and model uncertainty may destroy the robustness of LQG and result in instability [209]. The lack of robustness of LQG regulators to model uncertainty motivates the introduction of robust control in Section 8.8. For example, it is possible to *robustify* LQG regulators through a process known as loop-transfer recovery. However, despite robustness issues, LQG control is extremely effective for many systems, and is among the most common control paradigms.

In contrast to classical control approaches, such as proportional–integral–derivative (PID) control and designing faster inner-loop control and slow outer-loop control assuming a separation of timescales, LQG is able to handle multiple-input, multiple-output (MIMO) systems with overlapping timescales and multi-objective cost functions with no additional complexity in the algorithm or implementation.

8.7 Case Study: Inverted Pendulum on a Cart

To consolidate the concepts of optimal control, we will implement a stabilizing controller for an inverted pendulum on a cart, shown in Fig. 8.12. The full nonlinear dynamics are given by

$$\dot{x} = v, \tag{8.67a}$$

$$\dot{v} = \frac{-m^2L^2g\cos(\theta)\sin(\theta) + mL^2(mL\omega^2\sin(\theta) - \delta v) + mL^2u}{mL^2(M + m(1-\cos(\theta)^2))}, \tag{8.67b}$$

$$\dot{\theta} = \omega, \tag{8.67c}$$

$$\dot{\omega} = \frac{(m+M)mgL\sin(\theta) - mL\cos(\theta)(mL\omega^2\sin(\theta) - \delta v) - mL\cos(\theta)u}{mL^2(M + m(1-\cos(\theta)^2))}, \tag{8.67d}$$

where x is the cart position, v is the velocity, θ is the pendulum angle, ω is the angular velocity, m is the pendulum mass, M is the cart mass, L is the pendulum arm, g is the gravitational acceleration, δ is a friction damping on the cart, and u is a control force applied to the cart.

The function **pendcart**, defined in Code 8.2, may be used to simulate the full nonlinear system in (8.67).

Code 8.2: [MATLAB] Right-hand side function for inverted pendulum on cart.

```
function dx = pendcart(x,m,M,L,g,d,u)

Sx = sin(x(3));
Cx = cos(x(3));
D = m*L*L*(M+m*(1-Cx^2));

dx(1,1) = x(2);
dx(2,1) = (1/D)*(-m^2*L^2*g*Cx*Sx + m*L^2*(m*L*x(4)^2*Sx - d
   *x(2))) + m*L*L*(1/D)*u;
```

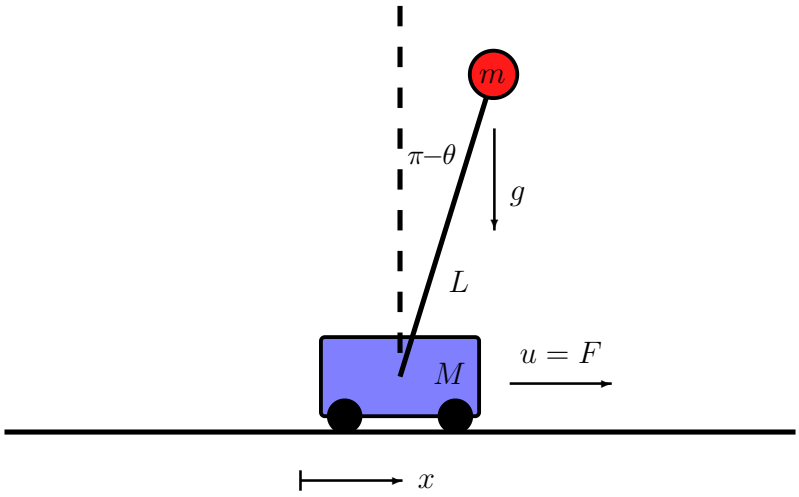

Figure 8.12: Schematic of inverted pendulum on a cart. The control forcing acts to accelerate or decelerate the cart. For this example, we assume the following parameter values: pendulum mass ($m = 1$), cart mass ($M = 5$), pendulum length ($L = 2$), gravitational acceleration ($g = -10$), and cart damping ($\delta = 1$).

```
dx(3,1) = x(4);
dx(4,1) = (1/D)*((m+M)*m*g*L*Sx - m*L*Cx*(m*L*x(4)^2*Sx - d*
    x(2))) - m*L*Cx*(1/D)*u;
```

Code 8.2: [Python] Right-hand side function for inverted pendulum on cart.

```
def pendcart(x,t,m,M,L,g,d,uf):
    u = uf(x) # evaluate anonymous function at x
    Sx = np.sin(x[2])
    Cx = np.cos(x[2])
    D = m*L*L*(M+m*(1-Cx**2))

    dx = np.zeros(4)
    dx[0] = x[1]
    dx[1] = (1/D)*(-(m**2)*(L**2)*g*Cx*Sx + m*(L**2)*(m*L*(x
        [3]**2)*Sx - d*x[1])) + m*L*L*(1/D)*u
    dx[2] = x[3]
    dx[3] = (1/D)*((m+M)*m*g*L*Sx - m*L*Cx*(m*L*(x[3]**2)*Sx
        - d*x[1])) - m*L*Cx*(1/D)*u;

    return dx
```