CALIFORNIA INSTITUTE OF TECHNOLOGY
Control and Dynamical Systems

CDS 110b

R. M. Murray

Lecture 2 – LQR Control

11 January 2006

This lecture provides a brief derivation of the linear quadratic regulator (LQR) and describes how to design an LQR-based compensator. The use of integral feedback to eliminate steady state error is also described. Two design examples are given: lateral control of the position of a simplified vectored thrust aircraft and speed control for an automobile.

1 Linear Quadratic Regulator

The finite horizon, linear quadratic regulator (LQR) is given by

$$\dot{x} = Ax + Bu \qquad x \in \mathbb{R}^n, u \in \mathbb{R}^n, x_0 \text{ given}$$
$$\tilde{J} = \frac{1}{2}\int_0^T \left(x^T Q x + u^T R u\right)\, dt + \frac{1}{2}x^T(T) P_1 x(T)$$

where $Q \geq 0$, $R > 0$, $P_1 \geq 0$ are symmetric, positive (semi-) definite matrices. Note the factor of $\frac{1}{2}$ is left out, but we included it here to simplify the derivation. Gives same answer (with $\frac{1}{2}x$ cost).

Solve via maximum principle:

$$H = x^T Q x + u^T R u + \lambda^T (Ax + Bu)$$

$$\begin{aligned}
\dot{x} &= \left(\frac{\partial H}{\partial \lambda}\right)^T &&= Ax + Bu \qquad && x(0) = x_0 \\
-\dot{\lambda} &= \left(\frac{\partial H}{\partial x}\right)^T &&= Qx + A^T\lambda \qquad && \lambda(T) = P_1 x(T) \\
0 &= \frac{\partial H}{\partial u} &&= Ru + \lambda^T B \quad \Longrightarrow \quad && u = -R^{-1}B^T\lambda.
\end{aligned}$$

This gives the optimal solution. Apply by solving *two point boundary value problem* (hard).

Alternative: guess the form of the solution, $\lambda(t) = P(t)x(t)$. Then

$$\dot{\lambda} = \dot{P}x + P\dot{x} = \dot{P}x + P(Ax - BR^{-1}B^T P)x$$
$$-\dot{P}x - PAx + PBR^{-1}BPx = Qx + A^T P x.$$

This equation is satisfied if we can find $P(t)$ such that

$$-\dot{P} = PA + A^T P - PBR^{-1}B^T P + Q \qquad P(T) = P_1$$

Remarks:

1. This ODE is called *Riccati ODE*.
2. Can solve for $P(t)$ backwards in time and then apply

   $$u(t) = -R^{-1}B^T P(t)x.$$

   This is a (time-varying) *feedback* control $\implies$ tells you how to move from *any* state to the origin.
3. Variation: set $T = \infty$ and eliminate terminal constraint:

   $$\begin{aligned} J &= \int_0^\infty (x^T Q x + u^T R u)\, dt \\ u &= -\underbrace{R^{-1}B^T P}_{K} x \qquad \text{Can show } P \text{ is } \textit{constant} \\ 0 &= PA + A^T P - PBR^{-1}B^T P + Q \end{aligned}$$

   This equation is called the *algebraic Riccati equation*.
4. In MATLAB, `K = lqr(A, B, Q, R)`.
5. Require $R > 0$ but $Q \geq 0$. Let $Q = H^T H$ (always possible) so that $L = \int_0^\infty x^T H^T H x + u^T R u\, dt = \int_0^\infty \|Hx\|^2 + u^T R u\, dt$. Require that (A, H) is *observable*. Intuition: if not, dynamics may not affect cost $\implies$ ill-posed.

2 Choosing LQR weights

$$\dot{x} = Ax + Bu \qquad J = \int_0^\infty \overbrace{\left(x^T Q x + u^T R u + x^T S u\right)}^{L(x,u)}\, dt,$$

where the S term is almost always left out.

Q: How should we choose Q and R?

1. Simplest choice: $Q = I$, $R = \rho I$ $\implies$ $L = \|x\|^2 + \rho\|u\|^2$. Vary ρ to get something that has good response.

2. Diagonal weights

$$Q = \begin{bmatrix} q_1 & & \\ & \ddots & \\ & & q_n \end{bmatrix} \qquad R = \rho \begin{bmatrix} r_1 & & \\ & \ddots & \\ & & r_n \end{bmatrix}$$

Choose each q_i to given equal effort for same "badness". Eg, x_1 = distance in meters, x_3 = angle in radians:

$$\text{1 cm error OK} \implies q_1 = \left(\frac{1}{100}\right)^2 \qquad q_1 x_1^2 = 1 \text{ when } x_1 = 1 \text{ cm}$$

$$\frac{1}{60} \text{ rad error OK} \implies q_3 = (60)^2 \qquad q_3 x_3^2 = 1 \text{ when } x_3 = \frac{1}{60} \text{ rad}$$

Similarly with r_i. Use ρ to adjust input/state balance.

3. Output weighting. Let $z = Hx$ be the output you want to keep small. Assume (A, H) observable. Use

$$Q = H^T H \qquad R = \rho I \qquad \implies \text{trade off } \|z\|^2 \text{ vs } \rho\|u\|^2$$

4. Trial and error (on *weights*)

3 State feedback with reference trajectory

Suppose we are given a system $\dot{x} = f(x, u)$ and a feasible trajectory (x_d, u_d). We wish to design a compensator of the form $u = \alpha(x, x_d, u_d)$ such that $\lim_{t\to\infty} x - x_d = 0$. This is known as the *trajectory tracking* problem.

To design the controller, we construct the error system. We will assume for simplicity that $f(x, u) = f(x) + g(x)u$ (i.e., the system is nonlinear in the state, but linear in the input; this is often the case in applications). Let $e = x - x_d$, $v = u - u_d$ and compute the dynamics for the error:

$$\begin{aligned} \dot{e} = \dot{x} - \dot{x}_d &= f(x) + g(x)u - f(x_d) + g(x_d)u_d \\ &= f(e + x_d) - f(x_d) + g(e + x_d)(v + u_d) - g(x_d)u_d \\ &= F(e, v, x_d(t), u_d(t)) \end{aligned}$$

In general, this system is time varying.

For trajectory tracking, we can assume that e is small (if our controller is doing a good job) and so we can linearize around $e = 0$:

$$\dot{e} \approx A(t)e + B(t)v$$

where

$$A(t) = \left.\frac{\partial F}{\partial e}\right|_{(x_d(t),u_d(t))} \qquad B(t) = \left.\frac{\partial F}{\partial v}\right|_{(x_d(t),u_d(t)} .$$

It is often the case that $A(t)$ and $B(t)$ depend only on x_d, in which case it is convenient to write $A(t) = A(x_d)$ and $B(t) = B(x_d)$.

Assume now that x_d and u_d are either constant or slowly varying (with respect to the performance criterion). This allows us to consider just the (constant) linear system given by $(A(x_d), B(x_d))$. If we design a state feedback controller $K(x_d)$ for each x_d, then we can regulate the system using the feedback

$$v = K(x_d)e.$$

Substituting back the definitions of e and v, our controller becomes

$$u = K(x_d)(x - x_d) + u_d$$

This form of controller is called a *gain scheduled* linear controller with *feedforward* u_d.

In the special case of a linear system

$$\dot{x} = Ax + Bu,$$

it is easy to see that the error dynamics are identical to the system dynamics (so $\dot{e} = Ae + Bv$) and in this case we do not need to schedule the gain based on x_d; we can simply compute a constant gain K and write

$$u = K(x - x_d) + u_d.$$

4 Integral action

The controller based on state feedback achieves the correct steady state response to reference signals by careful computation of the reference input u_d. This requires a perfect model of the process in order to insure that (x_d, u_d) satisfies the dynamics of the process. However, one of the primary uses of feedback is to allow good performance in the presence of uncertainty, and hence requiring that we have an *exact* model of the process is undesirable. An alternative to calibration is to make use of integral feedback, in which the controller uses an integrator to provide zero steady state error. The basic concept of integral feedback is described in more detail in AM05 [1]; we focus here on the case of integral action with a reference trajectory.

The basic approach in integral feedback is to create a state within the controller that computes the integral of the error signal, which is then used as a feedback term. We do this by augmenting the description of the system with a new state z:

$$\frac{d}{dt}\begin{bmatrix} x \\ z \end{bmatrix} = \begin{bmatrix} Ax + Bu \\ y - r \end{bmatrix} = \begin{bmatrix} Ax + Bu \\ Cx - r \end{bmatrix}$$

The state z is seen to be the integral of the error between the desired output, r, and the actual output, y. Note that if we find a compensator that stabilizes the system then necessarily we will have $\dot{z} = 0$ in steady state and hence $y = r$ in steady state.

Given the augmented system, we design a state space controller in the usual fashion, with a control law of the form

$$u = -K(x - x_d) - K_i z + u_d$$

where K is the usual state feedback term, K_i is the integral term and u_d is used to set the reference input for the nominal model. For a step input, the resulting equilibrium point for the system is given as

$$x_e = -(A - BK)^{-1}B(u_d - K_i z_e)$$

Note that the value of z_e is not specified, but rather will automatically settle to the value that makes $\dot{z} = y - r = 0$, which implies that at equilibrium the output will equal the reference value. This holds independently of the specific values of A, B and K, as long as the system is stable (which can be done through appropriate choice of K and K_i).

The final compensator is given by

$$\begin{aligned} u &= -K(x - x_e) - K_i z + u_d \\ \dot{z} &= y - r, \end{aligned}$$

where we have now included the dynamics of the integrator as part of the specification of the controller.

5 Example: Speed Control

The following example is adapted from AM05 [1].

Dynamics The speed control system of a car is one of the most common control systems encountered in everyday life. The system attempts to keep the speed of the car constant in spite of disturbances caused by changing slope of the road and variations in the wind and road conditions. The system measures the speed of the car and adjusts the throttle.

To model the complete system we start with the block diagram in Figure 1. Let v be the speed of the car and v_r the desired speed. The controller, which typically is of the PI type, receives the signals v and v_r and generates a control signal u that is sent to an actuator that controls throttle position. The throttle in turn controls the torque T delivered by the engine, which is then transmitted through gears and the wheels, generating a force F that moves the car. There are disturbance forces F_d due to variations in the slope of the road, θ, rolling resistance and aerodynamic forces. The cruise controller has a man-machine interface that allows the driver to set and modify the desired speed. There are also functions that

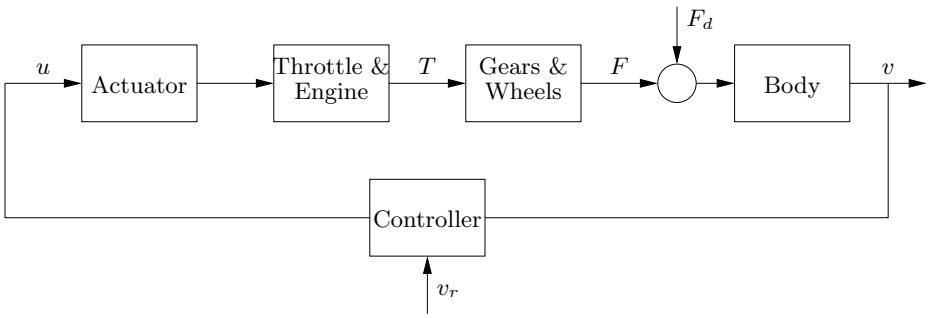

Figure 1: Block diagram of a speed control system for an automobile.

disconnects cruise control when the brake or the accelerator is touched as well as functions to resume cruise control,

The system has many components actuator, engine, transmission, wheels and car body, and a detailed model can be quite complicated. It turns out that however that the model required to design the cruise controller can be quite simple. In essence the model should describe how the car's speed is influenced by the slope of the road and the control signal u that drives the throttle actuator.

To model the system it is natural to start with a momentum balance for the car body. Let v be the speed, let m be the total mass of the car including passengers, let F the force generated by the contact of the wheels with the road, and let F_d be the disturbance force. The equation of motion of the car is simply

$$m\frac{dv}{dt} = F - F_d. \tag{1}$$

The force F is generated by the engine. The engine torque T is proportional to the rate of fuel injection and thus also to the signal u that controls throttle position. The torque depends on engine speed ω which is expressed by the torque curve shown in Figure 2. The torque has a maximum T_m at ω_m. Let n be the gear ratio and r the wheel radius. The engine speed is related to the velocity through

$$\omega = \frac{n}{r}v =: \alpha_n v,$$

and the driving force can be written as

$$F = u\frac{n}{r}T(\omega) = u\alpha_n T(\alpha_n v).$$

Typical values of α_n for gears 1 through 5 are $\alpha_1 = 40$, $\alpha_2 = 25$, $\alpha_3 = 16$, $\alpha_4 = 12$ and $\alpha_5 = 10$. A simple representation of the torque curve is

$$T(\omega) = uT_m\left(1 - \beta(\frac{\omega}{\omega_m} - 1)^2\right), \tag{2}$$

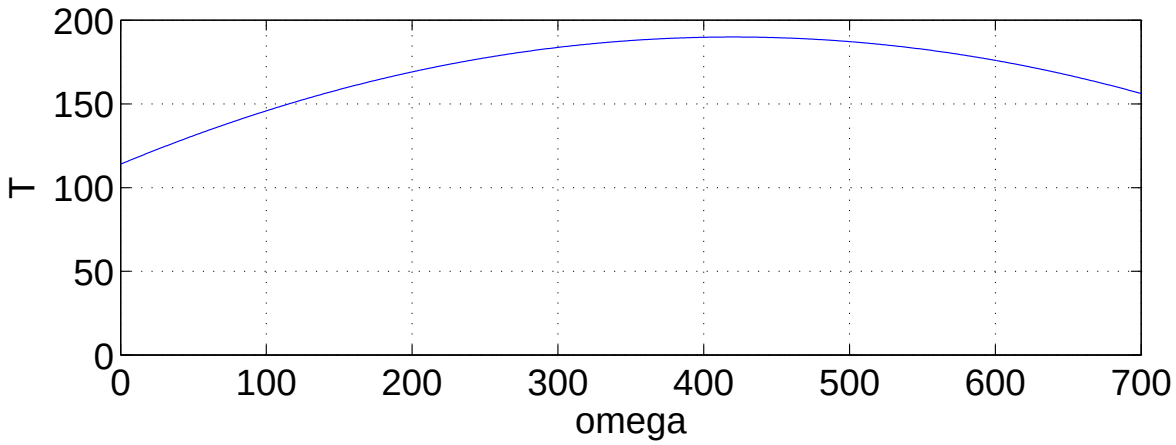

Figure 2: Torque curve for a typical car engine. The curves engine torque as a function of engine speed ω.

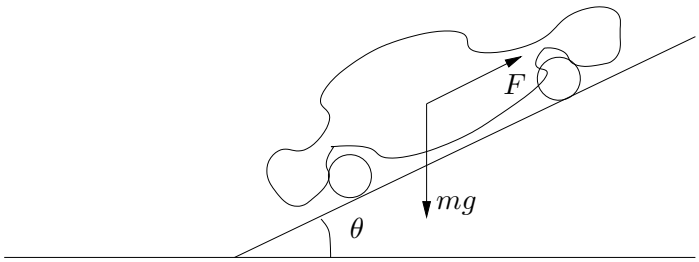

Figure 3: Schematic diagram of a car on a sloping road

where the maximum torque T_m is obtained for ω_m, and $0 \leq u \leq 1$ is the control signal. The curve in Figure 2 has parameters $T_m = 190$ Nm, $\omega_m = 420$ rad/sec (about 4000 rpm) and $\beta = 0.4$

The disturbance force F_d has three major components: F_g, F_r and F_a, due to gravity, rolling friction and aerodynamic drag, respectively. Letting the slope of the road be θ, gravity gives the retarding force $F_g = mg \sin\theta \approx mg\theta$, as illustrated in Figure 3. A simple modeling of rolling friction is

$$F_r = mgC_r$$

where C_r is the coefficient of rolling friction; a typical value is $C_r = 0.01$. Finally, the aerodynamic drag is proportional to the square of the speed:

$$F_a = \frac{1}{2}\rho C_v A v^2.$$

Typical parameters are $\rho = 1.3$, $C_v = 0.32$ and $A = 2.4$ m^2.

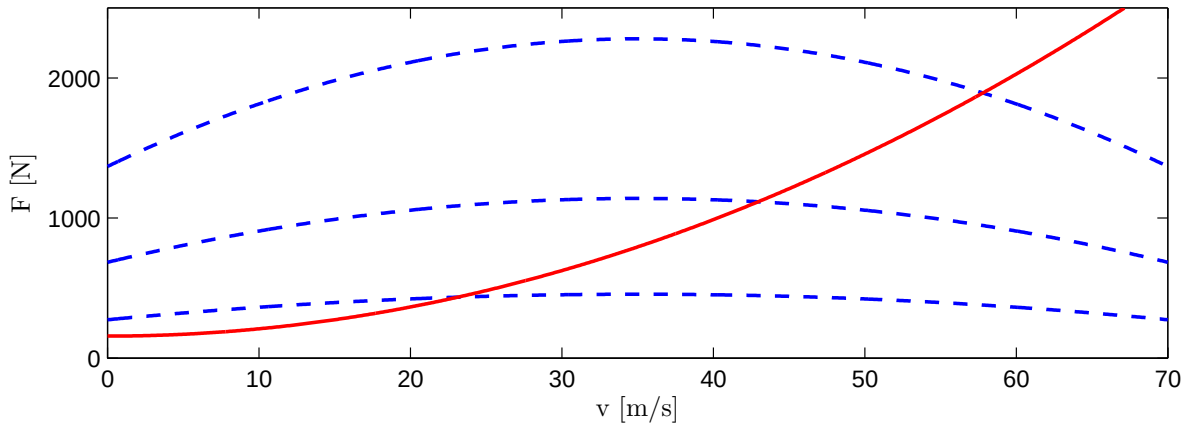

Figure 4: Driving force F (dashed) and disturbance force F_d (full) as functions of velocity. The car is in fourth gear ($\alpha_n = 12$), on a horizontal road ($\theta = 0$). The throttle is $u = 0.2$, 0.5 and 1.

Summarizing, we find that the car can be modeled by

$$\begin{aligned} m\dot{v} &= F - F_d \\ F &= \alpha_n u T(\alpha_n v) \\ F_d &= mgC_r + \frac{1}{2}\rho C_v A v^2 + mg\theta, \end{aligned} \tag{3}$$

where the function T is given by equation (2)

The model (3) is a dynamical system of first order. The state is the car velocity v, which is also the output. The input is the signal u that controls the throttle position, and the disturbance is the force F_d which depends on the slope of the road. The system is nonlinear because of the torque curve and the nonlinear character of the aerodynamic drag. There are also variations in the parameters, e.g. the mass of the car depends on the number of passengers and the load.

Linearization It follows from Equation (3) that there is an equilibrium when the force F applied by the engine balances the disturbance force F_d due to hills, aerodynamic drag and damping. Figure 4 shows the forces F and F_d when driving on a horizontal road. The figure shows that the force F_d (the solid curve) increases rapidly with increasing velocity, mainly due to the aerodynamic drag. Also notice that the driving force has a maximum due to the shape of the torque curve.

To explore the behavior of the system we will consider the equilibrium when the speed is constant at $v_0 = 25$ and the road is horizontal, $\theta = 0$. It follows from (3) that the throttle position is given by

$$u_0 \alpha_n T(\alpha_n v_0) - mgC_r - \frac{1}{2}\rho C_v A v_0^2 - mg\theta_0 = 0. \tag{4}$$

A Taylor series expansion of Equation (3) around the equilibrium gives

$$m\frac{d(v-v_0)}{dt} = ma(v-v_0) - mg(\theta-\theta_0) + mb(u-u_0) \tag{5}$$

where

$$\begin{aligned} ma &= u_0\alpha_n^2 T'(\alpha_n v_0) - \rho C_v A v_0 \\ mb &= \alpha_n T(\alpha_n v_0), \end{aligned} \tag{6}$$

and terms of second and higher order have been neglected. Introducing $v_0 = 25$, $\theta_0 = 0$ and numerical values for the car from Example **??** the model becomes

$$\frac{d(v-v_0)}{dt} = -0.004(v-v_0) + 3.61(u-u_0) - 9.81\theta \tag{7}$$

where $v_0 = 25$ and $u_0 = 0.218$. This linear model describes how small perturbations in the velocity about the nominal speed evolve in time.

The linearized dynamics of the process around an equilibrium point v_0, u_0 are given by

$$\begin{aligned} \dot{\tilde{v}} &= a\tilde{v} - mg\theta + mb\tilde{u} \\ y &= v = \tilde{v} + v_0 \end{aligned}$$

where $\tilde{v} = v - v_0$, $\tilde{u} = u - u_0$, m is the mass of the car and θ is the angle of the road. The constant a depends on the throttle characteristic and is given in Example **??**.

Control Design If we augment the system with an integrator, the process dynamics become

$$\begin{aligned} \dot{\tilde{v}} &= \frac{a}{m}\tilde{v} - g\theta + b\tilde{u} \\ \dot{z} &= r - y = (r - v_0) - \tilde{v}, \end{aligned}$$

or, in state space form,

$$\frac{d}{dt}\begin{bmatrix}\tilde{v}\\ z\end{bmatrix} = \begin{bmatrix}\frac{a}{m} & 0\\ -1 & 0\end{bmatrix}\begin{bmatrix}\tilde{v}\\ z\end{bmatrix} + \begin{bmatrix}b\\ 0\end{bmatrix}u + \begin{bmatrix}-g\\ 0\end{bmatrix}\theta + \begin{bmatrix}0\\ r - v_0\end{bmatrix}$$

Note that when the system is at equilibrium we have that $\dot{z} = 0$ which implies that the vehicle speed, $v = v_0 + \tilde{v}$, should be equal to the desired reference speed, r. Our controller will be of the form

$$\begin{aligned} \dot{z} &= r - y \\ u &= -K\tilde{v} - K_i z + u_d \end{aligned}$$

and the gains K, K_i and nominal input u_d will be chosen to stabilize the system and provide the correct input for the reference speed.

Assume that we wish to design the closed loop system to have characteristic polynomial

$$\lambda(s) = s^2 + a_1 s + a_2.$$

Setting the disturbance $\theta = 0$, the characteristic polynomial of the closed loop system is given by

$$\det\big(sI - (A - BK)\big) = s^2 + (bK - a/m)s - bK_z$$

and hence we set

$$K = (a_1 + \frac{a}{m})/b \qquad K_z = -a_2/b.$$

The resulting controller stabilizes the system and hence brings $\dot{z} = y - r$ to zero, resulting in perfect tracking. Notice that even if we have a small error in the values of the parameters defining the system, as long as the closed loop poles are still stable then the tracking error will approach zero. Thus the exact calibration required in our previous approach (using K_r) is not required. Indeed, we can even choose $K_r = 0$ and let the feedback controller do all of the work (Exercise **??**).

Integral feedback can also be used to compensate for constant disturbances. Suppose that we choose $\theta \neq 0$, corresponding to climbing a (linearized) hill. The stability of the system is not affected by this external disturbance and so we once again see that the car's velocity converges to the reference speed.

References

[1] K. J. Åström and R. M. Murray. *Analysis and Design of Feedback Systems*. Preprint, 2005. Available at `http://www.cds.caltech.edu/~murray/am05`.

[2] B. Friedland. *Control System Design: An Introduction to State Space Methods*. Dover, 2004.

[3] F. L. Lewis and V. L. Syrmos. *Optimal Control*. Wiley, second edition, 1995.

A MATLAB example: Caltech ducted fan

```
% L12_2dfan.m - ducted fan example for L12.2
% RMM, 14 Jan 03

%%
%% Ducted fan dynamics
%%
%% These are the dynamics for the ducted fan, written in state space
%% form.
%%

% System parameters
```

```
J = 0.0475; % inertia around pitch axis
m = 1.5; % mass of fan
r = 0.25; % distance to flaps
g = 10; % gravitational constant
gamma = 0.51; % counterweighted mass
d = 0.2;   % damping factor (estimated)
l = 0.05; % offset of center of mass

% System matrices (entire plant: 2 input, 2 output)
A = [  0     0     0     1     0     0;
       0     0     0     0     1     0;
       0     0     0     0     0     1;
       0     0  -gamma -d/m    0     0;
       0     0     0     0   -d/m    0;
       0     0 -m*g*l/J  0     0     0 ];

B = [  0     0;
       0     0;
       0     0;
      1/m    0;
       0    1/m;
      r/J    0    ];

C = [  1     0    0      0     0     0;
       0     1    0      0     0     0 ];

D = [  0     0;   0      0];

%%
%% Construct inputs and outputs corresponding to steps in xy position
%%
%% The vectors xd and yd correspond to the states that are the desired
%% equilibrium states for the system.  The matrices Cx and Cy are the
%% corresponding outputs.
%%
%% The way these vectors are used is to compute the closed loop system
%% dynamics as
%%
%% xdot = Ax + B u =>xdot = (A-BK)x + K xd
%%         u = -K(x - xd)    y = Cx
%%
%% The closed loop dynamics can be simulated using the "step" command,
%% with K*xd as the input vector (assumes that the "input" is unit size,
%% so that xd corresponds to the desired steady state.
%%
```

```
xd = [1; 0; 0; 0; 0; 0];  Cx = [1 0 0 0 0 0];
yd = [0; 1; 0; 0; 0; 0];  Cy = [0 1 0 0 0 0];

%%
%% LQR design
%%

% Start with a diagonal weighting
Q1 = diag([1, 1, 1, 1, 1, 1]);
R1a = 0.1 * diag([1, 1]);
K1a = lqr(A, B, Q1, R1a);

% Close the loop: xdot = Ax + B K (x-xd)
H1ax = ss(A-B*K1a,B(:,1)*K1a(1,:)*xd,Cx,0);
H1ay = ss(A-B*K1a,B(:,2)*K1a(2,:)*yd,Cy,0);
figure(1); step(H1ax, H1ay, 10);
legend('x', 'y');

% Look at different input weightings
R1b = diag([1, 1]); K1b = lqr(A, B, Q1, R1b);
H1bx = ss(A-B*K1b,B(:,1)*K1b(1,:)*xd,Cx,0);

R1c = diag([10, 10]); K1c = lqr(A, B, Q1, R1c);
H1cx = ss(A-B*K1c,B(:,1)*K1c(1,:)*xd,Cx,0);

figure(2); step(H1ax, H1bx, H1cx, 10);
legend('rho = 0.1', 'rho = 1', 'rho = 10');

% Output weighting
Q2 = [Cx; Cy]' * [Cx; Cy];
R2 = 0.1 * diag([1, 1]);
K2 = lqr(A, B, Q2, R2);

H2x = ss(A-B*K2,B(:,1)*K2(1,:)*xd,Cx,0);
H2y = ss(A-B*K2,B(:,2)*K2(2,:)*yd,Cy,0);
figure(3); step(H2x, H2y, 10);
legend('x', 'y');

%%
%% Physically motivated weighting
%%
%% Shoot for 1 cm error in x, 10 cm error in y.  Try to keep the angle
%% less than 5 degrees in making the adjustments.  Penalize side forces
%% due to loss in efficiency.
```

```
%%

Q3 = diag([100, 10, 2*pi/5, 0, 0, 0]);
R3 = 0.1 * diag([1, 10]);
K3 = lqr(A, B, Q3, R3);

H3x = ss(A-B*K3,B(:,1)*K3(1,:)*xd,Cx,0);
H3y = ss(A-B*K3,B(:,2)*K3(2,:)*yd,Cy,0);
figure(4); step(H3x, H3y, 10);
legend('x', 'y');

%%
%% Velocity control
%%
%% In this example, we modify the system so that we control the
%% velocity of the system in the x direction.  We ignore the
%% dynamics in the vertical (y) direction.  These dynamics demonstrate
%% the role of the feedforward system since the equilibrium point
%% corresponding to vd neq 0 requires a nonzero input.
%%
%% For this example, we use a control law u = -K(x-xd) + ud and convert
%% this to the form u = -K x + N r, where r is the reference input and
%% N is computed as described in class.
%%

% Extract system dynamics: theta, xdot, thdot
Av = A([3 4 6], [3 4 6]);
Bv = B([3 4 6], 1);
Cv = [0 1 0]; % choose vx as output
Dv = 0;

% Design the feedback term using LQR
Qv = diag([2*pi/5, 10, 0]);
Rv = 0.1;
Kv = lqr(Av, Bv, Qv, Rv);

% Design the feedforward term by solve for eq pt in terms of reference r
T = [Av Bv; Cv Dv]; % system matrix
Nxu = T \ [0; 0; 0; 1]; % compute [Nx; Nu]
Nx = Nxu(1:3); Nu = Nxu(4); % extract Nx and Nu
N = Nu + Kv*Nx; % compute feedforward term

%%
%% Design #1: no feedforward input, ud
%%
```

```
Nv1 = [0; 1; 0];
Hv1 = ss(Av-Bv*Kv, Bv*Kv*Nx, Cv, 0);
step(Hv1, 10);

%%
%% Design #2: compute feedforward gain corresponding to equilibrium point
%%

Hv2 = ss(Av-Bv*Kv, Bv*N, Cv, 0);
step(Hv2, 10);

%%
%% Design #3: integral action
%%
%% Add a new state to the system that is given by xidot = v - vd.  We
%% construct the control law by computing an LQR gain for the augmented
%% system.
%%

Ai = [Av, [0; 0; 0]; [Cv, 0]];
Bi = [Bv; 0];
Ci = [Cv, 0];
Di = Dv;

% Design the feedback term, including weight on integrator error
Qi = diag([2*pi/5, 10, 0, 10]);
Ri = 0.1;
Ki = lqr(Ai, Bi, Qi, Ri);

% Desired state (augmented)
xid = [0; 1; 0; 0];

% Construct the closed loop system (including integrator)
Hi = ss(Ai-Bi*Ki,Bi*Ki*xid - [0; 0; 0; Ci*xid],Ci,0);
step(Hi, 10);
```