Dynamics

This chapter discusses the numerical computation of solu8tions of ordinary differential equations ma‘odeling dynamics systems. Differential equations are equations involving derivatives that wre used to determine a function, which is the solution. Here, we discuss *ordinary* differential equations, equations involving derivatives with respedt to a single variable, which we call t. In general discussions, we seek a function, $x(t)$, and the differential equation is written in the form

$$\dot{x} = \frac{dx}{dt} = f(x) \; . \tag{1}$$

Here, both x and f have n components: $x(t) = (x_1(t), \cdots, x_n(t)) \in R^n$, and $f(t) = (f_1(t), \cdots, f_n(t)) \in R^n$. The components of $x(t)$ determine the state of the system at time t, and $f(x)$ represents the dynamics. In most real problems, f will depend on a number of problem parameters also.

For example, here is a model of a simple oscillator (such as a mass connected to a spring sliding on a table) subject to friction. The friction coefficient, γ also might change with time if friction causes the mass or the table to heat up. This is modeled through the equations

$$\gamma(t) = \gamma_0 + \gamma_1 \theta(t) \; , \tag{2}$$

where $\theta(t)$ is the temperature (say, in degrees centigrade) at time t and γ_0 and γ_1 are fixed coefficients. When $\gamma_1 > 0$, the friction coefficient increases when the temperature increases. An oscillator with mass m, spring constant k, position $q(t)$, and momentum $p(t)$ is governed by the equations

$$\begin{aligned} m\dot{q} &= p \; , & (3) \\ \dot{p} &= -kq - \gamma\dot{q} \; , & (4) \end{aligned}$$

The $-\gamma\dot{q}$ term in the p equation represents the force of friction. The rate of work done by this friction force is force $\times \; \dot{q} = \gamma\dot{q}^2$. We suppose the rate of heating is proportional to this rate of work and that heat dissipates with rate r, bringing the temperature back to the ambient temperature, θ_0. With this, the temperature evolution equation is

$$\dot{\theta} = -r(\theta - \theta_0) + \mu\gamma\dot{q}^2 \; . \tag{5}$$

We want to put this formulation into the generic form (1). The state of the system is determined by the three variables q, p, and θ. Although the friction coefficient, $\gamma(t)$, is time dependent, it is determined by the other variables. For this reason, $x(t) = (q(t), p(t), \theta(t))$. We identify f by writing explicit formulas for the components of x in terms of x and parameters. We find $f_1(x) = \dot{x}_1$ by solving (3) for $\dot{q}$:

$$\dot{q} = \frac{1}{m} p \; . \tag{6}$$

We find $f_2 = \dot{p}$ in terms of x, (i.e., in terms of q, p, and θ) by using (6) to put p instead of $\dot{q}$ on the right, and by using (2) to eliminate γ in favor of components of x. This gives

$$\dot{p} = -kq - \frac{\gamma_0 + \gamma_1\theta}{m} p \, . \tag{7}$$

In the same way we can eliminate $\dot{q}$ and γ from the right side of (**??**) to get

$$\dot{\theta} = -r(theta - \theta_0) + \frac{\mu(\gamma_0 - \gamma_1\theta)}{m^2} p^2 \, . \tag{8}$$

This puts the equations (3), (2), (4), (5) in the form (1), with

$$f(x) = \begin{pmatrix} \frac{1}{m}p \\ -kq - \frac{\gamma_0+\gamma_1\theta}{m}p \\ -r(\theta - \theta_0) + \frac{\mu(\gamma_0-\gamma_1\theta)}{m^2}p^2 \end{pmatrix} = \begin{pmatrix} \frac{1}{m}x_2 \\ -kx_1 - \frac{\gamma_0+\gamma_1\theta}{m}x_2 \\ -r(x_3 - \theta_0) + \frac{\mu(\gamma_0-\gamma_1 x_3)}{m^2}x_2^2 \end{pmatrix} \, .$$

Note that f depends on x and also on a large number of time independent parameters, m, k, γ_0, γ_1, r, and μ. This is typical.

We write the dynamics in the form (**??**) for the purpose of general mathematical discussion. In software it is fine to have something closer to the original formulation, such as:

```
int fEval( double x[3], double f[3]) { // Evaluate f for given x

   . . . (parameter definitions)

   double qDot, pDot, gamma, thetaDot;
   double q, p, theta;
   q     = x[0];  // Vector components start with 0 in C, as always.
   p     = x[1];
   theta = x[2];

   qdot     = (1/m)*p;
   gamma    = gamma0 + gamma1*theta;
   pDot     = -k*q - gamma*qdot ;
   thetaDot = -r*( theta - theta0 ) + mu*gamma*qDotqDot;

   f[0] = qDot;
   f[1] = pDot;
   f[2] = thetaDot;

   return 0; // Nothing can possibly go wrong.
```

The advantage of this point of view is that we can write a general program that solves any problem of the form (1).

We focus on the *initial value problem*. For this, in addition to the dynamics (1), we also specify *initial conditions*, $x(t_0) = x_0$. It is traditional to write x_0

for the initial conditions without intending it to mean the zeroth component of x. In the initial value problem, $x(t)$ represents the state of the system at time t. We presume to know the state at some specific time, t_0, and seek to compute the state at later times, $t > t_0$. The task, then, is to find $x(t)$ for $t > t_0$ given $x(t_0)$.

Because of Newton's law: $F = ma$, it is common in physical applications for problems to be formulated using second order differential equations. Suppose we have l particle coordinates $q = (q_1, \ldots, q_l)$ and corresponding force components $F = (F_1(q), \ldots, F_l(q))$. Then Newton's law is

$$m_j \ddot{q}_j = F_j(q) \ .$$

We put this into the standard format (1) using the extra variables $p_j = m_j q_j$ (not the only possible choice). Then we have

$$\begin{aligned} \dot{q}_j &= \frac{1}{m_j} p_j \ , \\ \dot{p}_j &= F_j(q) \ . \end{aligned}$$

This gives $n = 2l$, $x = (q_1, \ldots, q_l, p_1, \ldots, p_l)$, and $f_j = \dot{q}_j = \frac{1}{m_j} p_j$, and $f_j + l = \dot{p}_j = F_j(q)$, for $j = 1, \ldots, l$. In this process, we have replaced a system of l second order equations with an equivalent system of $n = 2l$ first order equations.

We could reduce higher order differential equations as well. Consider, for example, the single third order equation $\frac{d^3 q}{dt^3} = q^2$. For this, take $x_1 = q$, $x_2 = \dot{q}$, and $x_2 = \ddot{q}$. Then we have $f_1 = x_2$, $f_2 = x_3$, and $f_3 = x_1^2$. A single third order equation has become a system of three first order equations.