**Control Systems 414** **April 2007**

# Discrete State Space Control


## Mr I.K. Peddle


---

**Introduction to Direct Digital Design Using State Space Methods**

In the previous set of notes we learnt how to design controllers using 'classical methods'. In this set of notes we consider 'modern methods' of control system design i.e. state space. With state space design, we remain in the time domain and thus work directly with the differential equation model of our plant. It is important to realise that whether we work with transfer functions or with differential equations in state space form, the mathematics describes the same thing and the forms can be interchanged. The major advantage however of working with a state space model of a system is that the internal system state is explicitly maintained over time, where as with a transfer function, only the input output relationship is maintained.

In these notes we consider only direct digital design and do not consider emulation in state space systems. However, emulation techniques could be equally well applied to state space derived controllers.

For a direct digital design, it is necessary for us to convert our continuous time plant to a discrete equivalent that is capable of predicting the output of the plant at the sample instances, given that the control signal to the plant is updated every sample instance and held using a ZOH circuit in-between samples. In classical control we developed the formula for G(z) as a function of G(s). However, we now have a state space model of our plant and would like to derive a similar formula to convert this model to a discrete equivalent state space model. To do this, consider the continuous state space system,

$$\dot{\underline{x}} = F\underline{x} + Gu$$
$$y = H\underline{x} + Ju$$

The system equation describes a non-homogenous (there is a driving input) set of coupled linear differential equations with a state solution (for a derivation of this solution follow the mathematics in section 5.7 of Gopal – not required though),

$$\underline{x}(t) = e^{F(t-t_0)}\underline{x}(t_0) + \int_{t_0}^{t} e^{F(t-\tau)}Gu(\tau)d\tau$$

The above equation allows you to calculate the state vector at any time $t$ given the state vector at the starting time $t_0$ and the control input signal between $t_0$ and $t$. Note that the first term in the above equation is the homogenous solution and the second term is the particular solution (convolution of the input with the system's impulse response). Now, if we are looking for the discrete equivalent of the continuous state space model then we are interested in finding the state vector at the 'next sample instance' given the state vector at the 'current sample instance'. So, let's make $t_0 = kT$ (current sample instance) and look at the state vector T seconds later (next sample instance),

$$\underline{x}(kT+T) = e^{FT}\underline{x}(kT) + \int_{kT}^{kT+T} e^{F(kT+T-\tau)}Gu(\tau)d\tau$$

Since a ZOH circuit holds the control constant over the entire sample period, we can move the control input $u(\tau)$ and the input matrix G out of the integration (integration is only over one sample period). Then, after changing integration variables and simplifying we can write,

$$\underline{x}(k+1) = e^{FT}\underline{x}(k) + \int_0^T e^{F\eta}d\eta \cdot Gu(k)$$
$$= \Phi\underline{x}(k) + \Gamma u(k)$$

where,

$$\Phi = e^{FT} = I + FT + (FT)^2/2! + (FT)^3/3! + \ldots$$
$$\Gamma = \int_0^T e^{F\eta}d\eta \cdot G$$

If we write,

$$\Phi = I + FT\Psi$$

where,

$$\Psi = I + FT/2! + (FT)^2/3! + \ldots$$

then,

$$\Gamma = \int_0^T e^{F\eta}d\eta \cdot G = F^{-1}e^{F\eta}\Big|_0^T G = F^{-1}(\Phi - I)G = T\Psi G$$

So the discrete equivalent state space model becomes,

$$\underline{x}(k+1) = \Phi\underline{x}(k) + \Gamma u(k)$$
$$y(k) = H\underline{x}(k)$$

where the direct feed through of the control into the output equations has been omitted (J = 0). MATLAB's c2d.m function performs the above conversion,

```
>> sysC = ss(A,B,C,D);
>> sysD = c2d(sysC,T,'zoh');
```

The discrete equivalent state space model is obviously exactly the same as the discrete equivalent transfer function model G(z) that you are used to calculating. To show the relationship between the state space matrices and G(z), take the Z-transform of the state space equations,

$$zI\underline{X}(z) = \Phi\underline{X}(z) + \Gamma U(z)$$
$$Y(z) = H\underline{X}(z)$$

=> $\underline{X}(z) = (zI - \Phi)^{-1}\Gamma U(z)$

=> $Y(z) = H(zI - \Phi)^{-1}\Gamma U(z)$

=> $$G(z) = \frac{Y(z)}{U(z)} = H(zI - \Phi)^{-1}\Gamma = \frac{H \cdot adj(zI - \Phi)\cdot\Gamma}{\det(zI - \Phi)}$$

where the *adjoint* operator is defined on page 319 of Gopal. Thus, equations to calculate the poles and the zeros of a state space system are,

Zeros at: $H \cdot adj(zI - \Phi)\cdot\Gamma = 0$

Poles at: $\det(zI - \Phi) = 0$

The pole equation is the same equation used to calculate the eigenvalues of the system matrix $\Phi$. To calculate the poles and zeros using MATLAB the commands are,

```
>> poles = eig(Phi);
>> zeros = tzero(sysD);
```

**Two canonical form of interest for later on**

We will often have a continuous plant G(s) that we convert to a discrete equivalent G(z) (by hand say) and then want to convert G(z) to a particular state space form. Remember the choice of state vector for a particular system is arbitrary, as long as the vector can fully describe the state of a system. This means of course that there are an infinite number of state space representations that correspond to the same transfer function. Two special state space forms called the Control Canonical and Observer Canonical forms are of particular interest to us when it comes to reducing the calculation complexity when designing controllers and estimators. This is particularly useful if the controller or estimator needs to be designed by hand (as in during an exam!).

The control and observer canonical forms for a general third order transfer function are listed below. It should be clear how to adapt the results below for higher or lower order transfer function. Given,

$$G(z) = \frac{Y(z)}{U(z)} = \frac{b_0 z^3 + b_1 z^2 + b_2 z + b_3}{z^3 + a_1 z^2 + a_2 z + a_3}$$

we have,

*Control Canonical Form*

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \end{bmatrix} = \begin{bmatrix} -a_1 & -a_2 & -a_3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} u(k)$$

$$y(k) = \begin{bmatrix} b_1 - a_1 b_0 & b_2 - a_2 b_0 & b_3 - a_3 b_0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} + \begin{bmatrix} b_0 \end{bmatrix} u(k)$$

*Observer Canonical Form*

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \\ x_3(k+1) \end{bmatrix} = \begin{bmatrix} -a_1 & 1 & 0 \\ -a_2 & 0 & 1 \\ -a_3 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} + \begin{bmatrix} b_1 - a_1 b_0 \\ b_2 - a_2 b_0 \\ b_3 - a_3 b_0 \end{bmatrix} u(k)$$

$$y(k) = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_1(k) \\ x_2(k) \\ x_3(k) \end{bmatrix} + \begin{bmatrix} b_0 \end{bmatrix} u(k)$$

In this course you will only be expected to write a transfer function in one of these two forms as well as recognise these forms to simplify the hand calculations of either your control law or estimator design (we will see the simplifications that result later). Note how the observer canonical form matrices are related to the control canonical matrices,

$$\Phi_o = \Phi_c^T \qquad \Gamma_o = H_c^T \qquad H_o = \Gamma_c^T \qquad J_o = J_c$$

where, subscript ‘o’ represents observer form and ‘c’ represents control form. Finally, Gopal discusses Control Canonical and Observer Canonical forms on pages 433 to 435. However, they are referred to as First and Second Companion forms respectively and are derived with the states packed into the state vector in reverse order (the convention changes from book to book – the convention presented in these notes is more common however).

**Control Law Design**

[1]Consider the state space system,

$$\underline{x}(k+1) = \Phi \underline{x}(k) + \Gamma u(k)$$
$$y(k) = H \underline{x}(k)$$

The open loop poles are at,

$$\det(zI - \Phi) = 0$$

We would like to design a controller such that the closed loop poles are at certain desired locations. Define the desired pole locations with the characteristic equation,

$$\alpha_c(z) = z^n + \alpha_1 z^{n-1} + \ldots + \alpha_{n-1} z + \alpha_n = 0$$

Our strategy is to use full state feedback,

$$u(k) = -K \underline{x}(k)$$

For now we assume that the full state vector is available for feedback . The closed loop system becomes,

$$\underline{x}(k+1) = \Phi \underline{x}(k) - \Gamma K \underline{x}(k) = (\Phi - \Gamma K)\underline{x}(k)$$
$$y(k) = H \underline{x}(k)$$

with poles at,

$$\det(zI - (\Phi - \Gamma K)) = 0$$

So, let

$$\det(zI - (\Phi - \Gamma K)) = \alpha_c(z)$$

and solve for K. Note K has as many elements (degrees of freedom) as there are poles. This means that we can place the closed loop poles anywhere as long as the system is controllable from the input u(k). To test for controllability, we need to ensure that the controllability matrix (see Section 6.6 of Gopal for details – not required though),

$$C = \begin{bmatrix} \Gamma & \Phi\Gamma & \cdots & \Phi^{n-1}\Gamma \end{bmatrix}$$

where n is the order of the system, is of full rank. For SISO systems this is equivalent to the test,

$$\det(C) \neq 0$$

*Example: Full state feedback*

$$\underline{x}(k+1) = \begin{bmatrix} 1.5 & -0.5 \\ 1 & 0 \end{bmatrix} \underline{x}(k) + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u(k) \qquad T = 0.2s$$

$$y(k) = \begin{bmatrix} 0 & 1 \end{bmatrix} \underline{x}(k)$$

[1] Note that the rest of what is described in these notes is discussed in section 7.9 of Gopal. Chapter 7 of Gopal looks at state space design for continuous time systems and section 7.9 simply applies these results to discrete time systems (since they are basically the same).

*For the state space system above, design a full state feedback controller such that the closed loop system has equivalent s-plane poles at,*

$$s_{CL} = -\sigma \pm j\omega = -2 \pm j2$$

*Also draw a block diagram of the closed loop system.*

*Now,*

$$z_{CL} = e^{s_{CL}T} = e^{-\sigma T}[\cos(\omega T) \pm j\sin(\omega T)] = 0.62 \pm j0.26$$

=> $\alpha_c(z) = (z-0.62)^2 + (0.26)^2 = z^2 - 1.24z + 0.45$

*So,*

$$\det(zI - (\Phi - \Gamma K)) = \alpha_c(z)$$

$$\det\left(\begin{bmatrix} z & 0 \\ 0 & z \end{bmatrix} - \begin{bmatrix} 1.5 & -0.5 \\ 1 & 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}\begin{bmatrix} K_1 & K_2 \end{bmatrix}\right) = \alpha_c(z)$$

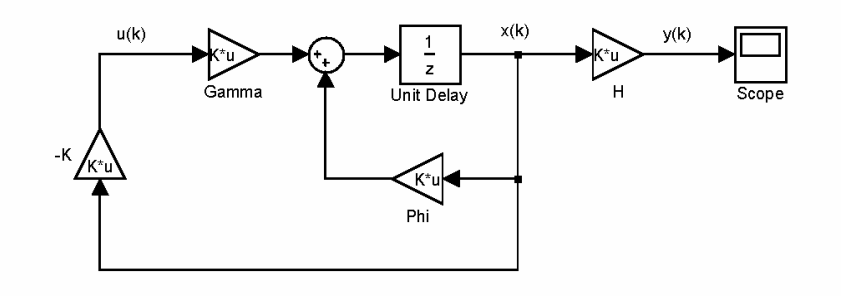
$$\det\left(\begin{bmatrix} z-1.5+K_1 & 0.5+K_2 \\ -1 & z \end{bmatrix}\right) = \alpha_c(z)$$

$$z^2 + (-1.5+K_1)z + (0.5+K_2) = \alpha_c(z) = z^2 - 1.24z + 0.45$$

=> $K_1 = 0.26$

$K_2 = -0.05$

=> $K = \begin{bmatrix} 0.26 & -0.05 \end{bmatrix}$




*Block diagram of system with full state feedback*
*(Note that at this stage the system has no reference input and is simply a regulator)*

If our state space system is in control canonical form then the full state feedback calculations are simplified,

$$\Phi = \begin{bmatrix} -a_1 & -a_2 & -a_3 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad \Gamma = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \qquad K = \begin{bmatrix} K_1 & K_2 & K_3 \end{bmatrix}$$

=> $\Phi - \Gamma K = \begin{bmatrix} -(a_1+K_1) & -(a_2+K_2) & -(a_3+K_3) \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$

which corresponds to a characteristic equation of,

$$z^3 + (a_1 + K_1)z^2 + (a_2 + K_2)z + (a_3 + K_3) = \alpha_c(z) = z^3 + \alpha_1 z^2 + \alpha_2 z + \alpha_3$$

=> $K_1 = \alpha_1 - a_1$

$K_2 = \alpha_2 - a_2$

$K_3 = \alpha_3 - a_3$

Notice that in the previous example the state space system was in control canonical form and thus the above shortcut could have been used. Verify the feedback gains using the shortcut for yourself (be careful with the signs!).

MATLAB: acker.m works by converting any state space model to control canonical form, calculating the feedback gain matrix and then converting the gain back so that it is applicable to the original state vector. place.m is more complicated and it works for systems with multiple inputs too! It uses the extra degrees of freedom provided by these inputs to not only place the eigenvalues of the closed loop system but to also 'shape' the eigenvectors such that the closed loop system is 'well conditioned'.

acker.m is suitable for low order (n < 10) systems and can handle repeated pole locations. place.m is better for high order systems but it can't handle repeated poles.

```
>> K = acker(Phi,Gamma,poles);
>> K = place(Phi,Gamma,poles)
```

**Estimator Design**

Previously we designed controllers using full state feedback. The state vector however is not usually directly available through measurements. Thus, we need to estimate the state vector given measurements y(k).

With reference to Figure 7.5 pg 496, the idea behind the estimator is to place a model of the plant in parallel with the actual plant and to drive them both with the same input. If the model's initial state vector is set equal to the plant's initial state vector then the state estimate (generated by the model) will track the actual state vector. However, there are always uncertainties in the plant model and in practice, without feedback, the state estimate would diverge from the true state. The solution is to use the measurement y(k) and to compare it with the model's predicted measurement and use the difference between the two to modify the state estimate in such a way that it converges to the true state vector.

Now for the maths of it,

$$\underline{x}(k+1) = \Phi \underline{x}(k) + \Gamma u(k) \qquad \text{(Plant dynamics equation)}$$

$$y(k) = H \underline{x}(k) \qquad \text{(Output equation)}$$

$$\underline{\hat{x}}(k+1) = \Phi \underline{\hat{x}}(k) + \Gamma u(k) + L(y(k) - H \underline{\hat{x}}(k)) \qquad \text{(Estimator equation)}$$

where $\underline{\hat{x}}(k)$ is the estimated state vector and L the estimator feedback gain. Define the state estimate error,

$$\underline{\tilde{x}}(k) \equiv \underline{x}(k) - \underline{\hat{x}}(k)$$

Then,

$$\begin{aligned}\underline{\tilde{x}}(k+1) &= \Phi \underline{x}(k) + \Gamma u(k) - \Phi \underline{\hat{x}}(k) - \Gamma u(k) - L(y(k) - H \underline{\hat{x}}(k)) \\ &= (\Phi - LH)\underline{\tilde{x}}(k)\end{aligned}$$

The final equation describes the state estimate error dynamics (i.e. how the difference between the true and estimated state vectors changes over time). We would like the error to die away to zero, typically as fast as possible, so we would like the poles of the error dynamics system matrix to be stable and fast (we will see when we do optimal estimation that there is a trade-off in choosing the speed of the estimator poles). Define the desired error dynamics poles with the characteristic equation,

$$\alpha_e(z) = z^n + \alpha_1 z^{n-1} + \ldots + \alpha_{n-1} z + \alpha_n = 0$$

Now the actual error dynamics poles are at,

$$\det(zI - (\Phi - LH)) = 0$$

Thus the estimator design equation becomes,

$$\det(zI - (\Phi - LH)) = \alpha_e(z)$$

Notes that L has as many elements as there are error dynamics poles which means that we can place the poles anywhere as long as the system is observable through the output y(k). For the system to be observable the observability matrix,

$$O = \begin{bmatrix} H \\ H\Phi \\ \vdots \\ H\Phi^{n-1} \end{bmatrix}$$

where n is the order of the system, must be of full rank. For SISO systems, this is equivalent to the test,

$$\det(O) \neq 0$$

Note that we can also use acker.m and place.m to calculate the estimator gain L. In the control problem our system matrix was,

$$\Phi - \Gamma K$$

and we found K to place the poles at desired locations. Now our system matrix is,

$$\Phi - LH$$

However, the eigenvalues of a matrix A are the same as the eigenvalues of $A^T$. So, we could just as well find the gain L to make,

$$(\Phi - LH)^T$$

have the desired poles. But,

$$(\Phi - LH)^T = \Phi^T - H^T L^T$$

which is now in the same form as the control problem (i.e. with the unknown gain in the same place). So the commands are,

```
>> L = acker(Phi',H',poles)';
>> L = place(Phi',H',poles)';
```

A second type of estimator

The estimator we have designed so far is called a predictor estimator since the state estimate at k+1 is based on measurements up to time k,

$$\underline{\hat{x}}(k+1) = \Phi \underline{\hat{x}}(k) + \Gamma u(k) + L(\underline{y}(k) - H\underline{\hat{x}}(k))$$

If the computer we are using for control is fast enough then we can use a current estimator that provides us a state vector estimate at k based on measurements up to k. Intuitively we would expect this estimate to be better for feedback purposes since it minimises the delay in the system.

The strategy is as follows. Begin with our old predictor estimate of the state vector $\bar{x}(k)$. When we get to time k we update the estimate using measurement y(k) as follows,

$$\hat{\underline{x}}(k) = \bar{\underline{x}}(k) + L_c(y(k) - H\bar{\underline{x}}(k)) \tag{1}$$

where, $\hat{x}(k)$ is now the estimate of the state vector at k using measurements up to k and $L_c$ is the current estimator feedback gain. This estimate could be used for feedback purposes. We now need to predict the state vector at time k+1 so that we can repeat the whole process. To do this, we use the system's dynamic model,

$$\bar{\underline{x}}(k+1) = \Phi\hat{\underline{x}}(k) + \Gamma u(k) \tag{2}$$

Note again that $\bar{x}(k+1)$ is our same old 'prediction' estimate of the state vector since it is based on measurements up to k only. If we substitute equation (1) into equation (2) then we can relate the current estimator gain to the predictor estimator gain (we will call the predictor estimator gain $L_p$ from now on to differentiate it from the current estimator gain $L_c$).

$$\begin{aligned}\bar{\underline{x}}(k+1) &= \Phi\cdot[\bar{\underline{x}}(k) + L_c(y(k) - H\bar{\underline{x}}(k))] + \Gamma u(k) \\ &= \Phi\bar{\underline{x}}(k) + \Gamma u(k) + \Phi L_c(y(k) - H\bar{\underline{x}}(k))\end{aligned}$$

=> $L_p = \Phi L_c$

=> $L_c = \Phi^{-1}L_p$ (3)

So the question now is, if we want a current estimator with certain error dynamics (i.e. poles) then where should the predictor estimator poles be placed (we want to solve for $L_p$ first and then calculate $L_c$)? If we look at equation (1) we see that the predictor and current estimates are related through a static equation (no dynamics). Thus, the two estimates are just different outputs of the same system i.e. they must have the same poles.

So the design procedure becomes,

- Decide on desired estimator error pole locations
- Calculate the predictor estimator gain $L_p$
- Calculate the current estimator gain $L_c$ using equation (3) above

This will allow us to implement the estimator using equations (1) and (2) and thus make the current estimate available for feedback. There is however a price to pay for using the current estimate for feedback purposes. The measurement update equation (1) can only take place when we actually get to time k. Thus we need to take the measurement, update the state as quickly as possible and then calculate the control input to the plant based on the updated state. These calculations will take a finite amount of time and if that time is a significant fraction of the sample period, then we will introduce a significant un-modelled delay into our system. The predictor estimator always has a full sample period to calculate the control.

*Example: Predictor and Current Estimator*

$$\underline{x}(k+1) = \begin{bmatrix}1.5 & -0.5\\ 1 & 0\end{bmatrix}\underline{x}(k) + \begin{bmatrix}1\\ 0\end{bmatrix}u(k) \qquad T = 0.2s$$

$$y(k) = \begin{bmatrix}0 & 1\end{bmatrix}\underline{x}(k)$$

*a) Design a predictor estimator. Place both error dynamics poles at z = 0. Draw the plant and estimator block diagrams.*

*b) Do as in (a) but now design a current estimator.*

*Design the predictor estimator,*

$$\det(zI-(\Phi-L_pH))=\alpha_e(z)$$

$$\det\left(\begin{bmatrix} z & 0 \\ 0 & z \end{bmatrix}-\begin{bmatrix} 1.5 & -0.5 \\ 1 & 0 \end{bmatrix}+\begin{bmatrix} L_1 \\ L_2 \end{bmatrix}\begin{bmatrix} 0 & 1 \end{bmatrix}\right)=\alpha_e(z)$$

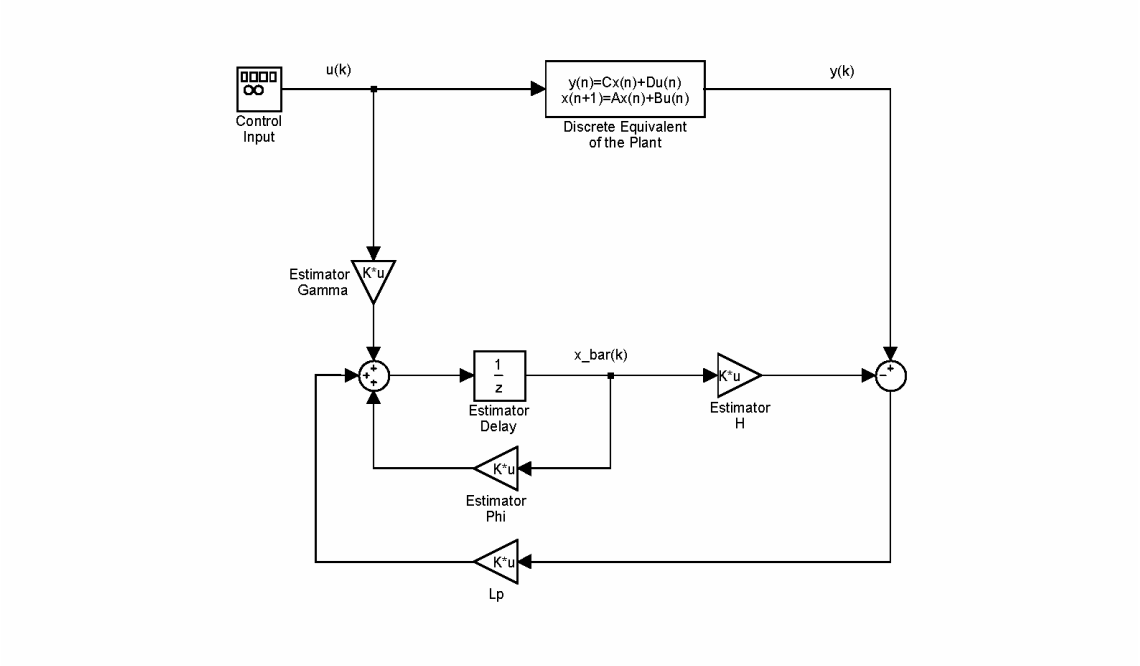
$$\det\left(\begin{bmatrix} z-1.5 & 0.5+L_1 \\ -1 & z+L_2 \end{bmatrix}\right)=\alpha_e(z)$$

$$z^2+(-1.5+L_2)z+(0.5+L_1-1.5L_2)=\alpha_c(z)=z^2$$

**=>** $L_2=1.5$

$L_1=1.75$

**=>** $L_p=\begin{bmatrix} 1.75 \\ 1.5 \end{bmatrix}$



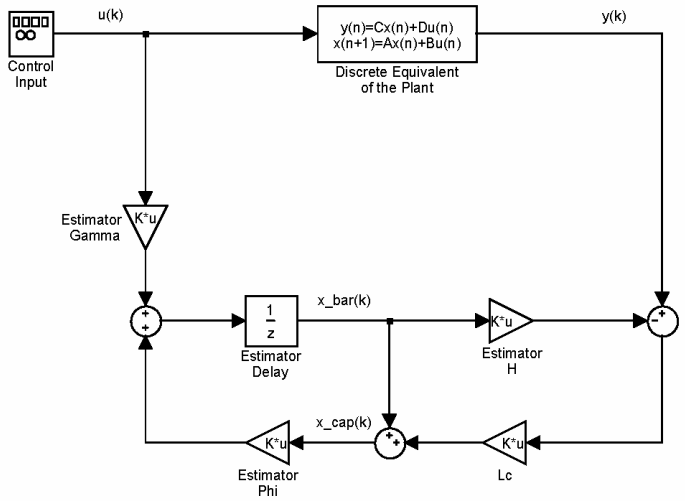

*Block diagram of predictor estimator*

*Design the current estimator,*

$$L_c=\Phi^{-1}L_p$$

$$=\begin{bmatrix} 1.5 & -0.5 \\ 1 & 0 \end{bmatrix}^{-1}\begin{bmatrix} 1.75 \\ 1.5 \end{bmatrix}=\frac{1}{0.5}\begin{bmatrix} 0 & 0.5 \\ -1 & 1.5 \end{bmatrix}\begin{bmatrix} 1.75 \\ 1.5 \end{bmatrix}$$

$$=\begin{bmatrix} 1.5 \\ 1 \end{bmatrix}$$




*Block diagram of current estimator*

Finally, if the system is in observer canonical form then the calculation of $L_p$ becomes very simple,

$$\Phi = \begin{bmatrix} -a_1 & 1 & 0 \\ -a_2 & 0 & 1 \\ -a_3 & 0 & 0 \end{bmatrix} \qquad H = \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \qquad L_p = \begin{bmatrix} L_1 \\ L_2 \\ L_3 \end{bmatrix}$$

=> $$\Phi - L_p H = \begin{bmatrix} -(a_1 + L_1) & 1 & 0 \\ -(a_2 + L_2) & 0 & 1 \\ -(a_3 + L_3) & 0 & 0 \end{bmatrix}$$

which corresponds to a characteristic equation of,

$$z^3 + (a_1 + L_1)z^2 + (a_2 + L_2)z + (a_3 + L_3) = \alpha_e(z) = z^3 + \alpha_1 z^2 + \alpha_2 z + \alpha_3$$

=> $$L_1 = \alpha_1 - a_1$$
$$L_2 = \alpha_2 - a_2$$
$$L_3 = \alpha_3 - a_3$$

**Combining the Controller and Estimator**

In this section I am only going to consider the predictor estimator case. The results are the same for the current estimator case. Let's write down all of the equations describing our closed loop system so far,

$$\underline{x}(k+1) = \Phi \underline{x}(k) + \Gamma u(k) \qquad \text{(Plant equation)}$$
$$\underline{\overline{x}}(k+1) = \Phi \underline{\overline{x}}(k) + \Gamma u(k) + L_p(y(k) - H\underline{\overline{x}}(k)) \qquad \text{(Estimator equation)}$$

$$u(k) = -K\underline{\overline{x}}(k) \qquad \text{(Feedback equation)}$$
$$y(k) = H\underline{x}(k) \qquad \text{(Output equation)}$$

Our closed loop system now has 2n states (and thus 2n poles) – n plant states and n estimator states. Since,

$$\underline{\tilde{x}}(k) = \underline{x}(k) - \underline{\bar{x}}(k) \qquad \Rightarrow \qquad \underline{\bar{x}}(k) = \underline{x}(k) - \underline{\tilde{x}}(k)$$

we can choose our state vector as,

$$\begin{bmatrix} \underline{x}(k) \\ \underline{\tilde{x}}(k) \end{bmatrix}$$

and still uniquely define the system. Remember, the estimator error dynamics are,

$$\underline{\tilde{x}}(k+1) = (\Phi - L_p H)\underline{\tilde{x}}(k)$$

and the control law can be written as,

$$u(k) = -K\underline{x}(k) + K\underline{\tilde{x}}(k)$$

So the entire closed loop system can be described by,

$$\begin{bmatrix} \underline{x}(k+1) \\ \underline{\tilde{x}}(k+1) \end{bmatrix} = \begin{bmatrix} \Phi - \Gamma K & \Gamma K \\ \underline{0} & \Phi - L_p H \end{bmatrix} \begin{bmatrix} \underline{x}(k) \\ \underline{\tilde{x}}(k) \end{bmatrix}$$

with poles at,

$$\det\left( \begin{bmatrix} zI & \underline{0} \\ \underline{0} & zI \end{bmatrix} - \begin{bmatrix} \Phi - \Gamma K & \Gamma K \\ \underline{0} & \Phi - L_p H \end{bmatrix} \right) = 0$$

$$\det\left( \begin{bmatrix} zI - (\Phi - \Gamma K) & -\Gamma K \\ \underline{0} & zI - (\Phi - L_p H) \end{bmatrix} \right) = 0$$

$$\det(zI - (\Phi - \Gamma K)) \cdot \det(zI - (\Phi - L_p H)) = 0$$

$$\alpha_c(z) \cdot \alpha_e(z) = 0$$

This result is called the separation principle and shows that we can design the controller and estimator independently of each other and when we put it all together (i.e. use the estimated state vector for feedback and not the actual state vector), we will still end up with the same poles! If we wanted to see what our combined controller/estimator compensator looked like as a transfer function D(z) we could write,

$$\underline{\bar{x}}(k+1) = \Phi\underline{\bar{x}}(k) + \Gamma u(k) + L_p(y(k) - H\underline{\bar{x}}(k))$$

$$u(k) = -K\underline{\bar{x}}(k)$$

which simplifies to,

$$\underline{\bar{x}}(k+1) = [\Phi - \Gamma K - L_p H]\underline{\bar{x}}(k) + L_p y(k)$$

$$u(k) = -K\underline{\bar{x}}(k)$$

These are the compensator state space equations, with input y(k) and output u(k). By taking the Z-transform of both equations we get,

$$zI\underline{\bar{X}}(z) = [\Phi - \Gamma K - L_p H]\underline{\bar{X}}(z) + L_p Y(z)$$

$$U(z) = -K\underline{\bar{X}}(z)$$

$$\Rightarrow \qquad \underline{\bar{X}}(z) = [zI - \Phi + \Gamma K + L_p H]^{-1} L_p Y(z)$$

$$U(z) = -K\underline{\bar{X}}(z)$$

=> $$\frac{U(z)}{Y(z)} = -K[zI - \Phi + \Gamma K + L_p H]^{-1} L_p = \frac{-K \cdot adj[zI - \Phi + \Gamma K + L_p H] \cdot L_p}{\det(zI - \Phi + \Gamma K + L_p H)}$$

with poles and zeros at,

Zeros at: $-K \cdot adj[zI - \Phi + \Gamma K + L_p H] L_p = 0$

Poles at: $\det(zI - \Phi + \Gamma K + L_p H) = 0$

Note: Other than for interest, there is no real need to calculate D(z) when we do a state space design.

Guidelines for pole placement

Most of the time we are given time response specifications (or something similar) for a system which define where 2 of the closed loop poles need to be. But what if we have a 4th order system? Where do we put the other 2 poles? Here are some guidelines.

- Place 2 dominant closed loop poles to meet the specifications
- Place the rest of them at a higher frequency so that they have little effect on the response
- Don't move open loop poles unnecessarily (uses control effort!)
- Don't change the frequency of high frequency poles. Just add some damping if necessary
- Estimator poles are typically placed 2 to 6 times faster than controller poles
- Optimal control/estimation techniques relieve the burden of deciding where to place poles

## Introducing the Reference Input

We now want to introduce a reference input r(k) into our system. We would like to do this in such a way that the system output y(k) is equal to the reference input in the steady state (i.e. design for unity DC gain). In the steady state the state vector will be constant at $x_{SS}$ and so will the control input at $u_{SS}$. For a system with one or more free integrators, $u_{SS} = 0$ but for a system with no free integrators, $u_{SS}$ = constant. Consider the reference input structure shown below,

*State command reference input structure (Draw in class)*

This structure is called the state command structure.

Note: $N_u = 0$ for system with one or more free integrators
No control is provided to the plant via the state feedback path in the steady state since $x_r$ is defined to be equal to $x_{SS}$.

Now how do we calculate $N_x$ and $N_u$ such that $y_{SS} = r$? Go to the maths!

$$\underline{x}_{ss} = \Phi \underline{x}_{ss} + \Gamma u_{ss}$$
$$y_{ss} = H \underline{x}_{ss}$$

with,

$$\underline{x}_{ss} = N_x r$$
$$u_{ss} = N_u r$$

So,

$$N_x r = \Phi N_x r + \Gamma N_u r$$
$$r = H N_x r$$

=> $$\begin{bmatrix} \underline{0} \\ 1 \end{bmatrix} = \begin{bmatrix} \Phi - I & \Gamma \\ H & 0 \end{bmatrix} \begin{bmatrix} N_x \\ N_u \end{bmatrix}$$

=> $$\begin{bmatrix} N_x \\ N_u \end{bmatrix} = \begin{bmatrix} \Phi - I & \Gamma \\ H & 0 \end{bmatrix}^{-1} \begin{bmatrix} \underline{0} \\ 1 \end{bmatrix}$$

Note that our control law has now become,

$$\begin{aligned} u(k) &= -K[x(k) - N_x r(k)] + N_u r(k) \\ &= -Kx(k) + (KN_x + N_u) r(k) \\ &= -Kx(k) + \overline{N} r(k) \end{aligned}$$

where,

$$\overline{N} = KN_x + N_u$$

*Example: Closed loop system without estimator*

$$\underline{x}(k+1) = \Phi \underline{x}(k) + \Gamma u(k) \qquad \textit{(Open loop system)}$$
$$u(k) = -Kx(k) + \overline{N} r(k) \qquad \textit{(Control Law)}$$

=> $$\begin{aligned} \underline{x}(k+1) &= (\Phi - \Gamma K)\underline{x}(k) + \Gamma \overline{N} r(k) \\ &= \Phi_{CL} \underline{x}(k) + \Gamma_{CL} r(k) \end{aligned} \qquad \textit{(Closed loop system)}$$

When you have an estimator you introduce the reference in the same way. This ensures that the estimator gets the same control input as the plant does. Thus, reference changes will not excite estimator error dynamics. With the reference introduced in this way the closed loop system will have the n controller poles and the n estimator poles but the n estimator poles will all have zeros on top of them (this is only the case for the reference input, not for the disturbance input). Thus a reference step response of the closed loop system will only display the controller pole dynamics.

*Example: Putting it all together*

*Consider the continuous state space system below,*

$$\dot{\underline{x}}(t) = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \underline{x}(t) + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u(t)$$

$$y(t) = \begin{bmatrix} 1 & 0 \end{bmatrix} \underline{x}(t)$$

*Design a discrete controller using state space methods such that the closed loop system has unity DC gain, an overshoot of 5% and a rise time of approximately 1.8s. Use a current estimator and place the estimator poles at z = 0. Use a sample period T of 0.2s.*

*Specs:*

$M_p = 5\%$ => $\zeta = 0.7$

$t_r = 1.8s$ => $\omega_n = 1 \ \ rad/s$

$$s_{CL} = -\zeta\omega_n \pm j\omega_n\sqrt{1-\zeta^2} = -0.7 \pm j0.7$$

$$z_{CL} = e^{s_{CL}T} = 0.86 \pm j0.12$$

*System:*

$$\begin{aligned}\Psi &= I + FT/2! + (FT)^2/3! + \dots \\ &= \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix} + \frac{1}{2}\begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}T + \frac{1}{6}\begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}\begin{bmatrix}0 & 1\\ 0 & 0\end{bmatrix}T^2 + \dots \qquad \textit{(Note: series truncates after term 2)} \\ &= \begin{bmatrix}1 & T/2\\ 0 & 1\end{bmatrix}\end{aligned}$$

$$\Phi = I - FT\Psi = \begin{bmatrix}1 & T\\ 0 & 1\end{bmatrix}$$

$$\Gamma = T\Psi G = \begin{bmatrix}T^2/2\\ T\end{bmatrix}$$

*which gives the discrete equivalent system,*

$$\underline{x}(k+1) = \begin{bmatrix}1 & 0.2\\ 0 & 1\end{bmatrix}\underline{x}(k) + \begin{bmatrix}0.02\\ 0.2\end{bmatrix}u(k) \qquad T = 0.2s$$

$$y(k) = \begin{bmatrix}1 & 0\end{bmatrix}\underline{x}(k)$$

*Control:*

$$\alpha_c(z) = (z-0.86)^2 + (0.12)^2 = z^2 - 1.72z + 0.754$$

$$\det(zI - (\Phi - \Gamma K)) = \alpha_c(z)$$

=> $K = \begin{bmatrix}0.85 & 1.31\end{bmatrix}$

*Estimator:*

$$\alpha_e(z) = z^2$$

$$\det(zI - (\Phi - L_p H)) = \alpha_e(z)$$

=> $L_p = \begin{bmatrix}2\\ 5\end{bmatrix}$

=> $L_c = \Phi^{-1}L_p = \begin{bmatrix}1\\ 5\end{bmatrix}$

*Reference:*

$$\begin{bmatrix} N_x \\ N_u \end{bmatrix} = \begin{bmatrix} \Phi - I & \Gamma \\ H & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0.2 & 0.02 \\ 0 & 0 & 0.2 \\ 1 & 0 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

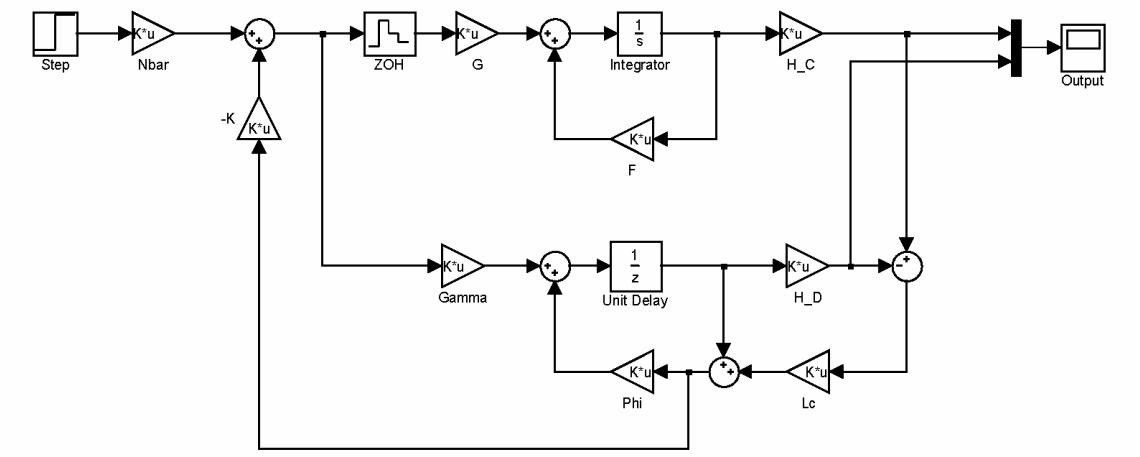
$$= \frac{1}{0.04} \begin{bmatrix} 0 & 0 & 0.04 \\ 0.2 & -0.02 & 0 \\ 0 & 0.2 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

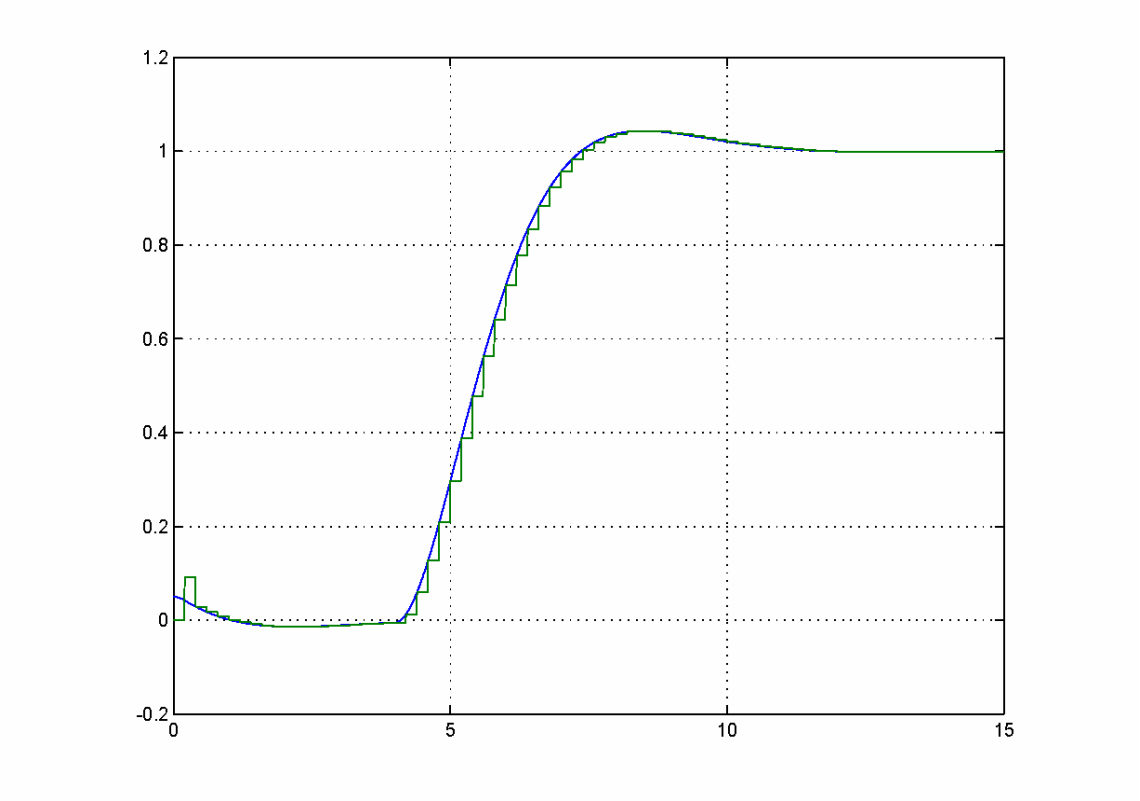
$$= \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

**=>** $N_x = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \qquad N_u = 0$ *(Note: Plant has 2 free integrators)*

**=>** $\overline{N} = KN_x + N_u = \begin{bmatrix} 0.85 & 1.31 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + 0 = 0.85$

*Simulation results,*






*The simulation begins with a difference between the true and estimated state vectors. Note how the estimator tracks the plant exactly after two cycles (because both estimator poles are at z=0). A unit step command is applied at 4s. The closed loop specifications have been met.*

**Integral Control**

At this stage the reference input is introduced through the gain $\bar{N}$. However, the value of $\bar{N}$ is sensitive to the parameters of the plant and thus the DC gain of the closed loop system will not be robust to plant uncertainties. Practically this presents a big problem because there are always uncertainties in the plant parameters but it is often very important that the output track the reference exactly in the steady state (unity DC gain).

Integral control can be used to solve this problem. More specifically, we seek to integrate the error between the reference input and the output and use the integrated state as part of our feedback. Doing this will ensure that our closed loop system (if stable) will have unity DC gain. To better understand this, consider an error signal that enters an integrator. For the output of the integrator to settle (i.e. reach steady state) then the error signal MUST go to zero. Let's look at the maths of this. Consider the open loop state space system,

$$\underline{x}(k+1) = \Phi \underline{x}(k) + \Gamma u(k)$$
$$y(k) = H \underline{x}(k)$$

It is desired that the output y(k) track a reference input signal r(k) with zero steady state error in spite of uncertainties in the plant parameters. To this end, we define a new state $x_I(k)$ that is the integral of the error between the reference and the output of the plant. We could use a number of discrete integration processes here - they all have the important property that the integration state reaches steady state when the input signal to the integrator goes to zero. Thus, it makes sense to use the simplest numerical integration routine i.e. forward rectangular integration. Thus,

$$\begin{aligned} x_I(k+1) &= x_I(k) + T\big(r(k) - y(k)\big) \\ &= x_I(k) + T\big(r(k) - H\underline{x}(k)\big) \\ &= x_I(k) - TH\underline{x}(k) + Tr(k) \end{aligned}$$

We now have an extra dynamic equation describing what is now an extra state in the system. We thus augment this equation to the original dynamics to for the complete open loop system dynamics with a block diagram as shown below,

$$\begin{bmatrix} \underline{x}(k+1) \\ x_I(k+1) \end{bmatrix} = \begin{bmatrix} \Phi & \underline{0} \\ -TH & 1 \end{bmatrix} \begin{bmatrix} \underline{x}(k) \\ x_I(k) \end{bmatrix} + \begin{bmatrix} \Gamma \\ 0 \end{bmatrix} u(k) + \begin{bmatrix} \underline{0} \\ T \end{bmatrix} r(k)$$

$$y(k) = \begin{bmatrix} H & 0 \end{bmatrix} \begin{bmatrix} \underline{x}(k) \\ x_I(k) \end{bmatrix}$$

*Block diagram of the compete augmented open loop system (Draw in class)*

Now the rest of the problem is easy. We have a new open loop system and we simply perform full state feedback to stabilise it (where the last state happens to be our integral state). Writing the open loop system as follows,

$$\begin{aligned} \underline{x}_a(k+1) &= \Phi_a \underline{x}_a(k) + \Gamma_a u(k) + \Gamma'_a r(k) \\ y(k) &= H_a \underline{x}_a(k) \end{aligned}$$

where the definition of the matrices with subscript $a$ is clear. Define the control law,

$$u(k) = -K_a \underline{x}_a(k) = -\begin{bmatrix} K & K_I \end{bmatrix} \begin{bmatrix} \underline{x}(k) \\ x_I(k) \end{bmatrix}$$

The closed loop system is thus,

$$\begin{aligned} \underline{x}_a(k+1) &= \Phi_a \underline{x}_a(k) - \Gamma_a K_a \underline{x}_a(k) + \Gamma'_a r(k) \\ &= \big[\Phi_a - \Gamma_a K_a\big] \underline{x}_a(k) + \Gamma'_a r(k) \end{aligned}$$

Given the desired characteristic equation $\alpha_c(z)$ for all of the closed loop poles (including the integrator pole now) we complete the design by setting,

$$\det(zI - (\Phi_a - \Gamma_a K_a)) = \alpha_c(z)$$

and solving for $K_a$. A block diagram of the closed loop system is shown below.

*Block diagram of closed loop system with integral control (Draw in class)*

Note that there is no need to explicitly go through the process of introducing a reference input anymore – the reference input is already introduced through $\Gamma_a'$ as shown by the closed loop dynamics. However, there is one final trick we can pull with integral control. If we adapt the control law slightly to be,

$$u(k) = -K_a \underline{x}_a(k) + \bar{N} r(k)$$

then it can be shown (not shown here nor is the derivation in the book – just accept for now!), that the closed loop system has a zero at,

$$z_I = 1 + \frac{TK_I}{\bar{N}}$$

Thus, if we choose $\bar{N}$ such that the zero lies on the closed loop integrator pole then we will not see the effect of the integrator dynamics in the closed loop system. This is very useful because we often want our integrator to have slow dynamics since it is only meant to strongly influence the steady state response. With this adaptation to the control law, the block diagram of the closed loop system looks as follows,

*Block diagram of closed loop system with integral control and feed-forward (Draw in class)*

In state space form the closed loop system is then,

$$\begin{aligned}\underline{x}_a(k+1) &= \Phi_a \underline{x}_a(k) + \Gamma_a\left(-K_a \underline{x}_a(k) + \bar{N}r(k)\right) + \Gamma'_a r(k) \\ &= \left[\Phi_a - \Gamma_a K_a\right]\underline{x}_a(k) + \left[\Gamma_a \bar{N} + \Gamma'_a\right] r(k)\end{aligned}$$

$$y(k) = H_a \underline{x}_a(k)$$

To summarise the design procedure then,

1) Augment the original n[th] order open loop system with the integrator dynamics
2) This in now your new open loop system
3) Place all n+1 poles of your new open loop system as desired using full state feedback
4) Calculate $\bar{N}$ such that the closed loop has a zero on the integrator pole

Finally, note that if you have designed an integral controller then it is not necessary to design an estimator to estimate the integrator state. This is because the integrator state exists inside your controller and is thus know 100% accurately. If you were to estimate it you would only loose accuracy! Thus you simply design an estimator as before for the original open loop system. The following example should clarify integral control further.

*Example: Integral Control of a Satellite*

*Consider the following discrete state space model of a satellite,*

$$\begin{bmatrix}\theta(k+1)\\ \omega(k+1)\end{bmatrix} = \begin{bmatrix}1 & 0.2\\ 0 & 1\end{bmatrix}\begin{bmatrix}\theta(k)\\ \omega(k)\end{bmatrix} + \begin{bmatrix}0.2\\ 2\end{bmatrix}u(k) \qquad T = 0.2s$$

$$\left[\theta(k+1)\right] = \begin{bmatrix}1 & 0\end{bmatrix}\begin{bmatrix}\theta(k)\\ \omega(k)\end{bmatrix}$$

*Note that the model is the same as that used in your notes on classical control (when we did the root locus example). Design a control system for the satellite to regulate the angle output with the same specifications as before i.e. 0.5Hz bandwidth and a peak overshoot of less than 20%. The steady state performance of the controller should not be sensitive to uncertainties in the plant gain. Furthermore, only the output angle of the satellite can be measured. Use a predictor estimator to estimate the state vector.*

*Our strategy is to design a full state feedback controller with integral control to make the system insensitive to the plant gain in the steady state. We will then design an estimator to estimate the satellite angle and angular rate from an angle measurement alone.*

*Let,*

$$x_I(k+1) = x_I(k) + T\left(\theta_r(k) - \theta(k)\right) \qquad T = 0.2s$$

*where $\theta_r(k)$ is the reference angle for the satellite control system. Augmenting the integrator dynamics to the plant dynamics,*

$$\begin{bmatrix}\theta(k+1)\\ \omega(k+1)\\ x_I(k+1)\end{bmatrix} = \begin{bmatrix}1 & 0.2 & 0\\ 0 & 1 & 0\\ -0.2 & 0 & 1\end{bmatrix}\begin{bmatrix}\theta(k)\\ \omega(k)\\ x_I(k)\end{bmatrix} + \begin{bmatrix}0.2\\ 2\\ 0\end{bmatrix}u(k) + \begin{bmatrix}0\\ 0\\ 0.2\end{bmatrix}\theta_r(k)$$

$$= \Phi_a \underline{x}_a(k) + \Gamma_a u(k) + \Gamma'_a \theta_r(k)$$

*This is now our new open loop system and we simply place all the poles (note, there are three poles now) as desired using a full state feedback control law,*

$$u(k) = -K_a \underline{x}_a(k) = -\begin{bmatrix} K & K_I \end{bmatrix} \underline{x}_a(k)$$

*This control law gives closed loop poles at,*

$$\det(zI - (\Phi_a - \Gamma_a K_a)) = \alpha_c(z)$$

*Now, the closed loop specifications determine where two of the three system poles go. One option is for us to place the third pole at a much higher frequency than the two dominant poles to minimise its effect. However, we know that we can place a real zero arbitrarily by changing the control law to,*

$$u(k) = -K_a \underline{x}_a(k) + \bar{N} r(k)$$

*Thus our strategy is to place two of the poles to meet the specifications and place the third (the integrator) pole at z = 0.9. We choose this value because it moves the integrator pole very little (it starts at z = 1), which will require little control effort. Without a zero, this pole would have a dominant effect. However, we will then choose* $\bar{N}$ *to place the zero on the pole at z = 0.9 to cancel its effect. From the example in your classical control notes, the two dominant poles should be placed at,*

$$z_{1,2} = 0.62 \pm j0.38$$

*Thus, the characteristic equation is,*

$$\begin{aligned} \alpha_c(z) &= (z - 0.9)\left((z - 0.62)^2 + (0.38)^2\right) \\ &= (z - 0.9)\left(z^2 - 1.24z + 0.529\right) \\ &= z^3 - 2.14z^2 + 1.645z - 0.476 \end{aligned}$$

*Now,*

$$\det\left(\begin{bmatrix} z & 0 & 0 \\ 0 & z & 0 \\ 0 & 0 & z \end{bmatrix} - \left(\begin{bmatrix} 1 & 0.2 & 0 \\ 0 & 1 & 0 \\ -0.2 & 0 & 1 \end{bmatrix} - \begin{bmatrix} 0.2 \\ 2 \\ 0 \end{bmatrix} \begin{bmatrix} K_\theta & K_\omega & K_I \end{bmatrix}\right)\right) = \alpha_c(z)$$

**=>**

$$\det\left(\begin{bmatrix} z - 1 + 0.2K_\theta & -0.2 + 0.2K_\omega & 0.2K_I \\ 2K_\theta & z - 1 + 2K_\omega & 2K_I \\ 0.2 & 0 & z - 1 \end{bmatrix}\right) = \alpha_c(z)$$

*The weave method for calculating the determinant for a three by three matrix could then be used and the coefficients of the characteristic equation matched up to solve for the gains. I am too lazy to do this (!) and am going to use MATLAB instead (this is what you would do in practice). In the exam I will not send you down a 'brute force' maths road. The questions will either be designed to work out a little easier or you will be asked to work up to a point such as I have done above and then explain how you would continue by hand and how you would do it with MATLAB. In MATLAB,*

```
>> zp = [0.9, 0.62 + j*0.38, 0.62 – j*0.38];
>> K = acker(Phi_a, Gamma_a, zp);
```

*This gives a feedback gain of,*

$$K_a = \begin{bmatrix} 0.8759 & 0.3424 & -0.3610 \end{bmatrix}$$

*Thus,*

$$K = \begin{bmatrix} 0.8759 & 0.3424 \end{bmatrix}$$
$$K_I = -0.3610$$

*The equation for placing the zero is,*

$$z_I = 1 + \frac{TK_I}{\bar{N}}$$

*Calculating* $\bar{N}$ *such that the zero lies at z = 0.9 gives,*

$$\begin{aligned}\bar{N} &= \frac{TK_I}{z_I - 1} \\ &= \frac{(0.2)(-0.3610)}{0.9 - 1} \\ &= 0.722\end{aligned}$$

*We need only estimate the original un-augmented state vector since the integral state is always exactly know (it exists inside the control computer). For a predictor estimator with poles at the origin set,*

$$\det(zI - (\Phi - L_p H)) = z^2$$

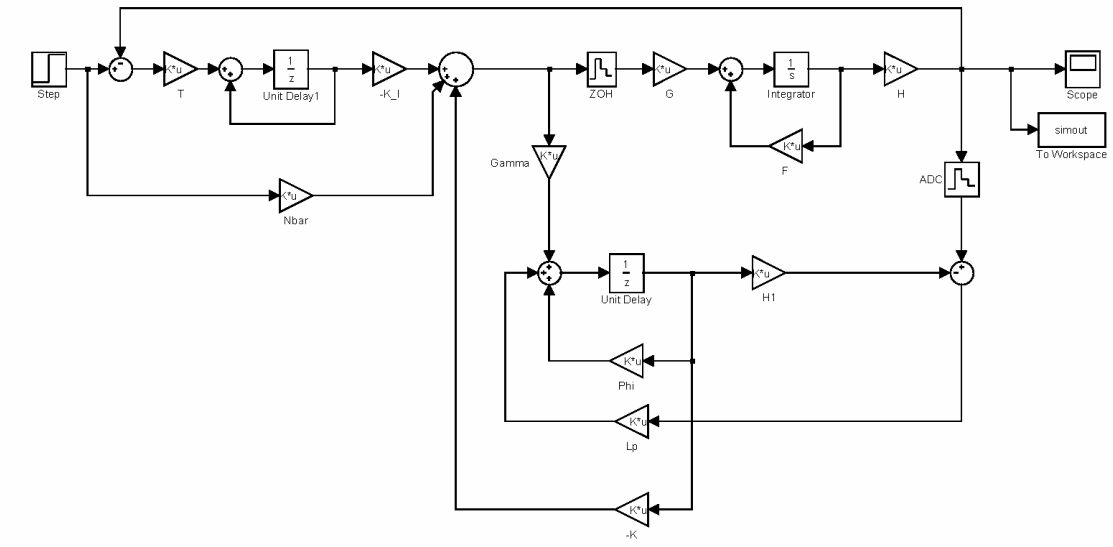
*with,*

$$\Phi = \begin{bmatrix} 1 & 0.2 \\ 0 & 1 \end{bmatrix} \qquad H = \begin{bmatrix} 1 & 0 \end{bmatrix}$$

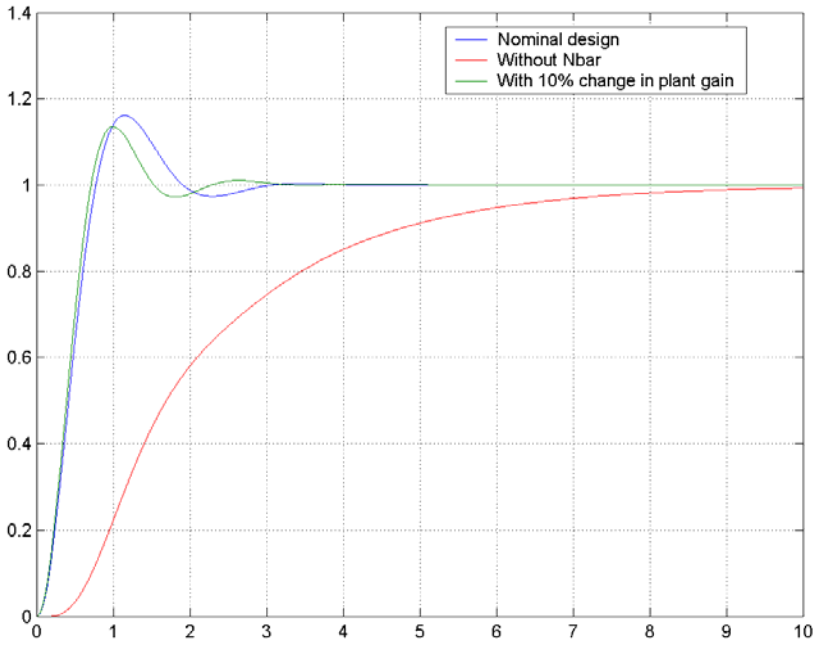
*and solve for* $L_p$*. This could easily be done by hand. Using MATLAB gives,*

$$L_p = \begin{bmatrix} 2 \\ 5 \end{bmatrix}$$

*A block diagram of the controller is given below with the step response shown. Note that the transient response specifications are easily met without the need for iteration as we had to when we designed the lead network in the classical control notes. A simulation is also run with the plant input gain varied by 10%. We see that although the transient response is disturbed, the DC gain of the closed loop system remains unity. Finally, the effect of leaving out the feed-forward gain* $\bar{N}$ *is also shown. Of course if you knew you were going to leave out this gain then you could design for a faster closed loop integrator pole.*




Simulink block diagram of the closed loop system




*Step response of the closed loop system*