

Chapter 2

Bacterial Growth in Chemostat

2.1 What Is a Chemostat

A chemostat, or bioreactor, is a continuous stirred-tank reactor (CSTR) used for continuous production of microbial biomass. It consists of a fresh water and nutrient reservoir connected to a growth chamber (or reactor), with microorganism. The mixture of fresh water and nutrient is pumped continuously from the reservoir to the reactor chamber, providing feed to the microorganism, and the mixture of culture and fluid in the growth chamber is continuously pumped out and collected. The medium culture is continuously stirred. Stirring ensures that the contents of the chamber is well mixed so that the culture production is uniform and steady. If the stirring speed is too high, it would damage the cells in culture, but if it is too low it could prevent the reactor from reaching a steady state operation. Figure 2.1 is a conceptual diagram of a chemostat.

Chemostats are used to grow, harvest, and maintain desired cells in a controlled manner. The cells grow and replicate in the presence of suitable environment with medium supplying the essential nutrient growth. Cells grown in this manner are collected and used for many different applications.

These applications include:

1. **Pharmaceutical:** for example in analyzing how bacteria respond to different antibiotics, or in production of insulin (by the bacteria) for diabetics.
2. **Food industry:** for production of fermented food such as cheese.
3. **Manufacturing:** for fermenting sugar to produce ethanol.

A question which arises in operating the chemostat is how to adjust the effluent rate, that is, the rate of pumping out the mixture. In order to operate the chemostat

Electronic supplementary material The online version of this chapter (doi: 10.1007/978-3-319-29638-8_2) contains supplementary material, which is available to authorized users.

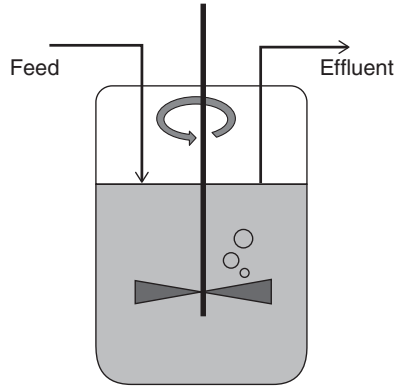


Fig. 2.1: Stirred bioreactor operated as a chemostat, with a continuous inflow (the feed) and outflow (the effluent). The rate of medium flow is controlled to keep the culture volume constant.

efficiently, the effluent rate should not be too small. But if this rate is too large, then the bacteria in the growth chamber may wash out. In order to determine the optimal rate of pumping out the mixture we need to use mathematics. In this chapter, we develop a simple mathematical model in order to determine the optimal effluent rate. A more comprehensive model will be developed in Chapter 8.

The mathematical model will be described by a differential equation. In this book we shall encounter many differential equations that model biological processes. We therefore review here some of the basic theory of differential equations.

2.2 Differential Equations

Differential equations of the **first order** have the form

$$\frac{dx}{dt} = f(x, t), \quad (2.1)$$

where $f(x, t)$ is a given function. Solving this equation means that we have to find a function $x(t)$ which satisfies

$$\frac{dx(t)}{dt} = f(x(t), t).$$

There are in fact many such solutions; but the solution will be unique if we prescribe a condition such as

$$x(t_0) = x_0 \quad (2.2)$$

for some values x_0 and t_0 . The system (2.1)–(2.2) is called an **initial value problem**; the initial time t_0 may be taken, for instance, to be $t_0 = 0$.

Example 2.1. The solution of the differential equation

$$\frac{dx}{dt} = t^2$$

is $x(t) = \frac{t^3}{3} + C$, where C is an arbitrary constant. If we prescribe initial condition $x(0) = 5$, then $C = 5$ and the unique solution is $x(t) = \frac{t^3}{3} + 5$.

Example 2.2. The solution of the initial value problem

$$\frac{dx}{dt} = x + 2, \quad x(0) = 3$$

is given by

$$x(t) = 5e^t - 2.$$

There are several classes of differential equations that can be solved explicitly, and they are introduced in the following subsections.

2.2.1 Linear Equations

Linear differential equations have the following form:

$$\frac{dx}{dt} + p(t)x = g(t), \quad (2.3)$$

where $p(t)$ and $g(t)$ are given functions of t . In order to solve such an equation we introduce the integral of $p(t)$,

$$P(t) = \int_0^t p(s)ds,$$

and multiply Eq. (2.3) by $e^{P(t)}$,

$$e^{P(t)} \frac{dx}{dt} + e^{P(t)} p(t)x(t) = e^{P(t)} g(t).$$

Note that

$$\frac{d}{dt}[e^{P(t)}x(t)] = e^{P(t)} \frac{dx}{dt} + e^{P(t)} p(t)x(t)$$

by the definition of $P(t)$. Therefore,

$$\frac{d}{dt}[e^{P(t)}x(t)] = e^{P(t)} g(t),$$

and by integrating both sides from 0 to t , we get

$$e^{P(t)}x(t) - x(0) = \int_0^t e^{P(s)} g(s)ds.$$

It follows that

$$x(t) = e^{-P(t)}x(0) + e^{-P(t)} \int_0^t e^{P(s)}g(s)ds \quad (2.4)$$

is the solution of (2.3) with prescribed $x(0)$.

Note that in Example 2.2 above $p(t) = -1$, $P(t) = -t$, $g(t) = 2$,

$$e^{-P(t)} \int_0^t e^{P(s)}g(s)ds = 2e^t \int_0^t e^{-s}ds = 2e^t(1 - e^{-t}),$$

and formula (2.4) yields

$$x(t) = e^t x(0) + 2(e^t - 1) = 5e^t - 2 \quad \text{if } x(0) = 3.$$

2.2.2 Separation of Variables

Differential equations with separable variables are of the form

$$\frac{dx}{dt} = g(x)h(t). \quad (2.5)$$

Rewriting this equation in the form

$$\frac{1}{g(x)} \frac{dx}{dt} = h(t)$$

we get, by integration with respect to t ,

$$\int \frac{dx}{g(x)} = \int h(t)dt,$$

from which we obtain the solution

$$G(x) = H(t) + C$$

where

$$G(x) = \int \frac{dx}{g(x)}, \quad H(t) = \int h(t)dt.$$

Example 2.3. Consider the equation

$$\frac{dx}{dt} = \frac{t}{x^2}.$$

Writing it in the form

$$x^2 \frac{dx}{dt} = t$$

we get, by integration,

$$\frac{x^3}{3} = \frac{t^2}{2} + C.$$

2.2.3 Homogeneous Equations

Differential equations that can be written in the form

$$\frac{dx}{dt} = g\left(\frac{x}{t}\right) \quad (2.6)$$

are called homogeneous equations. Such equations can be solved by introducing a new variable $v = \frac{x}{t}$, or $v(t) = \frac{x(t)}{t}$. Then

$$\frac{dx}{dt} = \frac{d}{dt}(tv) = v + t \frac{dv}{dt},$$

and Eq. (2.6) becomes

$$t \frac{dv}{dt} + v = g(v),$$

which is an equation with separable variables, namely,

$$\frac{dv}{dt} = \frac{g(v) - v}{t}.$$

Hence

$$\int \frac{dv}{g(v) - v} = \ln t + C.$$

If we denote the integral of $1/(g(v) - v)$ by $K(v)$, then the solution of Eq. (2.6) is given implicitly by the formula

$$K\left(\frac{x}{t}\right) = \ln t + C.$$

Example 2.4. The equation

$$\frac{dx}{dt} = \frac{x^2 + t^2}{xt}$$

can be written in the form

$$\frac{dx}{dt} = g\left(\frac{x}{t}\right), \text{ where } g(v) = v + \frac{1}{v}.$$

Setting $v = \frac{x}{t}$, we get

$$\int \frac{dv}{g(v) - v} = \ln t + C,$$

and the integral of the left-hand side is $\int v dv = \frac{v^2}{2}$. Hence

$$\frac{1}{2}\left(\frac{x}{t}\right)^2 = \ln t + C,$$

or

$$x^2 = 2t^2(\ln t + C).$$

2.2.4 Exact Equations

Consider a differential equation of the form

$$g(x,t)\frac{dx}{dt} + h(x,t) = 0. \quad (2.7)$$

If there is a function $F(x,t)$ such that

$$\frac{\partial F}{\partial x} = g, \quad \frac{\partial F}{\partial t} = h, \quad (2.8)$$

then Eq. (2.7) is called an **exact equation**, and it can be written in the form

$$\frac{\partial F}{\partial x} \frac{dx}{dt} + \frac{\partial F}{\partial t} = 0$$

or

$$\frac{dF(x(t),t)}{dt} = 0.$$

Hence,

$$F(x(t),t) = \text{constant},$$

and the solution of Eq. (2.7) is given implicitly by the equation

$$F(x,t) = c, \quad c \text{ is constant.}$$

We note that if there is a function F such that (2.8) holds, then

$$\frac{\partial g}{\partial t} = \frac{\partial^2 F}{\partial t \partial x} = \frac{\partial^2 F}{\partial x \partial t} = \frac{\partial h}{\partial x}.$$

Conversely, if the functions g and h are such that

$$\frac{\partial g}{\partial t} = \frac{\partial h}{\partial x}$$

then a function F satisfying (2.8) can be constructed by integration.

2.2.5 Integrating Factor

A differential equation can sometimes be made an exact equation by multiplying it by a function, called **integrating factor**. If $\mu = \mu(x,y)$ is to be an integrating factor for Eq. (2.7), then it has to satisfy the equation

$$\frac{\partial}{\partial t}(\mu g) = \frac{\partial}{\partial x}(\mu h),$$

or

$$\mu\left(\frac{\partial g}{\partial t} - \frac{\partial h}{\partial x}\right) + g\frac{\partial \mu}{\partial t} - h\frac{\partial \mu}{\partial x} = 0.$$

If

$$\frac{1}{g}\left(\frac{\partial g}{\partial t} - \frac{\partial h}{\partial x}\right) \text{ is a function } k(t) \text{ (of } t \text{ only),}$$

then we can find an integrating factor $\mu = \mu(t)$ by solving

$$\frac{1}{\mu} \frac{d\mu}{dt} = -k(t).$$

On the other hand if

$$\frac{1}{h}\left(\frac{\partial g}{\partial t} - \frac{\partial h}{\partial x}\right) \text{ is a function } m(x) \text{ (of } x \text{ only),}$$

then we can find an integrating factor $\mu = \mu(x)$ by solving

$$\frac{1}{\mu} \frac{d\mu}{dx} = m(x).$$

Example 2.5. Consider the equation

$$t(t-x)\frac{dx}{dt} + (3xt - x^2) = 0.$$

In this case we have

$$\frac{1}{g}\left(\frac{\partial g}{\partial t} - \frac{\partial h}{\partial x}\right) = \frac{1}{t} \quad \text{and } \mu(t) = t.$$

Multiplying the differential equation by t we obtain an exact differential equation

$$(t^3 - tx)\frac{dx}{dt} + (3xt^2 - \frac{1}{2}x^2) = 0$$

with $F(x, t)$ such that

$$\frac{\partial F}{\partial x} = t^3 - tx, \quad \frac{\partial F}{\partial t} = 3xt^2 - \frac{1}{2}x^2,$$

namely

$$F(x, t) = t^3x - \frac{1}{2}tx^2.$$

Hence the solution of the differential equation is

$$t^3x - \frac{1}{2}tx^2 = c, \quad c \text{ is constant.}$$

We have actually already encountered an integrating factor for the linear equation (2.3), namely, e^P . Indeed, after multiplying both sides of Eq. (2.3) by e^P we obtain an exact equation with

$$F(x, t) = e^{P(t)}x(t) - \int_0^t e^{P(s)}g(s)ds.$$

2.2.6 Existence of Solutions

So far we have shown how to solve explicitly some classes of differential equations. For general functions $f(x, t)$ the initial value problem (2.1)–(2.2) cannot be solved explicitly, but it can always be solved numerically, as will be shown in the numerical sections. The following theorem asserts that the initial value problem (2.1)–(2.2) has a unique solution.

Theorem 2.1. *Let $f(x, t)$ be a continuously differentiable function in a domain which contains a point (x_0, t_0) . Then the initial value problem (2.1)–(2.2) has a unique solution $x = x(t)$ for t in some interval which contains the point $t = t_0$.*

We shall be particularly interested in differential equation (2.1) where f is independent of t , namely,

$$\frac{dx}{dt} = f(x), \quad (2.9)$$

and $f(x)$ is continuously differentiable for all x . In this case Theorem 2.1 can be extended as follows:

Theorem 2.2. *The solution of the initial value problem (2.9), (2.2) exists for all positive t as long as $x(t)$ remains bounded.*

The proof of Theorems 2.1 and 2.2 can be found, for instance, in Reference [1].

Example 2.6. Consider the system

$$\frac{dx}{dt} = x^\alpha, \quad x(0) = 1, \quad (2.10)$$

where $0 < \alpha < \infty$. Rewriting the differential equation in the form

$$x^{-\alpha}dx = dt,$$

and we integrate it (note that the equation has separable variables) and use the initial condition to obtain

$$\frac{x^{1-\alpha}}{1-\alpha} = t + \frac{1}{1-\alpha},$$

or

$$x(t) = [(1-\alpha)t + 1]^{\frac{1}{1-\alpha}}.$$

If $0 < \alpha < 1$ then the solution exists for all $t > 0$ and $x(t) \rightarrow \infty$ as $t \rightarrow \infty$. If, however, $\alpha > 1$ then as t increases to $1/(\alpha - 1)$ the solution $x(t)$ increases to ∞ , so the solution exists only for $t < 1/(\alpha - 1)$.

The solution of (2.9), (2.2) can also be continued to $t < 0$, but again only as long as $x(t)$ remains bounded. One often refers to a solution of (2.9), $x(t)$ for $0 \leq t < \infty$, as a **trajectory**.

2.2.7 Differential Inequalities

We shall encounter in this book differential inequalities of the form

$$\frac{dx}{dt} + \mu x \leq b \text{ for } t > 0, \quad (2.11)$$

or

$$\frac{dx}{dt} + \mu x \geq b \text{ for } t > 0, \quad (2.12)$$

and we shall need to determine the behavior of $x(t)$ as $t \rightarrow \infty$. Consider first the inequality (2.11). Multiplying both sides by $e^{\mu t}$ (note that $e^{\mu t}$ is always positive for real μ) we get

$$\frac{d}{dt}(e^{\mu t} x(t)) \leq b e^{\mu t}$$

so that, by integration from 0 to t ,

$$e^{\mu t} x(t) - x(0) < \frac{b}{\mu} (e^{\mu t} - 1)$$

or

$$x(t) \leq e^{-\mu t} \left(x(0) - \frac{b}{\mu} \right) + \frac{b}{\mu}.$$

We conclude that if $x(0) \leq \frac{b}{\mu}$ then $x(t) \leq \frac{b}{\mu}$ for all $t > 0$. If however $x(0) > \frac{b}{\mu}$ then, for any small $\varepsilon > 0$,

$$x(t) < \frac{b}{\mu} + \varepsilon \quad (2.13)$$

if t is large enough, so that

$$e^{-\mu t} \left(x(0) - \frac{b}{\mu} \right) < \varepsilon.$$

Similarly, from the inequality (2.12) we can deduce that for any small $\varepsilon > 0$

$$x(t) > \frac{b}{\mu} - \varepsilon \quad (2.14)$$

if t is large enough. For later references we state:

Theorem 2.3. *If a function $x(t)$ satisfies the differential inequality (2.11), then, for any small $\varepsilon > 0$, (2.13) holds for all t sufficiently large. Similarly, if a function $x(t)$ satisfies the inequality (2.12), then, for any small $\varepsilon > 0$, (2.14) holds for all t sufficiently large.*

2.3 Equilibrium and Stability

If x_0 is a point such that $f(x_0) = 0$, then the unique solution of (2.9), (2.2) is clearly $x(t) \equiv x_0$. Such a point x_0 is called an **equilibrium point**, a **steady state**, or a **stationary point**. By Taylor's formula,

$$f(x) = f(x_0) + f'(x_0)(x - x_0) + (x - x_0)\varepsilon(x - x_0),$$

where $\varepsilon(x - x_0) \rightarrow 0$ if $x \rightarrow x_0$.

Suppose x_0 is an equilibrium point such that $f'(x_0) < 0$. Setting $y = x - x_0$ and by using Eq. (2.9), we then have

$$\frac{dy}{dt} = f'(x_0)y + y\varepsilon(y),$$

where $\varepsilon(y) \rightarrow 0$ if $y \rightarrow 0$.

If $|y|$ is small enough so that $|\varepsilon(y)| < \frac{1}{2}|f'(x_0)|$, then, for $y > 0$,

$$\frac{dy}{dt} < f'(x_0)y + \frac{1}{2}|f'(x_0)|y = f'(x_0)y - \frac{1}{2}f'(x_0)y = \frac{1}{2}f'(x_0)y,$$

so that

$$\frac{dy}{dt} < 0 \quad \text{if } y > 0$$

and $y = y(t)$ is decreasing toward $y = 0$. Similarly

$$\frac{dy}{dt} > 0 \quad \text{if } y < 0,$$

so that $y = y(t)$ is increasing toward $y = 0$.

Hence when $f'(x_0) < 0$, the solution $x(t)$, starting near x_0 , moves toward x_0 as t increases; in fact, $x(t) \rightarrow x_0$ as $t \rightarrow \infty$. We therefore call x_0 a **stable equilibrium** (or more precisely **asymptotically stable equilibrium**). Similarly, if

$$f'(x_0) > 0$$

then solutions initiating near x_0 move away from x_0 , as long as they are within a small distance from x_0 . We call such a point x_0 an **unstable equilibrium**.

A steady point x_0 is called **globally (asymptotically) stable** if $x(t) \rightarrow x_0$ for any trajectory $x(t)$ whose initial value $x(0)$ is not a steady point.

2.4 Growth Models

We need to develop a mathematical model describing the growth of bacteria population. The density x of bacteria is defined as the number of bacteria per unit volume. If the bacteria grow at a fixed rate r , then

$$x(t + \Delta t) - x(t) = rx(t)\Delta t,$$

or

$$\frac{x(t + \Delta t) - x(t)}{\Delta t} = rx(t),$$

and, taking $\Delta t \rightarrow 0$, we get

$$\frac{dx}{dt} = rx. \quad (2.15)$$

The explicit formula for the growth of x is then

$$x(t) = x(0) e^{rt}.$$

The **doubling time** T is defined by $x(T) = 2x(0)$, that is, the time for the bacteria to double in number, and it is given by

$$2 = e^{rT}, \quad \text{or} \quad T = \frac{\ln 2}{r}.$$

If a colony of bacteria, or other microorganism, is dying at rate s , then its density x satisfies

$$\frac{dx}{dt} = -sx, \quad (2.16)$$

and

$$x(t) = x(0)e^{-st}.$$

The population density is halved at time \bar{T} , called the **half-life**, given by

$$\bar{T} = \frac{\ln 2}{s}.$$

When bacteria are confined within a bounded chamber, they cannot grow exponentially forever, by following (2.15). There is going to be a **carrying capacity** B of the medium which the bacterial density cannot exceed. This situation is modeled by replacing the exponential growth (2.15) by the **logistic growth**

$$\frac{dx}{dt} = rx\left(1 - \frac{x}{B}\right). \quad (2.17)$$

The solution of (2.17) with an initial condition

$$x(0) = x_0$$

is given by

$$x(t) = \frac{B}{1 + (\frac{B}{x_0} - 1)e^{-rt}}. \quad (2.18)$$

Indeed, to derive (2.18), we rewrite (2.17) in the form

$$\frac{dx}{x(1 - \frac{x}{B})} = rdt,$$

or

$$(\frac{1}{x} + \frac{1}{B} \frac{1}{1 - \frac{x}{B}})dx = rdt,$$

and integrate to obtain

$$\ln x - \ln(1 - \frac{x}{B}) = rt + \text{const.}$$

Then

$$\frac{x}{1 - \frac{x}{B}} = Ce^{rt},$$

and solving for x we get

$$x(t) = \frac{Ce^{rt}}{1 + \frac{C}{B}e^{rt}} = \frac{B}{1 + \frac{B}{C}e^{-rt}}. \quad (2.19)$$

Substituting $t = 0$, $x(0) = x_0$, we find the value of C :

$$1 + \frac{B}{C} = \frac{B}{x_0}, \text{ or } C = \frac{x_0}{1 - \frac{x_0}{B}}.$$

Substituting C into Eq.(2.19), we obtain the formula (2.18) for the solution of Eq.(2.17). In the logistic growth equation (2.17) the point $x = B$ is a stable equilibrium. From (2.18) we see that $x = B$ is also a globally asymptotically stable equilibrium, since, for any initial value $x(0) = x_0$, $x(t) \rightarrow B$ as $t \rightarrow \infty$.

2.5 Modeling the Chemostat

Figure 2.2 shows a schematics of a chemostat with a stock of nutrient C_0 pumped into the chamber of the bacterial culture. We assume that the chemostat chamber is well stirred so that the nutrient concentration is constant at each time t . We then model the bacterial growth by the logistic equation (2.17), where r depends on the constant nutrient concentration C_0 . If we denote by s the rate of the bacterial outflow from the chamber, then the balance between growth and outflow is given by

$$\frac{dx}{dt} = rx(1 - \frac{x}{B}) - sx. \quad (2.20)$$

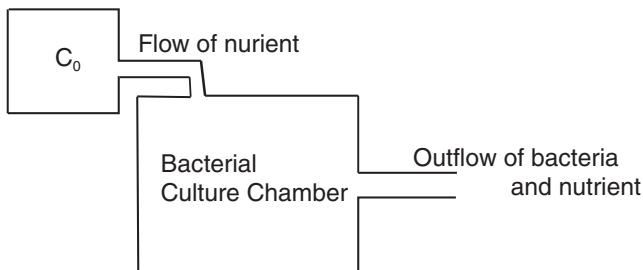


Fig. 2.2: The chemostat device.

We shall denote by $[X]$ the dimension of any quantity X . Then,

$$[x] = \frac{\text{number}}{\text{volume}}, \quad [B] = \frac{\text{number}}{\text{volume}},$$

$$[r] = \frac{1}{\text{time}}, \quad [s] = \frac{1}{\text{time}}.$$

There are two equilibrium points to (2.20), namely, $x = 0$, and $x = (1 - \frac{s}{r})B$. Note that if the outflow rate is less than the growth rate of the bacteria, that is, if $s < r$, then $x = 0$ is an unstable equilibrium, whereas $x = (1 - \frac{s}{r})B$ is a stable equilibrium. If $s > r$, then $x = 0$ is a stable equilibrium, whereas the equilibrium point $x = (1 - \frac{s}{r})B$ is not biologically relevant since it is negative.

Consider the case $s < r$ and $x(0) < (1 - \frac{s}{r})B$. Since $(1 - \frac{s}{r})B$ is a stable equilibrium, if $x(0)$ is near $(1 - \frac{s}{r})B$, it will remain smaller than $(1 - \frac{s}{r})B$ and will converge to it as $t \rightarrow \infty$. We can actually solve $x(t)$ explicitly: writing

$$\frac{1}{rx(1 - \frac{x}{B}) - sx} = \frac{1}{r-s} \left(\frac{1}{x} + \frac{r/B}{(r-s) - rx/B} \right)$$

we have

$$\frac{1}{r-s} \left[\frac{dx}{x} + \frac{r/B}{(r-s) - rx/B} dx \right] = dt.$$

By integration

$$\frac{1}{r-s} [\ln x - \ln((r-s) - rx/B)] = t + \text{const},$$

or

$$\frac{x}{(r-s) - rx/B} = ce^{(r-s)t} \quad (c \text{ is constant}).$$

Hence

$$\left(\frac{1}{c} e^{-(r-s)t} + \frac{r}{B} \right) x = r - s,$$

or

$$x(t) = \frac{r-s}{\frac{r}{B} + \frac{1}{c}e^{-(r-s)t}}. \quad (2.21)$$

We see that $x(t) \rightarrow (1 - \frac{s}{r})B$ as $t \rightarrow \infty$, whenever $x(0) < (1 - \frac{s}{r})B$. Note that the formula (2.21) is valid also when $x(0) > (1 - \frac{s}{r})B$ and that c is determined by

$$x(0) = \frac{r-s}{\frac{r}{B} + \frac{1}{c}}, \quad \text{or} \quad \frac{1}{c} = \frac{r-s}{x(0)} - \frac{r}{B}.$$

The chemostat operator would like to adjust the outflow rate s so as to get the largest output of bacteria. The mathematical model we developed can determine the optimal rate. Indeed, at steady state the outflow rate s is to be multiplied by the steady state of the bacteria, which is $x = (1 - \frac{s}{r})B$. The function $s(1 - \frac{s}{r})B$ takes its maximum at $s = \frac{r}{2}$, and with this output rate the maximum outflow per unit time is $\frac{1}{2}rB$.

Summary. The chemostat operates most efficiently when $s = \frac{r}{2}$, that is, when the outflow rate is half the inflow rate.

Problem 2.1. Find the general solution of the differential equations

- (i) $\frac{dx}{dt} + x = 3e^t$;
- (ii) $\frac{dx}{dt} = -2tx + t$;
- (iii) $t \frac{dx}{dt} + \alpha x = t^2$, $\alpha > 0$.

Problem 2.2. Find the solution of the initial value problems

- (i) $\frac{dx}{dt} - tx = t$, $x(0) = 2$;
- (ii) $\frac{dx}{dt} - 3x = t + 2$, $x(0) = -1$.

Problem 2.3. Find the solution of the initial value problems

- (i) $\frac{dx}{dt} = \frac{t}{x}$, $x(1) = 3$;
- (ii) $\frac{dx}{dt} = \frac{1+x^2}{xt}$, $x(1) = 2$.

Problem 2.4. Solve the equation

$$\frac{dx}{dt} = \frac{x+4t}{x+t} \quad \text{with } x(0) = 3.$$

Problem 2.5. Find the solution of

$$(2xt + \frac{1}{x}) \frac{dx}{dt} = x^2, \quad x(3) = 1.$$

Problem 2.6. Find the solution of

$$(3x^3 + xt^2) \frac{dx}{dt} + 2x^2t = 0, \quad x(2) = 8.$$

Problem 2.7. Consider the equation

$$\frac{dx}{dt} = x(x-a)(x-2), \quad 0 < a < 2.$$

It has three steady points, $x = 0$, $x = 2$, and $x = a$. Determine which of them is stable.

Problem 2.8. Consider the equation

$$\frac{dx}{dt} = (x-a)(2-x), \quad x(0) < a,$$

where $a < 2$. Find the solution explicitly in either the forms $t = t(x)$ or $x = x(t)$, and use it to prove the following:

- (i) If $x(0) > a$ then the solution exists for all $t > 0$ and $x(t) \rightarrow 2$ as $t \rightarrow \infty$;
- (ii) If $x(0) < a$ then the solution exists for $t < T$, where $T = \frac{1}{2-a} \ln \left| \frac{2-x(0)}{a-x(0)} \right|$, and $x(t) \rightarrow -\infty$ as $t \rightarrow T$.

2.6 Numerical Simulations – Introduction to MATLAB

MATLAB is a software developed by The MathWorks, Inc., and it is widely used in science and engineering. MATLAB is a high-level language and interactive environment for numerical computation, symbolic calculation, and visualization. It is also known for its easy handling of matrices and vectors. To access this software, in many universities, students can install licensed MATLAB software (you can request from the IT department in your school), and individual licenses can also be purchased through MathWorks website.

We will refer the readers to MathWorks' website for details of installation and launch of the software. In this chapter, we will introduce some basics of MATLAB and prompt to solving an ODE problem with MATLAB. The codes and explanations about MATLAB are based on the version of MATLAB R2014b.

The introduction here is elementary and not comprehensive, but it will give the readers the basic idea of how MATLAB operates and how to use this software to solve our models. We strongly encourage the readers to practice along when reading through numerical sections in this book.

2.6.1 Scalar Calculations

Once we launch MATLAB, the default window will have several compartments: a panel with function buttons, and main columns "Current folder," "Command Window," and "Workspace." We can change to the directory that we would like to work in, and the corresponding folders and subfolders will show in the "Current

Folder” part. The “Command Window” is for us to enter commands and do some calculations, and the “Workspace” will save the variables that have been used in our calculations.

MATLAB can do basic calculations as in regular calculators. MATLAB recognizes the usual arithmetic operation: + (addition), – (subtraction), * (multiplication), / (division), ^ (power). In the Command Window, we will see the prompt sign (>>), and we can type after prompt sign and press enter, which will give us the result of calculation. In the following, we show the MATLAB commands in teletype font. For example,

```
>> (5*2+3.5) / 5
ans =
2.7000
```

If we do not want to see the display of the answer, we can add a semicolon (;) after the command to suppress the display. We can also store the result into a variable that the user assigns, for example:

```
>> x = (5*2+3.5) / 5
x =
2.7000
```

If now we check the Workspace column, we will see that ‘x’ is stored and the value is also shown in that column. If we did not specify the name of the variable, the result will be stored in ‘ans’ in the Workspace. It is worth noting that a valid variable name starts with a letter, followed by letters, digits, or underscores. MATLAB is case sensitive, so B and b are not the same variable. We should avoid creating variable names that conflict with function names (functions will be introduced later).

MATLAB recognizes different types of numbers: (1) integer (example: 112, –2185); (2) real number (example: 2.452, –100.448); (3) complex (example: $-0.11 + 4.4i$, $i = \sqrt{-1}$); (4) Inf (infinity); (5) NaN (not a number).

All the calculations in MATLAB are done in double precision, which means that the numbers are accurate up to about 15 significant figures. However, we may not see that many digits on the display window, and this is because the default output format is to display 4 decimal places. If you type `format long` in the command window followed by pressing enter, for all the numbers shown in the command window, you will see the full display of all the digits. The command `format short` will switch back to display of 4 decimal places. To know about more format, type `help format`. In general, this `help` command is very useful when we would like to know how to use a command or a function; we simply type `help xx`, in which `xx` is the command of interest.

MATLAB has some built-in trigonometric functions and elementary functions. We choose some commonly used ones to list in Table 2.1.

When we code, it is usually important to make comments in the codes. These comments explain what the commands are for, so that the codes are easier to read later. In MATLAB, we use the percentage sign (%) to begin a comment, and MATLAB will take all the characters after (%) as comments and those will not be executed. For example:

```
>> y = (5*2+3.5)/5^2 % store the result in variable y,
and show the result on the screen.
```


Table 2.1: Commonly used MATLAB built-in functions. One can substitute ‘x’ in the table by numbers or other variables.

MATLAB build-in functions	descriptions
abs(x)	absolute value of x
sqrt(x)	square root of x
sin(x)	sine of x in radians
sind(x)	sine of x in degrees
cos(x)	cosine of x in radians
cosd(x)	cosine of x in degrees
tan(x)	tangent of x in radians
cot(x)	cotangent of x in radians
sec(x)	secant of x in radians
csc(x)	cosecant of x in radians
asin(x)	inverse sine of x in radians
acos(x)	inverse cosine of x in radians
atan(x)	inverse tangent of x in radians
sinh(x)	hyperbolic sine of x in radians
cosh(x)	hyperbolic cosine of x in radians
exp(x)	exponential of x
log(x)	natural logarithm of x
log2(x)	base 2 logarithm of x
log10(x)	base 10 logarithm of x
ceil(x)	round x toward infinity
floor(x)	round x toward minus infinity
round(x)	round x to the nearest integer

If the operation is too long, one can use ‘...’ to extend the command to the next line, for example:

```
>> z = 10*sin(pi/3)*...
>> sin(pi^2/4)
```

A convenient way to record the commands we are typing is to use ‘diary FILE-NAME’, for example:

```
>> diary myfile
>> x = sqrt(5);
>> y = exp(x);
>> diary off
```

In the same directory, if you open the file ‘myfile,’ we will see the records of commands and outputs. We can turn the diary back on by using ‘diary on.’

2.6.2 Vector and Matrix Operations

In previous examples, we have discussed how to use MATLAB to do the usual scalar calculations. In fact, MATLAB is very powerful when it comes to calculations of

vectors and matrices, and it is a vector oriented programming language. For this reason, we should maximize the use of vector and matrix operations in our codes.

In the previous section, variables were used to store scalars. Here we show that they can also be used to store vectors. The following is an example to assign a vector to a variable:

```
>> s = [1 3 5 2]; % note the use of [], and the spaces
between the numbers; one can also use comma (,) to
separate the numbers
>> t = 2*s + 1 % 1 will be added to all the entries of 2*s
t =
3 7 11 5
```

In the above example, MATLAB uses [] to establish a row vector [1 3 5 2] and stores it in the variable *s*, and does an operation on it to make a new row vector [3 7 11 5] and stores it in the variable *t*. To extract one element from the vector or part of the vector to do operations, we type:

```
>> t(3) % display the third entry of vector t
ans =
11
>> t(3) = 2 % assign another value to the third entry of
vector t
t =
3 7 2 5
>> 2*t - 5*s
ans =
1 -1 -21 0
```

As we have learned in linear algebra, in order to add or subtract, two vectors need to have the same length.

```
>> a = [1 2 3]; b = [5 6];
>> a + b
Error using +
Matrix dimensions must agree.
```

The above message means we have inconsistent matrix or vector dimensions, so we need to go back to check the dimensions of our matrices or vectors. Although we cannot add or subtract *a* and *b*, we can combine them to form a new vector, for example,

```
>> cd = [-b, 3*a]
cd =
-5 -6 3 6 9
```

Sometimes, we need vectors whose entries are part of an arithmetic sequence, a convenient way to define it is to use the colon notation:

```
>> 1:2:6 % this will generate a row vector, starting
at 1, ending at 6, with increment 2
ans =
1 3 5
```

```
>> 3:10 % without specifying the increment, it will
be set as 1
```

```
ans =
```

```
3 4 5 6 7 8 9 10
```

With this trick, we can easily extract a part of a vector, and do operations:

```
>> t(2:4) - 1 % this will be the same as typing
```

```
t([2 3 4]) - 1
```

```
ans =
```

```
6 1 4
```

We have learned how to define and use row vectors. The operations for column vectors are similar. The only difference is that the entries of a column vector are separated by semicolon (;) or by making a new line.

```
>> cv = [-1; pi; exp(2)]
```

```
cv =
```

```
1.0000
```

```
3.1416
```

```
7.3891
```

```
>> cv2 = [1
```

```
2
```

```
3]
```

```
cv2 =
```

```
1
```

```
2
```

```
3
```

The row and column vectors can be transposed to become column and row vectors, respectively. The transpose of a vector or matrix is done by putting an apostrophe after the variable name.

```
>> cv', t'
```

```
ans =
```

```
1.0000 3.1416 7.3891
```

```
ans =
```

```
3
```

```
7
```

```
2
```

```
5
```

Similar to creating vectors, an $m \times n$ matrix can be created by adding a semicolon (;) after the end of each row. As in row and column vectors, entries in a row are separated by spaces or commas, while rows are separated by using semicolons or by making a new line. For example:

```
>> A = [1 2 3 4; 5 6 7 8; 9 10 11 12]
```

```
A =
```

```
1 2 3 4
```

```
5 6 7 8
```

```
9 10 11 12
```

We can extract or change any single entry in the matrix

```
>> A(2,3) = 5; % change the (2,3) entry of A to 5
or extract part of the matrix
```

```
>> B = A(2,1:3) % take the second row, the first
to third column, store as a new matrix B
```

```
>> B =
    5    6    7
```

We can combine matrices, as long as the dimensions are consistent.

```
>> A = [A B'] % transpose B, make it as the last column
vector and merge with A
```

```
A =
    1    2    3    4    5
    5    6    7    8    6
    9   10   11   12    7
```

We can extract the whole row or column by using semicolon

```
>> A(:,3) % note here ':' can be replaced by '1:end',
that is, 1 to end
```

```
A =
    3
    7
   11
>> A(1,:)
A =
    1    2    3    4    5
```

Then we can redefine, or delete a row or a column from a matrix A:

```
>> A(:,2) = [] % delete the second column of A
(: represents all the rows, [] is an empty vector)
>> A = [A; 4 3 2 1; 0 -1 -2 -3]; % adding the fourth and
fifth row in the matrix A
```

To find out the dimension of a matrix, we use the command “size.”

```
>> size(A') % the output is [number of rows, number of
columns]
```

```
ans =
    4    5
```

To obtain the length of a vector, we use “length.”

```
>> length(A(1,:))
ans =
    4
```

There are some built-in matrix generating functions,

```
>> ones(2,3) % this generates a 2x3 matrix with ones
>> zeros(4,4) % this generates a 4x4 matrix with zeros
>> eye(5) % this generates a 5x5 identity matrix
>> diag([1 3 5]) % this generates a matrix with 1 3 5 on
its diagonal
```

Next, let us do matrix-matrix or matrix-vector multiplication. When we use `*` in the matrix operations, it will operate as the matrix-matrix multiplication in linear algebra. For example,

```
>> X = [1 2 3; 0 2 4]; Y = [5 2; 1 1; 10 7]; W = X*Y
W =
    37    25
    42    30
```

If we try

```
>> X*X
```

then we will see an error message about the matrix dimension, because an $m \times n$ matrix can only be multiplied by an $n \times k$ matrix. Sometimes we would like to perform component-by-component operations, but not matrix-matrix multiplications; for that purpose we need to use `.*` instead of `*`. The following commands will give different results:

```
>> W.* W % component-by-component operation
>> W * W % matrix-matrix multiplication
```

and we will find that `X.*X` works because it is a component-by-component operation. Note that the use of `.*` requires the two matrices to have the same size. This component-wise operation of matrices can also be used for division (`./`) and exponents (`.^`).

Problem 2.9. Try the following command to generate a vector \mathbf{x} .

```
>> x = 0:0.01:2
```

What is the x you see on MATLAB? Then use the command

```
>> y = sin(x)
```

to generate another vector \mathbf{y} , what is \mathbf{y} ?

Problem 2.10. Let $\mathbf{x} = [2, 5, 1, 6]$.

- Add 15 to each element. [Hint: `x + 15`.]
- Add 3 to only the odd-indexed elements of \mathbf{x} .
- Output the vector whose elements are squares of the corresponding elements of \mathbf{x} . [Hint: `.*` or `.^`]

Problem 2.11. Let $\mathbf{x} = [3, 1, 6, 8]'$ and $\mathbf{y} = [2, 1, 3, 5]'$ (\mathbf{x} and \mathbf{y} are column vectors).

- Add \mathbf{x} to \mathbf{y} .
- Raise each element of \mathbf{x} to the power specified by the corresponding element in \mathbf{y} .
- Divide each element of \mathbf{y} by the corresponding element in \mathbf{x} . [Hint: `./`]
- Multiply each element in \mathbf{x} by the corresponding element in \mathbf{y} , and call the result ' \mathbf{z} '. [Hint: `.*`]
- Add up the elements in \mathbf{z} and assign the result to a variable called ' \mathbf{w} '. [Hint: `w=sum(z)`.]
- Compute $(\sin \mathbf{x})' * \mathbf{y} - \mathbf{w}$.

2.6.3 Program Files

MATLAB program (script) files are essentially text files with a file extension ‘.m’. We can start a new script file simply by clicking an icon on the MATLAB window called ‘New Script’ (name may vary in different systems or versions). A program file can contain a series of commands to be executed (scripts), or it can contain a function that accepts input arguments and produces output. Let’s open a new script file and type the following in the file:

```
a = 3.5;
b = 1;
x = sin(a) - b;
```

Save this file as ‘testscript.m’ in our working directory. In the command window, type

```
>> testscript
```

and press enter, and you will see

```
x =
-1.3508
```

We can also execute the script file by directly clicking the ‘Run’ button on the script file window.

Another way to obtain the result is to make the file a function file. Open another script file, and name it ‘fun1.m’. In the file, type and save

```
function x = fun1(a,b)
x = sin(a) - b;
```

In the command window, type

```
>> fun1(3.5,1)
```

and you can see that it produces the same result as before. We can try more sets of (a,b)

```
>> a1 = fun1(2,-1)
>> a2 = fun1(-2.4,10)
```

2.6.4 Numerical Algorithms for Solving Ordinary Differential Equations

Most of the time, the solution of an ordinary differential equation problem (2.1) does not have a closed-form solution. In this case, one looks for numerical solutions that approximate the exact solution. Since numerical solutions are just approximations, it is also important to understand the accuracy and robustness of the numerical method.

Suppose the initial value problem is

$$\frac{dx}{dt} = f(x,t), \quad t \geq t_0, \quad x(t_0) = x_0. \quad (2.22)$$

Let t_n be some time point with $t_n \geq t_0$, then by integrating the equation from t_n to t , one gets

$$x(t) = x(t_n) + \int_{t_n}^t f(x, \tau) d\tau \approx x(t_n) + (t - t_n)f(x(t_n), t_n). \quad (2.23)$$

The approximation of the integral in (2.23) is good as long as t is sufficiently close to t_n . Suppose we would like to compute the solution of (2.22) at $t = T$, $T > t_0$. To get an approximate solution at time T , we can discretize the interval $[t_0, T]$ into N uniform subintervals $[t_n, t_{n+1}]$, $n = 0, \dots, N-1$, with $t_N = T$ and $t_{n+1} - t_n = h = \frac{T}{N}$. We call h the step size. We will use lowercase x to denote the exact solution of (2.22) and capital X to denote the approximate solution.

Using the approximation in (2.23), we then define a numerical scheme by

$$X_{n+1} = X_n + hf(X(t_n), t_n). \quad n = 0, \dots, N-1, \quad (2.24)$$

where X_n is the approximation of $x(t_n)$. This is called the **forward Euler Method**, named after Leonhard Euler (1707–1783). The error of this scheme is $O(h)$, which can be formally derived from the Taylor expansion. Hence, the smaller the time step size is, the more accurate the approximate solution will be. Generally, a numerical scheme is called k -th order accurate if the error is $O(h^k)$, where h is the discretization size. So Euler method is first order accurate. Although nowadays there are many high order accurate schemes to solve ordinary differential equations, Euler method is still a classical one when we first learn numerical methods. In MATLAB, we have some options of using Runge-Kutta methods [2] to solve ordinary differential equations, which will be introduced as follows.

Using MATLAB to Solve ODE

When solving problem (2.22) with MATLAB, we need to provide three pieces of information for the program:

1. the right-hand side function $f(x, t)$;
2. the initial condition $x(t_0) = x_0$;
3. the integration interval $[t_0, T]$.

The first step is to define functions in MATLAB. Recall that we introduced in the previous subsection about using a function file to define a single function, which reads in arguments and produces outputs. Another way is to use 'function handle,' a MATLAB value that provides a means of calling a function indirectly. For example, to define $f(x, t) = t - 2x$, we can type in the command window

```
>> f = @(x,t) t-2*x; %The @ operator constructs a
function handle for this function
>> f(3,1)
ans =
    -5
```

Problem 2.12. Try to use a script file to define the above function.

Now, to solve a simple ODE

$$\frac{dx}{dt} = t - 2x, \quad 0 \leq t \leq 2, \quad x(0) = 1,$$

we can type the following in the command window:

```
>> g = @(t,x) (t-2*x);
>> tspan = [0, 0.2]; % integrate the ODE from 0 to 0.2
>> x0 = 2; % the initial condition x(0) = 2
>> [t,x] = ode45(g,tspan,x0)
```

Note that the first argument in the function g is ' t ' and the second is ' x '; we have to keep this order (time is first, followed by other variables) when we define functions for MATLAB ODE solvers. The ODE solver we used is 'ode45,' a built-in Runge-Kutta solver in MATLAB.

Also, when we output variables t and x , we can see that t and x are column vectors. The vector t records the discrete time points in the MATLAB simulations, starting at 0 (the initial time) and ending at 0.2 (the final time). The vector x is of the same length as t , and the elements are the approximate solutions at time corresponding to elements in vector t (the first element of x is 2, which is the initial condition).

We can save the above commands in a script file so that we do not have to retype next time. A slightly different version is to use a function file to define the right-hand side function $f(x,t)$, see Algorithms 2.1 and 2.2 (run 'main_BacterialGrowth.m', and 'fun_BacterialGrowth.m' is to be called when 'ode45' is solving the ODE). A plot of x versus t will be shown by the 'plot' command.

Algorithm 2.1. Main script file to solve $dx/dt = t - 2x$ (main_BacterialGrowth.m)

```
%%% This code solves the ODE dx/dt=t-2x, 0<=t<=0.2 with x(t=0)=2

tspan = [0,0.2]; % integrate the ode from 0 to 0.2
x0 = 2;          % the initial condition x(0) = 2
[t,x] = ode45('fun_BacterialGrowth',tspan,x0);
plot(t,x)
```

Algorithm 2.2. fun_BacterialGrowth.m

```
%%% This function will be called by main_BacteriaGrowth.m
function dx = fun_BacterialGrowth(t,x)
dx = t - 2*x;
```

Problem 2.13. Write a code to solve the ODE (refer to Eq. (2.17))

$$\frac{dx}{dt} = x \left(1 - \frac{x}{2} \right), \quad 0 \leq t \leq 10,$$

with initial condition $x(0) = 0.5$.

- (a) Run the code and get the two column vectors of discrete time points and the corresponding approximate solutions.
- (b) Use the vector of time points to compute a vector containing the exact solution at those time points. [Hint: refer to formula (2.18); exponential of x is 'exp(x)' in MATLAB.]
- (c) Compute the absolute value of the difference between the approximate and exact solutions.
- (d) Plot the numerical solution and the exact solution on the same figure with different markers and different colors (refer to the numerical section of Chapter 3 for plotting).

Problem 2.14. Solve the equation in Problem 2.8 with $a = 1$ numerically in the form $x = x(t)$ when (i) $x(0) = \frac{1}{2}$, (ii) $x(0) = \frac{3}{2}$. For (i), plot x for the time interval when finite solution exists (starting from 0); for (ii), plot x for $0 \leq t \leq 10$.

Introduction to Mathematical Biology

Modeling, Analysis, and Simulations

Chou, C.S.; Friedman, A.

2016, VII, 172 p. 49 illus., 38 illus. in color., Hardcover

ISBN: 978-3-319-29636-4