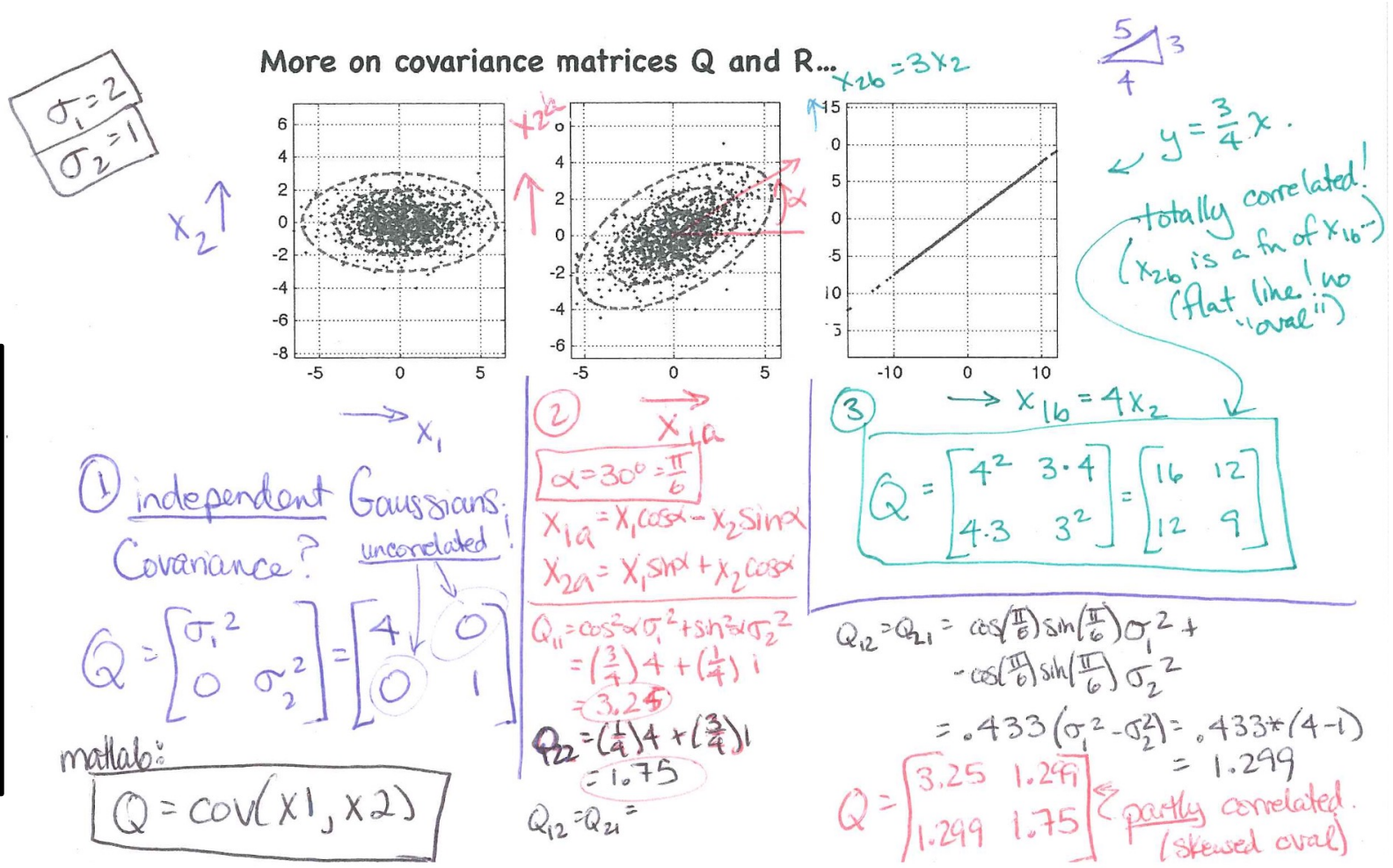
(see next slide...)

(Bad choice of variable name! ...but here, "mu" is just a variable name. "Sigma" is the 2nd arg of normrnd.)

Terms NOT on the diagonal of the covariance matrix are what cause the "skewed" distributions (middle and at right), versus the non-skewed ellipsoid (at left).

Only case (1) has a diagonal Q matrix.

The off-diagonal terms in **(2)** and **(3)** cause that "skewed" (i.e., slanted) shape.



Toy Example with:

x_1 = scalar guess of what x(k) is, e.g., a sensor measurement,

$$x_1(k) = y(k) = x(k) + v(k) \quad , \quad \text{where } C_d=1 \text{ is assumed.}$$

x_2 = (uncorrelated) second guess of what x(k) is, e.g., a model-based guess,

$$x_2(k) = A_d\hat{x}(k-1) + B_d(u(k-1) + w(k-1)).$$

Here, imagine just one state in x, and that y = x + v, where v is sensor noise.

- Sensor noise is variable v(k), at time step k.

At time step k, we can ALSO guess at what x is based on our last estimate.

- Process noise is variable w(k), at time step k.

We seek a "blended estimate" here.

If neither x_1 nor x_2 is a "biased" guess, then we want to average them via:

$$x_{tot} = x_3 = ax_1 + (1\text{-}a)x_2$$

i.e., a + (1-a) = 1, so the weights (a times x_1, and (1-a) times x_2) add to 1.

The **optimal solution** for weight "a" **minimizes the expected error**.

Toy Example with:

x_1 = scalar guess of what x(k) is, e.g., a sensor measurement,

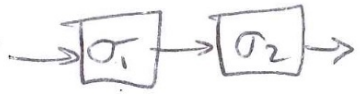
$$x_1(k) = y(k) = x(k) + v(k) \quad , \quad \text{where } C_d=1 \text{ is assumed.}$$

x_2 = (uncorrelated) second guess of what x(k) is, e.g., a model-based guess,

$$x_2(k) = A_d\hat{x}(k-1) + B_d(u(k-1) + w(k-1)).$$

Kalman Filter Toy Example [Optimal Estimation]

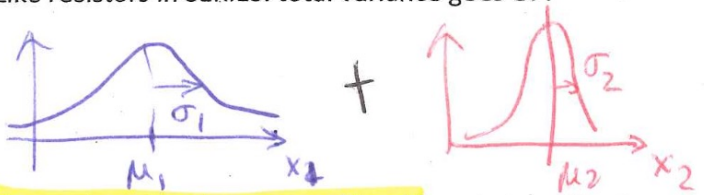
The elegance of the Kalman filter depends on the assumption of GAUSSIAN, WHITE NOISE.



There are two way to combine Gaussians...

(1) Two noisy variables get ADDED to one another.
Like resistors in SERIES: total variance goes UP.

"series"



$$x_3 = x_1 + x_2$$

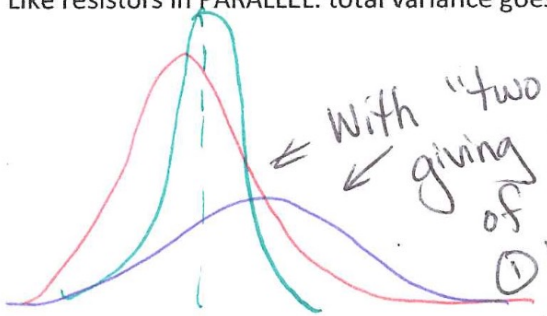
$$\sigma_3^2 = \sigma_1^2 + \sigma_2^2$$

↑ total noise goes up.

$$\mu_3 = \mu_1 + \mu_2$$

(1) Two noisy estimates are combined for a total estimate.
Like resistors in PARALLEL: total variance goes DOWN.

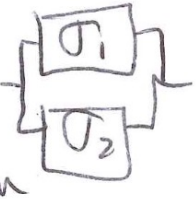
"parallel"



⇐ With "two sets of eyes" (in parallel) giving independent guesses of $\hat{x}$ / model vs measurement),

① best estimate is weighted toward "less noisy" guess

② estimate has smaller σ than either guess alone:



$$\frac{1}{\sigma_3^2} = \frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}$$

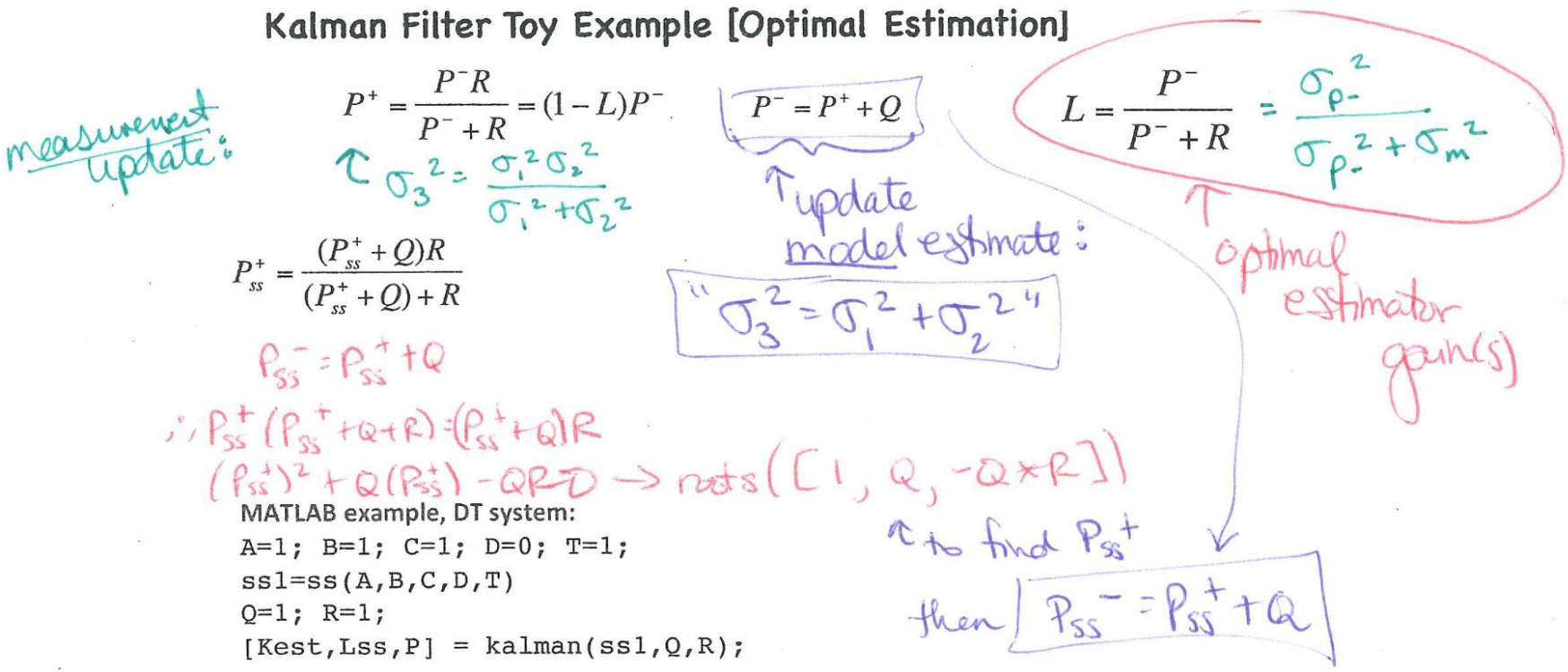
$$\sigma_3^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 + \sigma_2^2}$$

$$\mu_3 = \left(\frac{\sigma_2^2}{\sigma_1^2+\sigma_2^2}\right)\mu_1 + \left(\frac{\sigma_1^2}{\sigma_1^2+\sigma_2^2}\right)\mu_2$$

Toy Example, continued…

Generalizing:

- **R is the covariance matrix** (here, just sigma squared) for sensor noise, v
- Q is the covariance matrix (here, also a scalar) for process noise, w
- **P^- and P^+ are keeping track of MODEL-BASED covariance in our estimate over time**, before (-) **versus after (+) each new sensor measurement** is "blended in".



See Matlab example, below. Compare Lp1 and Lp2, and the resulting estimator poles…

```
% Kalman_example_code.m
%
% Example using the "pendulum up" case.

Xref = zeros(4,1); % theta = 0 is "pendulum up"
u0 = 0;            % u0 is required u at equilibrium (just zero here)
T = 0.005;         % sample time (seconds)

[Ad,Bd,A,B] = Get_Matrices(Xref,u0,T)
Cd = [1 0 0 0; 0 0 1 0]; Dd = [0;0];
dtss = ss(Ad,Bd,Cd,Dd,T);  % create a "DT state space system" (with T)

% Only the RELATIVE values in Q vs R matter.
% - Here, we will focus ONLY on the effects of change R: sensor noise
Q = 1; % Process Noise matrix: noise added to u

% Recall, xhat(n+1) = Ad*xhat(n) + Bd*u(n) + Lp*(y(n) - yhat(n)),
% So:
%     Larger values in Lp (in general) mean:
%        + "more trust" of measurements, in y
%        - "less trust" of model-based estimate, in yhat



R1 = 1e4;              % Sensor Noise matrix (here, a scalar)
[~,Lp1,~] = kalman(dtss,Q,R1)
est_poles_1 = eig(Ad-Lp1*Cd)
splane_1 = (1/T)*log(est_poles_1)


R2 = 1;              % This is MUCH SMALLER sensor noise...
[~,Lp2,~] = kalman(dtss,Q,R2)  % So gains in Lp2 will be LARGER
est_poles_2 = eig(Ad-Lp2*Cd)
splane_2 = (1/T)*log(est_poles_2)
```

```
Lp1 =
    0.0001   -0.0015
    0.0002   -0.0077
   -0.0015    0.0510
   -0.0077    0.2599
est_poles_1 =
    1.0000
    0.9748
    0.9247
    0.9774
splane_1 =
   -0.0022
   -5.1002
  -15.6672
   -4.5701
Lp2 =
    0.0011   -0.0014
    0.0003   -0.0078
   -0.0014    0.0510
   -0.0071    0.2604
est_poles_2 =
    0.9247
    0.9989
    0.9742
    0.9780
splane_2 =
  -15.6547
   -0.2238
   -5.2306
   -4.4588
>>
```