Mathematical Methods for Biology, Part 1

Dmitry Kondrashov

Table of contents

Preface

In this book you will find a collection of mathematical ideas, computational methods, and modeling tools for describing biological systems quantitatively. Biological science, like all natural sciences, is driven by experimental results. As with other sciences, there comes a point when accumulated data needs to be analyzed quantitatively, in order to formulate and test explanatory hypotheses. Biology has reached this stage, thanks to an explosion of data from molecular biology techniques, such as large-scale DNA sequencing, protein structure determination, data on gene regulatory networks, and signaling pathways. Quantitative skills have become necessary for anyone hoping to make sense of biological research.

Mathematical modeling necessarily involves making simplifying assumptions. Reality is generally too complex to be captured in a few equations, and this is especially true for living systems. Simplicity in modeling has at least two virtues: first, simple models can be grasped by our limited minds, and second, it allows for meaningful testing of the assumptions against the evidence. A complex model that fits the data may not provide any insights about how the system works, whereas a simple model which does not fit all the data can indicate where the assumptions break down. We will learn how to construct progressively more sophisticated models, beginning with the ridiculously simple.

modeling assumptions: theoretical and empirical

A mathematical model postulates a precise relationship between several quantities, attempting to mimic the behavior of a real system. All models rest on a set of assumptions, postulating how various quantities are interrelated. These assumptions generally come from two sources: a scientific theory, or experimental observations. For instance, a model of molecular motion may rest on the assumption that Newton's laws hold true. On the other hand, the observation that a drug injected into the bloodstream of a mammal is metabolized with an exponential time dependence is empirical. The benefit of models based on well-established theories, sometimes known as "first-principles models", is that they can be constructed without prior experimental knowledge of a particular system. Newton's laws apply to all sorts of classical mechanics objects, ranging in size from molecules to planets. Some prefer first-principles models, because they rely on well-established scientific principles, while others will argue that an empirical model more accurately reflects the behavior of the system at hand. From a mathematical standpoint, there is no difference between the two types of models. We will use the same tools to construct and analyze models, regardless of their origin.

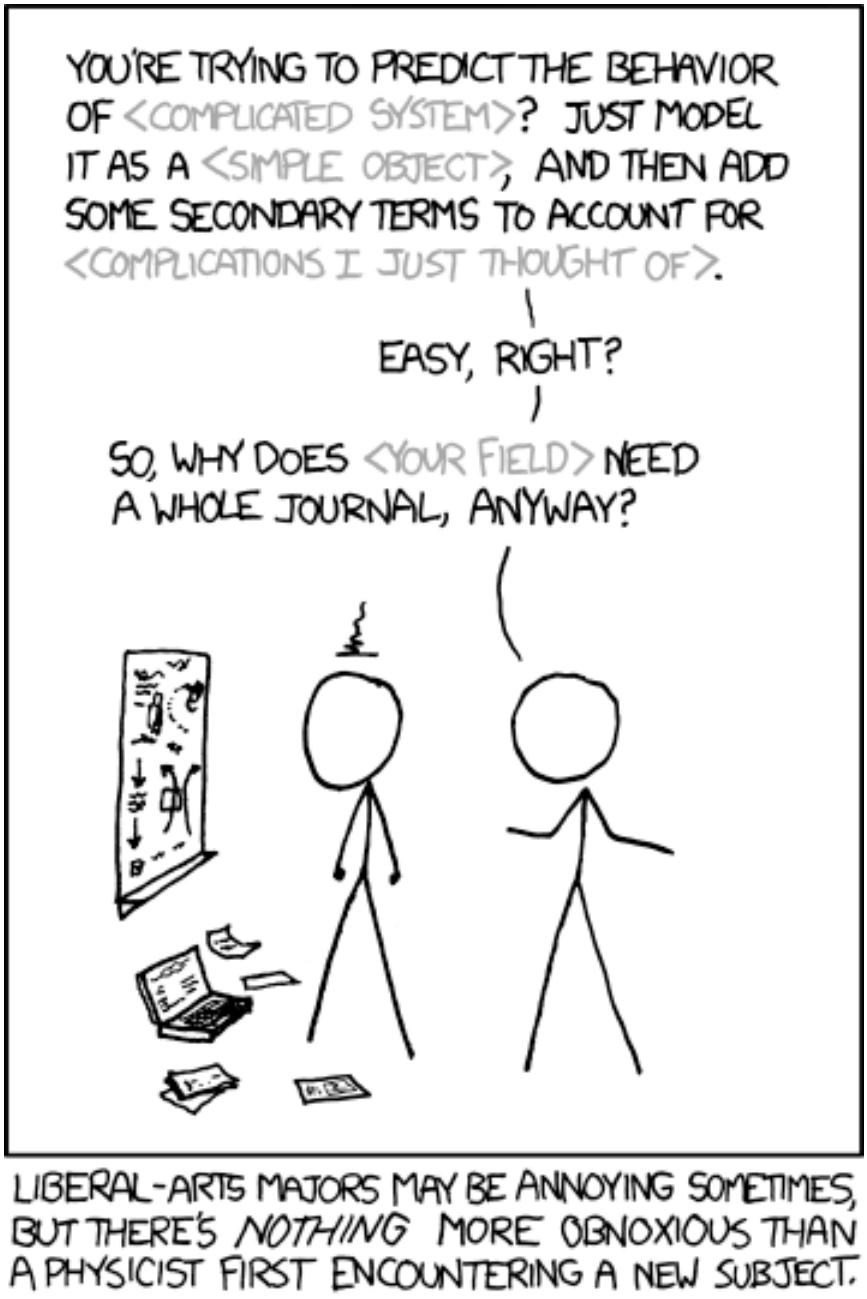


Figure 1: Beware: a little knowledge of mathematical modeling can lead to arrogance. <http://xkcd.com/793/

A stated assumption can be written as a mathematical relationship, usually in the form of an equation relating quantities of interest. A postulated assumption may be expressed in words as "X is proportional to Y", and can be written as the following equation: $X = aY$. Another model may postulate a relationship "X is inversely proportional to the product of Y and Z", which is expressed as $X = a/YZ$.

Suppose we want to model the relationship between the height of individuals (H) and their weight (W). Measuring those quantities in some population results in the observation that the weight is proportional to the height, with an additive correction. Then we can write the following mathematical model, based on the empirical evidence: $W = aH + c$

In electricity, Ohm's law governs the relationship between the flow of charged particles, called current (I), the electric potential (V) and the resistance of a conductor (R). This law states that the current through a conductor is proportional to the potential and inversely proportional to the resistance, and thus can be mathematically formulated:

$$I = \frac{V}{R}$$

variables and parameters

Mathematical models formulate relationships between different quantities that can be measured in real systems. There are two different types of quantities in models: *variables* and *parameters*. The same measurable quantity can be a variable or a parameter, depending on the role it plays in the model. A variable typically varies, either in time or in space, and the model tracks the changes in its value. On the other hand, a parameter typically usually stays the same for a particular manifestation of the model, e.g. an individual or a specific population. However, parameters can vary from individual to individual, or from population to population.

In the height and weight model above, the numbers H are W are the variables, which can change between different individuals. The parameters a and c can either be estimated from data for various subpopulations. Perhaps the values of the parameters are different for young people than for older people, or they are different for those who exercise regularly versus those who do not. Once the parameters have been set, one can predict W given H, or vice versa. Of course, since this is a model, it is only an approximation of reality. The deviations of predictions of the model from actual height or weight for an individual may tell us something interesting about the physiology of the individual.

There are three quantities in the equation for Ohm's law, and the distinction between variables and parameters depends on the actual system that is being modeled. In order to distinguish between the two, consider which quantity is set prior to the experiment, and which one may vary over the course of the situation we are trying to model. For instance, if voltage is being applied to a material with constant resistance, and the potential may be varied, then V is the

independent variable, I is the dependent variable, and R is a parameter. On the other hand, if the setup uses a variable resistor (known as a potentiometer or pot), and the voltage remains constant, then V is a parameter, while I and R are variables. If both the voltage V and the resistance R can vary at the same time, then all three quantities are variables.

units and dimensions

Each variable and parameter has its own *dimension*, which describes the physical or biological meaning of the quantity. Examples are time, length, number of individuals, or concentration per time. It is important to distinguish the dimension of a quantity from the *units* of measurement. The same quantity can be measured in different units: length can be in meters or feet, population size can be expressed in individuals or millions of individuals. The value of a quantity depends on the units of measurement, but its essential dimensionality does not.

There is a fundamental requirement of mathematical modeling: all the terms in an equation must agree in dimensionality; e.g. time cannot be added to number of sheep, since this sum has no biological meaning. In order to express this rule, we will write the dimension of a quantity X as $[X]$. While X refers to a numerical value, $[X]$ describes its physical meaning. Then the above statement can be illustrated by the following example:

$$aX = bY^2 \implies [aX] = [bY^2]$$

In the equation \$W = a H + c \$ all the terms must have the dimension of weight, because that is the meaning of the left hand side of the equation. Therefore, c has the dimensions of weight as well. H of course has the dimension of length, so this implies that the parameter a has dimensions of weight divided by length. This can be summed up as follows:

$$[W] = [c] = weight;\ [H] = length;\ [a] = \frac{weight}{length}$$

While the dimensions are set by the equation, the units of these quantities can vary. Weight can be expressed in pounds, kilograms, or stones, and length can be represented in inches, meters, or light years.

The dimensions of current are defined to be the amount of charge moving per unit of time, and the dimensions of voltage are energy per unit of charge. This allows us to find the dimensions of resistance by the following basic algebra:

$$[V] = \frac{energy}{charge} = \frac{[I]}{[R]} = \frac{charge/time}{[R]} \implies [R] = \frac{charge^2}{energy * time}$$

Electric potential is measured in volts, and current in amperes. The standard unit of resistance is the Ohm, which is defined as one volt per ampere. But regardless of the choice of units, the dimensions of these quantities remains.

A quantity may be made *dimensionless* by expressing it in terms of particular *scale*. For instance, we can express the height of a person as a fraction of the mean height of the population. A tall person will have height expressed as a number greater than 1, and a short one will have height less than 1. Note that this dimensionless height has no units - they have been divided out by scaling the height by the mean height. In fact, the word dimensionless is somewhat misleading: while such quantities have no scale in the context of the algebraic relationship, a quantity retains its physical significance after rescaling: height expressed as a fraction of some chosen length still represents height. Nevertheless, the accepted term in dimensionless quantity, and we will stick with this convention. Later in the book we will learn how to use the technique of rescaling to simplify and analyze dynamic models.

1 One variable in discrete time

All living things change over time, and this evolution can be quantitatively measured and analyzed. Mathematics makes use of equations to define models that change with time, known as *dynamical systems*. In this unit we will learn how to construct models that describe the time-dependent behavior of some measurable quantity in life sciences. Numerous fields of biology use such models, and in particular we will consider changes in population size, the progress of biochemical reactions, the spread of infectious disease, and the spikes of membrane potentials in neurons, as some of the main examples of biological dynamical systems.

Many processes in living things happen regularly, repeating with a fairly constant time period. One common example is the reproductive cycle in species that reproduce periodically, whether once a year, or once an hour, like certain bacteria that divide at a relatively constant rate under favorable conditions. Other periodic phenomena include circadian (daily) cycles in physiology, contractions of the heart muscle, and waves of neural activity. For these processes, theoretical biologists use models with *discrete time*, in which the time variable is restricted to the integers. For instance, it is natural to count the generations in whole numbers when modeling population growth.

This chapter is devoted to analyzing dynamical systems in which time is measured in discrete steps. We will build dynamic models, find their mathematical solutions, and then use Python to compute the solutions and plot them. In this chapter you will learn to:

- build discrete-time models of populations using rate paramaters
- define and verify mathematical solutions of these models
- use Python to compute and plot solutions

1.1 Building dynamic models

Let us construct our first models of biological systems! We will start by considering a population of some species, with the goal of tracking its growth or decay over time. The variable of interest is the number of individuals in the population, which we will call N. This is called the dependent variable, since its value changes depending on time; it would make no sense to say that time changes depending on the population size. Throughout the study of dynamical systems, we will denote the independent variable of time by t. To denote the population size at time t, we can write $N(t)$ but sometimes use N_t.

1.1.1 static population

In order to describe the dynamics, we need to write down a rule for how the population changes. Consider the simplest case, in which the population stays the same for all time. (Maybe it is a pile of rocks?) Then the following equation describes this situation:

$$N(t+1) = N(t)$$

This equation mandates that the population at the next time step be the same as at the present time t. This type of equation is generally called a *difference equation*, because it can be written as a difference between the values at the two different times:

$$N(t+1) - N(t) = 0$$

This version of the model illustrates that a difference equation at its core describes the *increments* of N from one time step to the next. In this case, the increments are always 0, which makes it plain that the population does not change from one time step to the next.

1.1.2 exponential population growth

Let us consider a more interesting situation: as a colony of dividing bacteria. such as *E. coli*, shown in {numref}`fig-cell-div`. We assume that each bacterial cell divides and produces two daughter cells at fixed intervals of time, and let us further suppose that bacteria never die. Essentially, we are assuming a population of immortal bacteria with clocks. This means that after each cell division the population size doubles. As before, we denote the number of cells in each generation by $N(t)$, and obtain the equation describing each successive generation:

$$N(t+1) = 2N(t)$$

It can also be written in the difference form, as above:

$$N(t+1) - N(t) = N(t)$$

The increment in population size is determined by the current population size, so the population in this model is forever growing. This type of behavior is termed *exponential growth* and we will see how to express the solution algebraically in the next section.

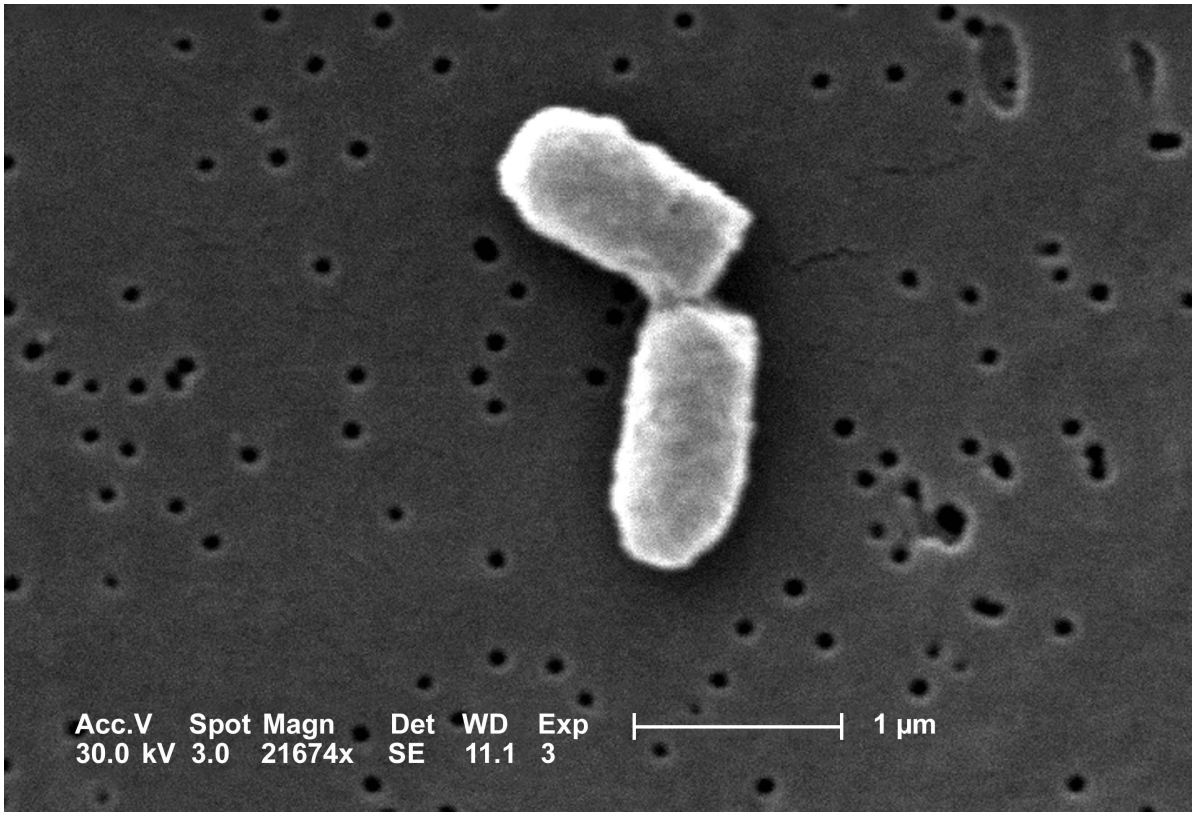


Figure 1.1: Scanning electron micrograph of a dividing *Escherichia coli* bacteria (image by Evangeline Sowers, Janice Haney Carr (CDC) in public domain via Wikimedia Commons)

1.1.3 example with birth and death

Suppose that a type of fish lives to reproduce only once after a period of maturation, after which the adults die. In this simple scenario, half of the population is female, a female always lays 1000 eggs, and of those, 1% survive to maturity and reproduce. Let us set up the model for the population growth of this idealized fish population. The general idea, as before, is to relate the population size at the next time step $N(t+1)$ to the population at the present time $N(t)$.

Let us tabulate both the increases and the decreases in the population size. We have $N(t)$ fish at the present time, but we know they all die after reproducing, so there is a decrease of $N(t)$ in the population. Since half of the population is female, the number of new offspring produced by $N(t)$ fish is $500N(t)$. Of those, only 1% survive to maturity (the next time step), and the other 99% ($495N(t)$) die. We can add all the terms together to obtain the following difference equation:

$$N(t+1) = N(t) - N(t) + 500N(t) - 495N(t) = 5N(t)$$

The number 500 in the expression is the *birth rate* of the population per individual, and the negative terms add up to the *death rate* of 496 per individual. We can re-write the equation in difference form:

$$N(t+1) - N(t) = 4N(t)$$

This expression again generates growth in the population, because the birth rate outweighs the death rate. (**allman_mathematical_2003?**)

1.1.4 dimensions of birth and death rates

What distinguishes a mathematical model from a mathematical equation is that the quantities involved have a real-world meaning. Each quantity represents a measurement, and associated with each one are the *units* of measurement, which are familiar from science courses. In addition to units, each variable and parameter has a meaning, which is called the *dimension* of the quantity. For example, any measurement of length or distance has the same dimension, although the units may vary. The value of a quantity depends on the units of measurement, but its essential dimensionality does not. One can convert a measurement in meters to that in light-years or cubits, but one cannot convert a measurement in number of sheep to seconds - that conversion has no meaning.

Thus leads us to the fundamental rule of mathematical modeling: **terms that are added or subtracted must have the same dimension**. This gives mathematical modelers a useful tool called *dimensional analysis*, which involves replacing the quantities in an equation with

their dimensions. This serves as a check that all dimensions match, as well as allowing to deduce the dimensions of any parameters for which the dimension was not specified.

In the case of population models, the birth and death rates measure the number of individuals that are born (or die) within a reproductive cycle for every individual at the present time. Their dimensions must be such that the terms in the equation all match:

$$[N(t+1) - N(t)] = [population] = [r][N(t)] = [r] * [population]$$

This implies that $[r]$ is algebraically dimensionless. However, the meaning of r is the rate of change of population over one (generation) time step. r is the birth or death rate of the population *per generation*, which is what makes is dimensionless. If the length of the generation were to change, but the reproduction and death per generation remain the same, then the paramter r would be the same, because it had been *rescaled* by the length of the generation. If they were to be reported in *absolute* units (e.g. individuals per year) then the rate would be different.

1.1.5 general demographic model

We will now write a general difference equation for any population with constant birth and death rates. This will allow us to substitute arbitrary values of the birth and death rates to model different biological situations. Suppose that a population has the birth rate of b per individual, and the death rate d per individual. Then the general model of the population size is:

$$N(t+1) = (1 + b - d)N(t)$$

(lin-pop)

The general equation also allows us to check the dimensions of birth and death rates, especially as written in the incremental form: $N(t+1) - N(t) = (b-d)N(t)$. The change in population rate over one reproductive cycle is given by the current population size multiplied by the difference of birth and death rates, which as we saw are algebraically dimensionless. The right hand side of the equation has the dimensions of population size, matching the difference on the left hand side. (**edelstein-keshet_mathematical_2005?**)

1.2 Solutions of linear difference models

We saw in the last section that we can write down equations to describe, step by step, how a variable changes over time. Let us define what the terminology of these equations:

Definition

An equation to describe a variable (e.g. N) that changes over discrete time steps described by the integer variable t is called a *difference equation* or a *discrete-time dynamic model*. These equations can be written in two ways, either in *recurrent form*:

$$N(t+1) = f(N(t))$$

(recur-eq)
or in *increment form*:

$$N(t+1) - N(t) = g(N(t))$$

(recur-eq)

1.2.1 simple linear difference models

Having set up the difference equation models, we would naturally like to solve them to find out how the dependent variable, such as population size, varies over time. A solution may be *analytic*, meaning that it can be written as a formula, or *numeric*, in which case it is generated by a computer in the form of a sequence of values of the dependent variable over a period of time. In this section, we will find some simple analytic solutions and learn to analyze the behavior of difference equations which we cannot solve exactly.

Definition

A function $N(t)$ is a *solution* of a difference equation $N(t+1) = f(N(t))$ if it satisfies that equation for all values of time t.

For instance, let us take our first model of the static population, $N(t+1) = N(t)$. Any constant function is a solution, for example, $N(t) = 0$, or $N(t) = 10$. There are actually as many solutions as there are numbers, that is, infinitely many! In order to specify exactly what happens in the model, we need to specify the size of the population at some point, usually, at the "beginning of time", $t = 0$. This is called the *initial condition* for the model, and for a well-behaved difference equation it is enough to determine a unique solution. For the static model, specifying the initial condition is the same as specifying the population size for all time.

Now let us look at the general model of population growth with constant birth and death rates. We saw in equation {eq}`lin-pop` above that these can be written in the form $N(t+1) = (1+b-d)N(t)$. To simplify, let us combine the numbers into one growth parameter $r = 1+b-d$, and write down the general equation for population growth with constant growth rate:

$$N(t+1) = rN(t)$$

(lin-pop-r)

To find the solution, consider a specific example, where we start with the initial population size $N_0 = 1$, and the growth rate $r = 2$. The sequence of population sizes is: 1, 2, 4, 8, 16, etc. This is described by the formula $N(t) = 2^t$.

In the general case, each time step the solution is multiplied by r, so the solution has the same exponential form. The initial condition N_0 is a multiplicative constant in the solution, and one can verify that when $t = 0$, the solution matches the initial value:

$$N(t) = r^t N_0$$

(lin-pop-sol)

I would like the reader to pause and consider this remarkable formula. No matter what the birth and death parameters are selected, this solution predicts the population size at any point in time t.

In order to verify that the formula for $N(t)$ is actually a solution in the meaning of definition, we need to check that it actually satisfies the difference equation for all t, not just a few time steps. This can be done algebraically by plugging in $N(t+1)$ into the left side of the dynamic model and $N(t)$ into the right side and checking whether they match. For $N(t)$ given by equation {eq}`lin-pop-sol`, $N(t+1) = r^{t+1}N_0$, and thus the dynamic model becomes:

$$r^{t+1}N_0 = r \times r^t N_0$$

Since the two sides match, this means the solution is correct.

The solutions in equation {eq}`lin-pop-sol` are exponential functions, which have a limited menu of behaviors, depending on the value of r. If $r > 1$, multiplication by r increases the size of the population, so the solution $N(t)$ will grow (see {numref}`fig-exp-growth`. If $r < 1$, multiplication by r decreases the size of the population, so the solution $N(t)$ will decay (see {numref}`fig-exp-decay`). Finally, if $r = 1$, multiplication by r leaves the population size unchanged, like in the pile of rocks model. Here is the complete classification of the behavior of population models with constant birth and death rates (assuming $r > 0$):

Classification of solutions of linear dynamic models

For a difference equation $N(t+1) = rN(t)$, solutions can behave in one of three ways:

- $|r| > 1$: $N(t)$ grows without bound
- $|r| < 1$: $N(t)$ decays to 0

- $|r| = 1$: the absolute value of $N(t)$ remains constant

Se examples of graphs of solutions of such equations with r greater than 1 in {numref}`fig-exp-growth` and solutions for r less than 1 in {numref}`fig-exp-decay`.

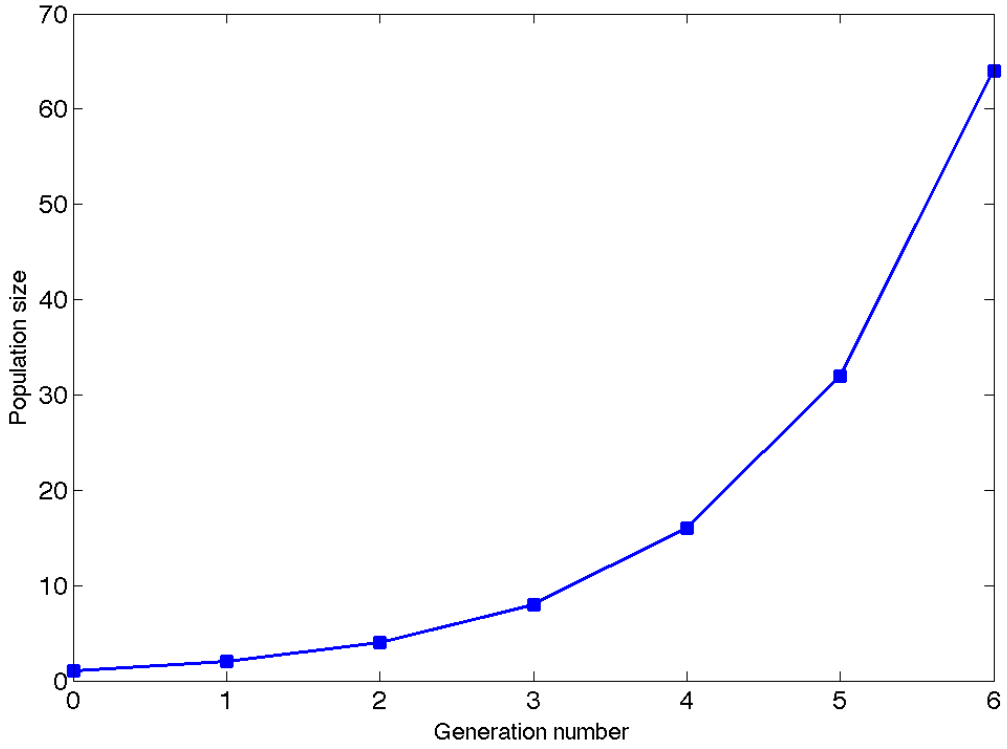


Figure 1.2: Growth of a population that doubles every generation over 6 generations

1.2.2 linear difference models with a constant term

Now let us consider a dynamic model that combines two different rates: a proportional rate (rN) and a constant rate which does not depend on the value of the variable N. We can write such a generic model as follows:

$$N(t+1) = rN(t) + a$$

The right-hand-side of this equation is a linear function of N, so this is a linear difference equation with a constant term. What function $N(t)$ satisfies it? One can quickly check that that the same solution $N(t) = r^t N_0$ does not work because of the pesky constant term a:

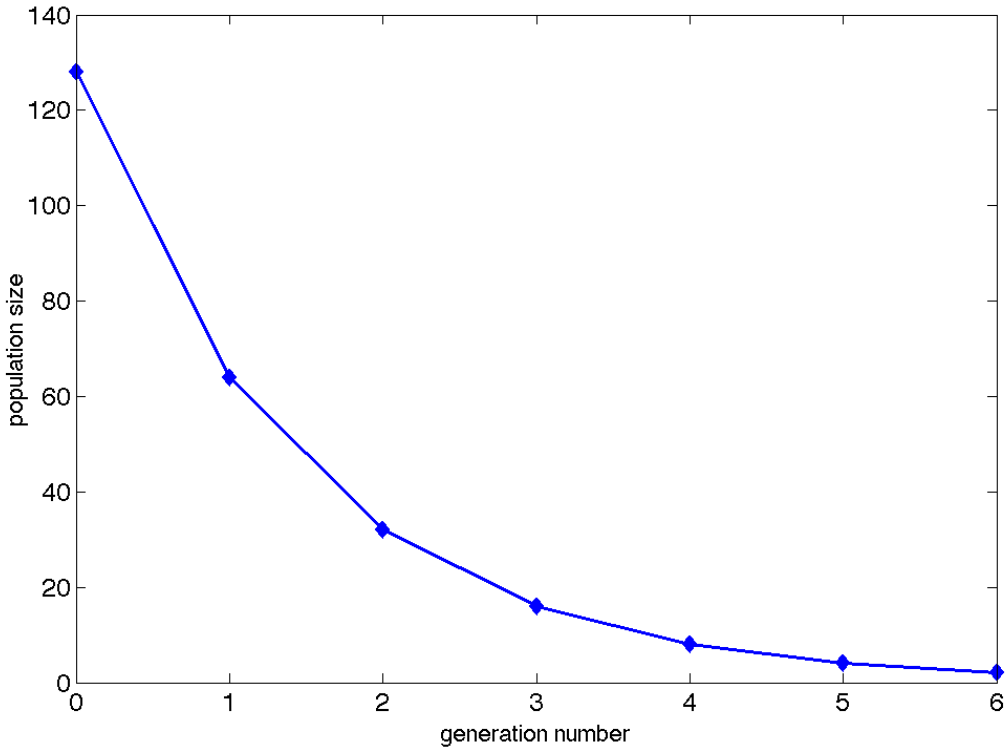


Figure 1.3: Decay of a population in which half the individuals die every time step over 6 generations

$$r^{t+1}N_0 \neq r \times r^t N_0 + a$$

To solve it, we need to try a different form: specifically, an exponential with an added constant. The exponential can be reasonably surmised to have base r as before, and then leave the two constants as unknown: $N(t) = c_1 r^t + c_2$. To figure out whether this is a solution, plug it into the linear difference equation above and check whether a choice of constants can make the two sides agree:

$$N(t+1) = c_1 r^{t+1} + c_2 = rN(t) + a = rc_1 r^t + rc_2 + a$$

This equation has the same term $c_1 r^{t+1}$ on both sides, so they can be subtracted out. The remaining equation involves only c_2, and its solution is $c_2 = a/(1-r)$. Therefore, the general solution of this linear difference equation is the following expreis which is determined from the initial value by plugging $t = 0$ and solving for c.

$$N(t) = cr^t + \frac{a}{1-r}$$

Example. Take the difference equation $N(t+1) = 0.5N(t) + 40$ with initial value $N(0) = 100$. The solution, according to our formula is $N(t) = c0.5^t + 80$. At $N(0) = 100 = c + 80$, so $c = 20$. Then the compete solution is $N(t) = 20 * 0.5^t + 80$. To check that this actually works, plug this solution back into the difference equation:

$$N(t+1) = 20 \times 0.5^{t+1} + 80 = 0.5 \times (20 \times 0.5^t + 80) + 40 = 20 \times 0.5^{t+1} + 80$$

The equation is satisfied and therefore the solution is correct.

2 Plotting in Python

You can find an introduction to the plotting library matplotlib here.

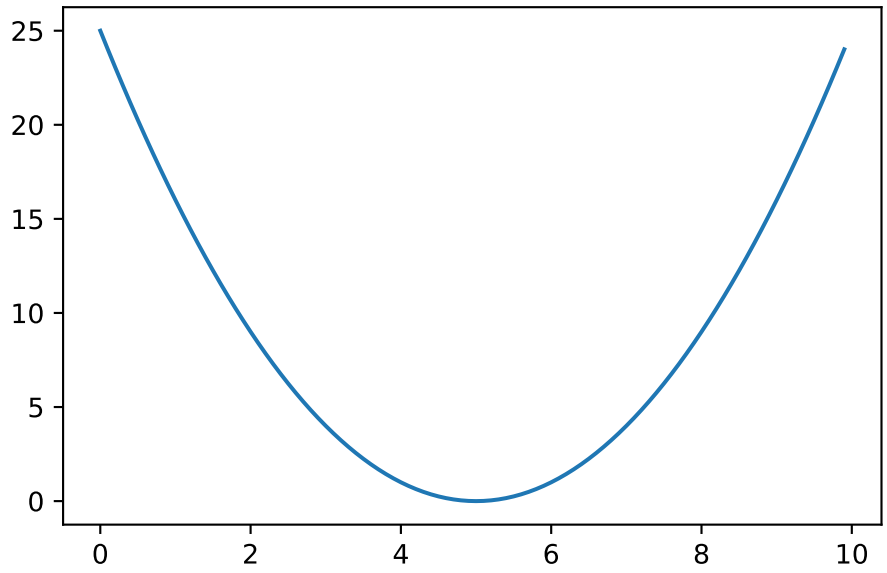
```
# Import packages
import numpy as np # package for work with arrays and matrices
import matplotlib.pyplot as plt # package with plotting capabilities
```

2.0.1 arrays and basic plotting

Here is an example of performing calculations with arrays (vectors) of values and plotting the results:

```
x = np.arange(0,10,0.1) # create an array of numbers between 0 and 10 with step 0.1
print(np.shape(x))
y = (x-5)**2 # do calculations on all the array values, call it y
plt.plot(x,y) # plot x vs y
plt.show()
```

```
(100,)
```



A two dimensional array (matrix) can be defined as follows, and the function np.shape prints out the number of rows and columns in the matrix:

```
x = np.array([[1,2,3],[4,5,6]])
print(np.shape(x))
print(x)
```

```
(2, 3)
[[1 2 3]
 [4 5 6]]
```

An example of concatenating a text string together with a numeric variable, which can then be used for labels or legends in plots:

```
prob = 0.5
string1 = 'The value of prob is ' + str(prob)
print(string1)
```

```
The value of prob is 0.5
```

2.1 Numeric solutions of discrete models

Difference equations, as we saw above, can be written in the form of $x_{t+1} = f(x_t)$. At every step, the model takes the current value of the dependent variable x_t, feeds it into the function $f(x)$, and takes the output as the next value x_{t+1}. The same process repeats every iteration, which is why difference equations written in this form are called *iterated maps*.

Computers are naturally suited for precise, repetitive operations. In our first example of a computational algorithm, we will iterate a given function to produce a sequence of values of the dependent variable x. We only need two things: to specify a computer function $f(x)$, which returns the value of the iterated map for any input value x, and the initial value x_0. Then it is a matter of repeating the operation of evaluating $f(x_t)$ and storing it as the next value x_{t+1}. Below is the pseudocode for the algorithm. Note that I will use arrows to indicated variable assignment, square brackets [] for indexing of vector, and start indexing at 0, consistent with python convention.

Iterative solution of difference equations:

- define the iterated map function $F(x)$
- set N to be the number of iterations (time steps)
- set the initial condition x_0
- initialize array x with initial value x_0
- for i from 0 to $N-1$
  - $x[i+1] \leftarrow F(x[i])$

The resulting sequence of values $x_0, x_1, x_2, ..., x_N$ is called a *numeric solution* of the given difference equation. It has two disadvantages compared to an analytic solution: first, the solution can only be obtained for a specific initial value and number of iterations, and second, any computer simulation inevitably introduces some errors, for instance from round-off. In practice, however, most complex dynamical systems have to solved numerically, as analytical solutions are difficult or impossible to find.

2.1.1 using for loops for iterative solutions of dynamic models

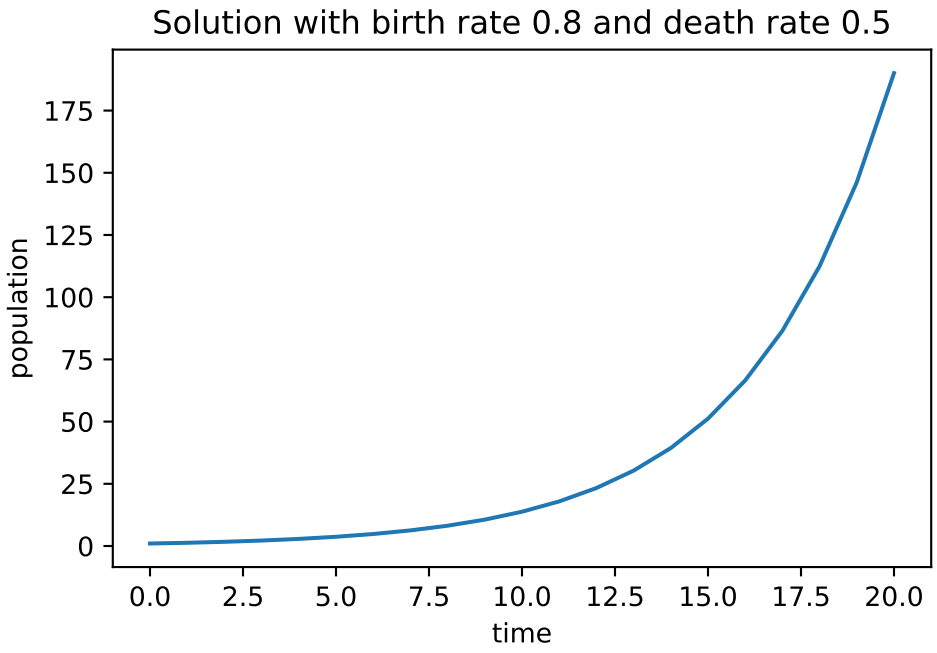
Here is a generic linear demographic model

$$x(t+1) = x(t) + bx(t) - dx(t) = rx(t)$$

Example of a script for producing a numeric solution of a discrete time dynamic model:

```
numsteps = 20 # number of iterations
birth = 0.8 # birth rate
death =  0.5 # death rate
pop = np.zeros(numsteps+1) # initialize solution array
pop[0] = 1 # initial value
t = np.arange(numsteps+1) # initialze time vector
print(t)

for i in range(numsteps):
     pop[i+1] = pop[i] + birth*pop[i] - death*pop[i]# linear demographic model

plt.plot(t, pop) # plot solution
plt.xlabel('time')
plt.ylabel('population')
title = 'Solution with birth rate ' + str(birth) + ' and death rate ' + str(death)
plt.title(title)
plt.show()
```

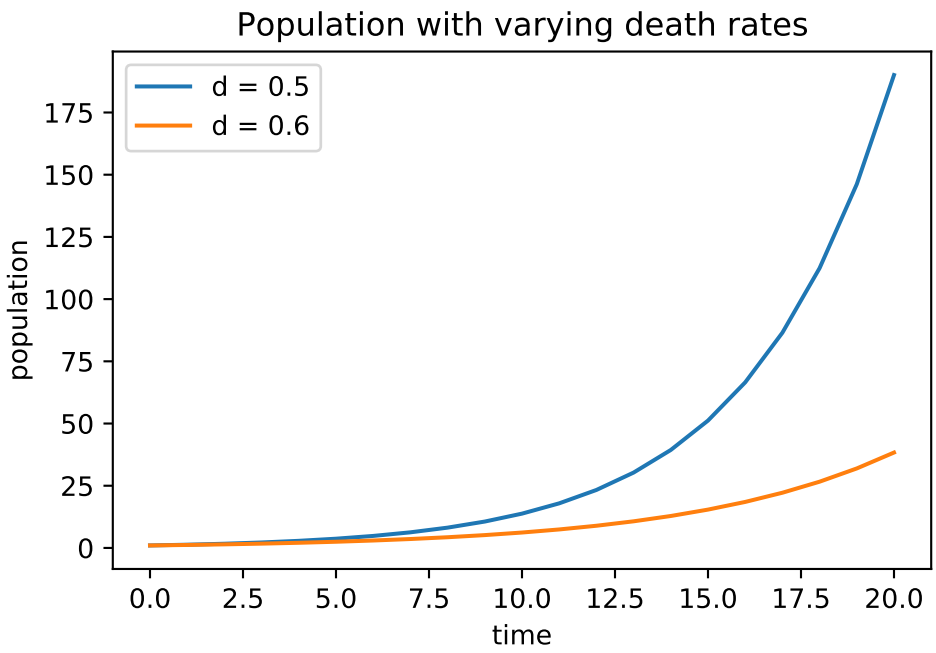
```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20]
```



2.1.2 plotting multiple curves with a legend

Multiple solution plots can be overlayed on the same figure, as long as the `plt.show()` is only used once in the end. For multiple graphs it's best to use multiple colors and a legend to label different curves, using the option label in the `plt.plot` function and adding the function `plt.legend()` before producing the figure. Here's an example with solutions of the demographic model with different death rates:

```
numsteps = 20 # number of iterations
birth = 0.8 # birth rate
death =  0.5 # death rate
pop = np.zeros(numsteps+1) # initialize solution array
pop[0] = 1 # initial value
t = np.arange(numsteps+1) # initialze time vector
for i in range(numsteps):
     pop[i+1] = pop[i] + birth*pop[i] - death*pop[i]# linear demographic model

plt.plot(t, pop, label = 'd = '+str(death)) # plot solution
plt.xlabel('time')
plt.ylabel('population')
death =  0.6 # death rate
pop = np.zeros(numsteps+1) # initialize solution array
pop[0] = 1 # initial value
t = np.arange(numsteps+1) # initialze time vector
for i in range(numsteps):
     pop[i+1] = pop[i] + birth*pop[i] - death*pop[i]# linear demographic model
plt.plot(t, pop, label = 'd = '+str(death)) # plot solution

title = 'Population with varying death rates'
plt.title(title)
plt.legend()
plt.show()
```



2.1.3 random number generators

Numpy provides a variety of random number generators, and we'll use these functions in the course for many purposes. Here is an example of producing arrays of random normally distributed numbers. The function requires inputs of the mean, the standard deviation, and the number of random values (or size of the array):

```
mu = 5

sigma = 0.5

num = 30

norm_sample = np.random.normal(mu, sigma, num)

print(norm_sample)

print("The mean of the sample is " + str(np.mean(norm_sample)))

print("The standard deviation of the sample is " + str(np.std(norm_sample)))
```

```
[6.01795616 5.43487299 4.5647434  5.41161873 4.80872723 5.18361005
```

```
 5.82756085 4.91704363 5.11600001 5.76689915 5.97596978 4.02992914
 4.58119978 5.26175587 4.62075976 4.57266743 5.30264525 5.3674887
 5.0195385  4.31442091 5.15298383 6.26784174 4.74555664 4.3856282
 5.12493271 6.21916198 5.30773887 4.75252993 5.09079378 6.04353917]
The mean of the sample is 5.172870471078734
The standard deviation of the sample is 0.5785158798242253
```

3 Nonlinear discrete-time dynamic models

In this chapter we will analyze nonlinear discrete dynamical systems. Their solutions, as those of nonlinear ODEs, exhibit much more interesting behaviors than the exponential solutions of linear equations, and are typically not solvable analytically. There may be multiple fixed points, some stable and others unstable, and even crazier behaviors are possible that are not permitted in smooth-flowing ODEs. Specifically, we will see solutions that oscillate, and those that behave without any pattern at all, that are called chaotic. You will learn to do the following in this chapter:

- build the logistic population model
- find equilibrium values of nonlinear discrete-time models
- analyze the stability of equilibria based on the graph of the updating function
- write down stability conditions analytically
- use Python to make cobweb plots
- understand the term chaos

3.1 Logistic population model

Linear population growth models assume that the per capita birth and death rates are constant, that is, they stay the same regardless of population size. The solutions for these models either grow or decay exponentially, but in reality, populations cannot grow without bounds. It is generally true that the larger a population grows, the more scarce the resources, and survival becomes more difficult. For larger populations, this could lead to higher death rates, or lower birth rates, or both.

To incorporate this effect into a quantitative model we will assume there are separate birth and death rates, and that the birth rate declines as the population grows, while the death rate increases:

$$b = b_1 - b_2 N(t);\ d = d_1 + d_2 N(t)$$

To model the rate of change of the population, we need to multiply the rates b and d by the population size N, since each individual can reproduce or die. Also, since the death rate d decreases the population, we need to put a negative sign on it. The resulting model is:

$$N(t+1) - N(t) = (b-d)N(t) = [(b_1 - d_1) - (b_2 + d_2)N(t)]N(t)$$

A simpler way of writing this equation is to let $r = 1 + b_1 - d_1$ and $K = b_2 + d_2$, leading to the following iterated map:

$$N(t+1) = (r - KN(t))N(t)$$

(discr-log)

This is called the *logistic model* of population growth. As you see, it has two different parameters, r and K. If $K = 0$, the equation reduces to the old linear population model. Intuitively, K is the parameter describing the effect of increasing population on the population growth rate. Let us analyze the dimensions of the two parameters, by writing down the dimensions of the variables of the difference equation. The dimensional equation is:

$$N(t+1) = [population] = [r - KN(t)]N(t) == ([r] - [K] \times [population]) \times [population]$$

Matching the dimensions on the two sides of the equation leads us to conclude that the dimensions of r and k are different:

$$[r] = 1;\ [K] = \frac{1}{[population]}$$

The difference equation for the logistic model is *nonlinear*, because it includes a second power of the dependent variable. In general, it is difficult to solve nonlinear equations, but we can still say a lot about this model's behavior without knowing its explicit solution.

3.2 Qualitative analysis of difference equations

3.2.1 fixed points or equilibria

We have seen that the solutions of difference equations depend on the initial value of the dependent variable. In the examples we have seen so far, the long-term behavior of the solution does not depend dramatically on the initial condition. In more complex systems that we will encounter, there are special values of the dependent variable for which the dynamical system is constant, like in the pile of rocks model.

Definition

For a difference equation in recurrent form $x(t+1) = f(X(t))$, a point x^* which satisfies $f(x^*) = x^*$ is called a *fixed point* or *equilibrium*. If the initial condition is a fixed point, $x_0 = x^*$, the solution will stay at the same value for all time, $x(t) = x^*$.

The reason these special points are also known as equilibria is due to the precise balance between growth and decay that is mandated at a fixed point. In terms of population modeling, at an equilibrium the birth rates and the death rates are equal. Speaking analytically, in order to find the fixed points of a difference equation, one must solve the equation $f(x^*) = x^*$. It may have none, or one, or many solutions.

Example. The linear population models which we analyzed in the previous sections have the mathematical form $N(t+1) = rN(t)$ (where r can be any real number). Then the only fixed point of those models is $N^* = 0$, that is, a population with no individuals. If there are any individuals present, we know that the population will grow to infinity if $|r| > 1$, and decay to 0 if $|r| < 1$. This is true even for the smallest population size, as long as it is not exactly zero.

Example. Let us go back to the example of a linear difference equation with a constant term. The equation is $ N(t+1) = -0.5N(t) +10$, and we saw that the numerical solutions all converged to the same value, regardless of the initial value. Let us find the equilibrium value of this model using the definition:

$$N^* = -0.5N^* + 10 \Rightarrow 1.5N^* = 10 \Rightarrow N^* = 10/1.5 = 20/3$$

If the initial value is equal to the equilibrium, $N(0) = 20/3$, then the solution will remain constant for all time, since the next value $N(t+1) = -0.5 * 20/3 + 10 = 20/3$ remains the same.

Example: discrete logistic model. Let us use the simplified version of the logistic equation $N(t+1) = r(1 - N(t))N(t)$ and set the right-hand side function equal to the variable N to find the fixed points N^*:

$$r(1 - N^*)N^* = N^*$$

There are two solutions to this equation, $N^* = 0$ and $N^* = (r-1)/r$. These are the fixed points or the equilibrium population sizes for the model, the first being the obvious case when the population is extinct. The second equilibrium is more interesting, as it describes the *carrying capacity* of a population in a particular environment. If the initial value is equal to either of the two fixed points, the solution will remain at that same value for all time. But what happens to solutions which do not start a fixed point? Do they converge to a fixed point, and if so, to which one?

3.2.2 stability criteria for fixed points

What happens to the solution of a dynamical system if the initial condition is very close to an equilibrium, but not precisely at it? Put another way, what happens if the equilibrium is *perturbed*? To answer the question, we will no longer confine ourselves to the integers, to be interpreted as population sizes. We will instead consider, abstractly, what happens if the smallest perturbation is added to a fixed point. Will the solution tend to return to the fixed point or tend to move away from it? The answer to this question is formalized in the following definition (**strogatz_nonlinear_2001?**):

> **i** Definition
>
> For a difference equation $x(t+1) = f(x(t))$, a fixed point x^* is *stable* if for a sufficiently small number ϵ, the solution $x(t)$ with the initial condition $x_0 = x^* + \epsilon$ approaches the fixed point x^* as $t \to \infty$. If the solution $x(t)$ does not approach x^* for any nonzero ϵ, the fixed point is called *unstable*.

The notion of stability is central to the study of dynamical systems. Typically, models more complex than those we have seen cannot be solved analytically. Finding the fixed points and determining their stability can help us understand the general behavior of solutions without writing them down. For instance, we know that solutions never approach an unstable fixed point, whereas for a stable fixed point the solutions will tend to it, from some range of initial conditions.

There is a mathematical test to determine the stability of a fixed point. From standard calculus comes the Taylor expansion, which approximates the value of a function near a given point. Take a general difference equation written in terms of some function $x(t+1) = f(x(t))$. Let us define the *deviation* from the fixed point x^* at time t to be $\epsilon(t) = x_t - x^*$. Then we can use the linear (first-order) Taylor approximation at the fixed point and write down the following expression:

$$x(t+1) = f(x^*) + \epsilon(t) f'(x^*) + ...$$

The ellipsis means that the expression is approximate, with terms of order $\epsilon(t)^2$ and higher swept under the rug. Since we take $\epsilon(t)$ to be small, those terms are very small and can be neglected. Since x^* is a fixed point, $f(x^*) = x^*$. Thus, we can write the following difference equation to describe the behavior of the deviation from the fixed point X^*:

$$x(t+1) - x^* = \epsilon(t+1) = \epsilon(t) f'(x^*)$$

We see that we started out with a general function defining the difference equation and transformed it into a linear equation for the deviation $\epsilon(t)$. Note that the multiplicative constant

here is the derivative of the function at the fixed point: $f'(x^*)$. This is called the *linearization* approach, which is an approximation of a dynamical system near a fixed point with a linear equation for the small perturbation.

We found the solution to simple linear equations, which we can use describe the behavior of the perturbation to the fixed point. The behavior **depends on the value of the derivative of the updating function** $f'(X^*)$:

Important Fact

For a difference equation $x(t+1) = f(x(t))$, a fixed point x^* can be classified as follows:

- $|f'(x^*)| > 1$: the deviation $\epsilon(t)$ grows, and the solution moves away from the fixed point; fixed point is *unstable*
- $|f'(x^*)| < 1$: the deviation $\epsilon(t)$ decays, and the solution approaches the fixed point; fixed point is *stable*
- $|f'(x^*)| = 1$: the fixed point may be stable or unstable, and more information is needed

We now know how to determine the stability of a fixed point, so let us apply this method to some examples.

Example: linear difference equations. Let us analyze the stability of the fixed point of a linear difference equation, e.g. $N(t+1) = -0.5N(t) + 10$. The derivative of the updating function is equal to -0.5. Because it is less than 1 in absolute value, the fixed point is stable, so solutions converge to this equilibrium. We can state more generally that any linear difference equation of the form $N(t+1) = aN(t)+b$ has one fixed point, which is equal to $N^* = b/(1-a)$. This fixed point is stable if $|a| < 1$ and unstable if $|a| > 1$.

Example: discrete logistic model. In the last subsection we found the fixed points of the simplified logistic model. To determine what happens to the solution, we need to determine the stability of both equilibria. Since the stability of fixed points is determined by the derivative of the defining function at the fixed points, we compute the derivative of $f(N) = rN - rN^2$ to be $f'(N) = r - 2rN$, and evaluate it at the two fixed points $N^* = 0$ and $N^* = (r-1)/r$:

$$f'(0) = r; \quad f'((r-1)/r) = r - 2(r-1) = 2 - r$$

Because the intrinsic death rate cannot be greater than the birth rate, we know that $r > 0$. Therefore, we have the following *stability conditions* for the two fixed points:

- the fixed point $N^* = 0$ is stable for $r < 1$, and unstable for $r > 1$;
- the fixed point $N^* = (r-1)/r$ is stable for $1 < r < 3$, and unstable otherwise.

3.3 Analysis of logistic population model

3.3.1 rescaling the logistic model

First, let us do one more modification of the model, by taking the parameter r as the common multiple:

$$N_{t+1} = r(1 - \frac{K}{r}N_t)N_t$$

As we saw, the parameter K has dimension of inverse population size, and that the parameter r is dimensionless. We can now use rescaling of the variable N to simplify the logistic model. The goal is to reduce the number of parameters, by canceling some, and bringing the rest into one place, where they can be combined into a *dimensionless group*. Here is how this is accomplished for this model:

1. Pick a number of the same dimension as the variable, called the scale, and divide the variable by it. In this case, let the scale for population be r/K, so then the new variable is

$$\tilde{N} = \frac{NK}{r} \Longrightarrow N = \frac{\tilde{N}r}{K}$$

Since the parameter K has dimension of inverse population size, NK is in the dimensionless variable $\tilde{N}$.

2. Substitute $\tilde{N}/K$ for N in the equation:

$$\frac{\tilde{N}_{t+1}r}{K} = r\left(1 - \frac{K\tilde{N}_t r}{rK}\right)\frac{\tilde{N}_t r}{K}$$

3. Canceling all the parameters on both sides, we just have the dimensionless growth rate r, as our only parameter:

$$\tilde{N}_{t+1} = r(1 - \tilde{N}_t)\tilde{N}_t$$

On the surface, we merely used algebraic trickery to simplify the equation, but the result is actually rather deep. By changing the dimension of measurement of the population from individuals (N) to the dimensionless fraction of the carrying capacity ($\tilde{N}$) we found that there is only one parameter r that governs the behavior of this model. We will see in the next two section that varying this parameter leads to dramatic changes in the dynamics of the model population. (**edelstein-keshet_mathematical_2005?**)

3.3.2 fixed point analysis

The first step for qualitative analysis of a nonlinear model is to find the fixed points. We use the dimensionless version of the logistic equation, and the right-hand side function equal to the value of the special values N^* (fixed points):

$$r(1 - N^*)N^* = N^*$$

There are two solutions to this equation, $N^* = 0$ and $N^* = (r - 1)/r$. These are the fixed points or the equilibrium population sizes for the model, the first being the obvious case when the population is extinct. The second equilibrium is more interesting, as it describes the *carrying capacity* of a population in a particular environment. To determine what happens to the solution, we need to evaluate the stability of both equilibria.

We have seen in the analytical section that the stability of fixed points is determined by the derivative of the defining function at the fixed points. The derivative of $f(N) = rN - rN^2$ is $f'(N) = r - 2rN$, and we evaluate it at the two fixed points:

$$f'(0) = r; \;\; f'((r-1)/r) = r - 2(r-1) = 2 - r$$

Because the intrinsic death rate cannot be greater than the birth rate, we know that $r > 0$. Therefore, we have the following stability conditions for the two fixed points:

- the fixed point $N^* = 0$ is stable for $r < 1$, and unstable for $r > 1$;
- the fixed point $N^* = (r - 1)/r$ is stable for $1 < r < 3$, and unstable otherwise.

We can plot the solution for the population size of the logistic model population over time. We see that, depending on the value of the parameter r (but not on k), the behavior is dramatically different:

Case 1: $r < 1$**.** The fixed point at $N^* = 0$ is stable and the fixed point is unstable $N^* = (r - 1)/r$. The solution tends to 0, or extinction, regardless of the initial condition, which is illustrated in figure [fig:sol_logistic_2] for $r = 0.8$.

Case 2: $1 < r < 3$**.** The extinction fixed point $N^* = 0$ is unstable, but the carrying capacity fixed point $N^* = (r - 1)/r$ is stable. We can conclude that the solution will approach the carrying capacity for most initial conditions. This was shown in figure [fig:sol_logistic_1] for $r = 1.5$ and is illustrated in figure [fig:sol_logistic_3] for $r = 2.8$. Notice that although the solution approaches the carrying capacity equilibrium in both cases, when $r > 2$, the solution oscillates while converging to its asymptotic value, foreshadowing the behavior when $r > 3$.

Case 3: $r > 3$**.** Strange things happen: there are no stable fixed points, so there is no value for the solution to approach. As we saw in the previous section, the solution can undergo so-called period two oscillations, which are shown in figure [fig:sol_logistic_4] with $r = 3.3$.

However, even stranger behavior is observed when the parameter r crosses the threshold of about 3.59. Figure [fig:sol_logistic_5] shows the behavior of the solution for $r = 3.6$, which is no longer periodic, and instead seems to bounce around without any discernible pattern. This dynamics is known as *chaos*.

4 Graphical analysis of difference equations

In addition to calculating numeric solutions, computers can be used to perform *graphical analysis* of discrete time models. A lot of information can be gleaned by plotting the graph of the updating function of an recurrent difference equation $x_{t+1} = f(x_t)$. Here is a summary of what we can learn from the graph of the function $f(x)$:

Information from graphs of updating functions

1. The location of the fixed points of the iterated map. Since the condition for a fixed point is $f(x) = x$, they can be found at the intersections of the graph of $y = f(x)$ and $y = x$ (the identity straight line).

2. The stability of fixed points. We learned that the derivative of $f(x)$ at a fixed point determines its stability. Graphically, this means that the slope of $f(x)$ at the point of intersection with $y = x$ can be used for this purpose; if it is steeper (in absolute value) than the straight line $y = x$, then the fixed point is unstable, but if its slope is less than one in absolute value, the equilibrium is stable.

3. Graphical iteration of the difference equation. The value of the function $f(x)$ gives the value of x at the next time step, and this fact can be used to produce a graph of successive values of the dependent variable: $x_0, x_1, x_2, ...$

Below we will demonstrate how to plot the updating function in Python and how to perform this analysis.

4.0.1 plot of the updating function

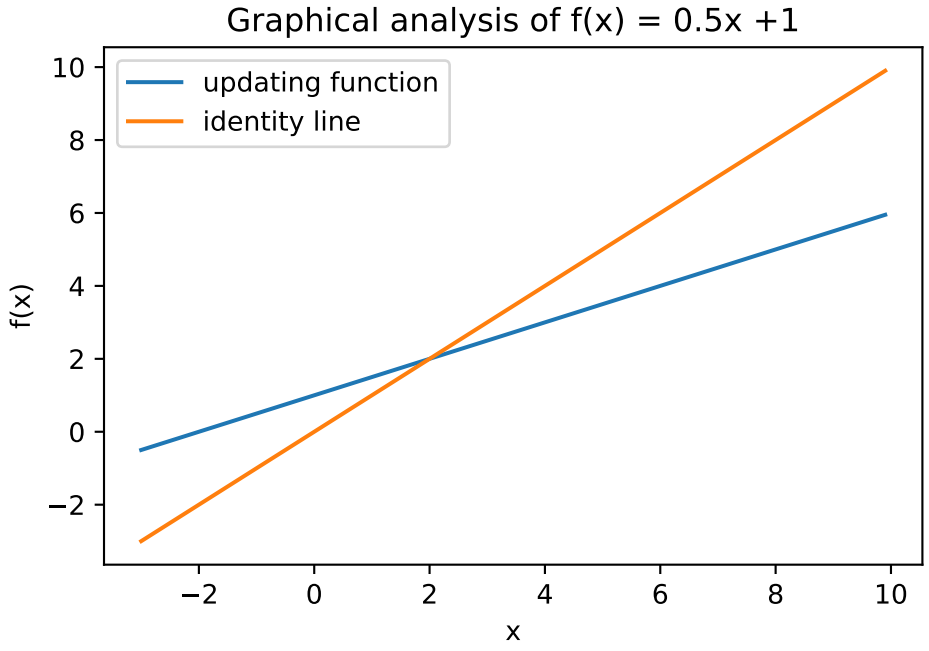
Let us consider a linear discrete-time model:

$$x_{t+1} = 5x_t - 10$$

Instead of iterating this equation starting from a particular initial value to produce a sequence of values, we will plot the function $f(x) = 0.5x - 10$ and use it to predict how solutions behave starting from *any* initial value. To do this, we will plot this function over a range of x values, along the the line $y = x$.

```
# Import packages
import numpy as np # package for work with arrays and matrices
import matplotlib.pyplot as plt # package with plotting capabilities
```

```
x = np.arange(-3,10,0.1) # range of x values
fx = 0.5*x + 1 # values of the updating function
plt.plot(x, fx, label = 'updating function') # plot the updating function
plt.plot(x, x, label = 'identity line')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('Graphical analysis of f(x) = 0.5x +1')
plt.legend()
plt.show()
```



The plot show thats the updating function intersects the identity line at $x = 2$, which is the (only) fixed point of this dynamic model. But it is stable or unstable? And how will solutions behave?

4.0.2 cobweb plot

Let us exploit the idea in the third point for graphical analysis of an iterated map. Starting with some initial condition x_0, the value of x_1 is given by $f(x_0)$. To show this graphically, starting the point x_0 on the axis, draw a vertical line to $y = f(x_0)$. Next, draw a horizontal line to the graph of $y = x$. Since the y and x coordinates are equal, we now have the value of $x_1 = f(x_0)$ as the x coordinate. Then, repeat the process by drawing a vertical line to $y = f(x_1)$, and the a horizontal line $y = x$, etc. The resulting sequence of x coordinates is a quick way of assessing the dynamics of the iterated map. For instance, the values may converge to a fixed point, or grow to infinity, or bounce around without settling down. The resulting graph of alternating vertical and horizontal line segments is called a cobweb plot:

Cobweb plot pseudocode

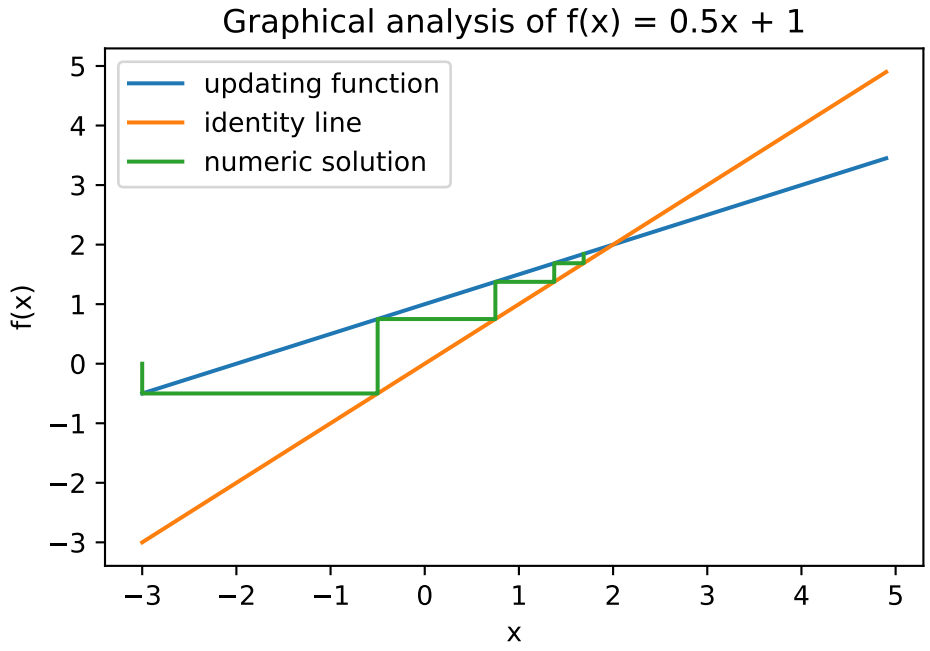
- define the updating function $f(x)$
- plot the graph of $f(x)$
- plot the identity line $y = x$
- set n to be the number of steps
- initialize an array x of length 2*n
- initialize an array y of length 2*n
- set $x[0]$ to the initial value
- set $y[0]$ to 0
- for n steps repeat (with i increasing by 2):
  - set $x[i+1] \leftarrow x[i]$
  - set $y[i+1] \leftarrow f(x[i])$
  - set $x[i+2] \leftarrow y[i+1]$
  - set $y[i+2] \leftarrow y[i+1]$
- plot the sequence of points (x, y) on the same plot

Here is an implementation of the cobweb plot for the same linear model as above.

```
x = np.arange(-3,5,0.1) # range of x values
fx = 0.5*x + 1 # values of the updating function
plt.plot(x, fx, label = 'updating function') # plot the updating function
plt.plot(x, x, label = 'identity line')

# the cobweb plot script
n = 5 # number of steps
x = np.zeros(n*2)
y = np.zeros(n*2)
x[0] = -3
```

```
print(x)
print(np.arange(0,2*n,2))
for i in np.arange(0,2*(n-1),2):
    x[i+1] = x[i] # keep the same x coordinate
    y[i+1] = 0.5*x[i] + 1 # the updating function
    x[i+2] = y[i+1] # move to the next x value
    y[i+2] = y[i+1] # keep the same y coordinate
x[2*n-1] = x[2*n-2] # finish the last half-iteration
y[2*n-1] = 0.5*x[2*n-2] + 1 # finish the last half-iteration

plt.plot(x,y, label = 'numeric solution')
plt.xlabel('x')
plt.ylabel('f(x)')
plt.title('Graphical analysis of f(x) = 0.5x + 1')
plt.legend()
plt.show()
```

```
[-3.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
[0 2 4 6 8]
```



You can see that the solution, shown in green, start from initial value of -3 and steps upward

toward the fixed point at 2. You can experiment by trying a different initial value, say 10 (you will have to change the domain of the plotted function) but you should still observe convergence to the fixed point at 2.

This could be predicted from the analysis performed in the previous section: the slope of the updating function is 0.5 (everywhere, since it's a straight line) and since that is less than 1 in absolute value, the fixed point is stable, and solutions are expected to converge to it.

We now have at our disposal analytical, numerical, and graphical tools to analyze and predict the behavior of a dynamical system. In the next section we will use all three to analyze a more complex model of population growth.

4.1 Graphical analysis of the logistic model

As we saw, we can learn a lot about the behavior of a dynamical system from analyzing the graph of the defining function. Let us consider two quadratic functions for the logistic model: $f(N) = 2N(1 - N/2)$ and $f(N) = 4N(1 - N/4)$.

First, plotting the graphs of $y = f(N)$ and $y = N$, allows us to find the fixed points of the logistic model. Since it is a , we see that there are fixed points at $N = 0$ for both functions, and carrying capacity sizes at $N = 2$ and $N = 3$, respectively. The reader should check that this is in agreement with the analytic prediction of $N^* = (r - 1)/r$.

Second, we can obtain information about stability of the two fixed points by considering the slope of the curve $y = f(N)$ at the points where it crosses $y = N$. On the graph of the first function, the slope is clearly 0, which indicates that the fixed point is stable, in agreement with the analytical prediction. On the graph of the second function, the slope is negative and steeper than -1. This indicates that the fixed point is unstable, again consistent with our analysis above.

Third, we graph a few iterations of the cobweb plot to obtain an idea about the dynamics of the population over time. As expected, for the first function with $r = 2$, the solution quickly approaches the carrying capacity ({numref. In the second function, however, $r = 4$ and the carrying capacity is unstable. In {numref}`fig-cobweb2` we observe a wild pattern of jumps that never approach any particular value.

We have seen how graphical tools can be used to analyze and predict the behavior of a dynamical system. In the case of the logistic model, we never found the analytic solution, because it frequently does not exist as a formula. Finding the fixed points and analyzing their stability, in conjunction with looking at the behavior of a cobweb plot, allowed us to describe the dynamics of population growth in the logistic model, without doing any "mathematics". Together, the analytical and graphical analysis provide complementary tools for biological modelers.

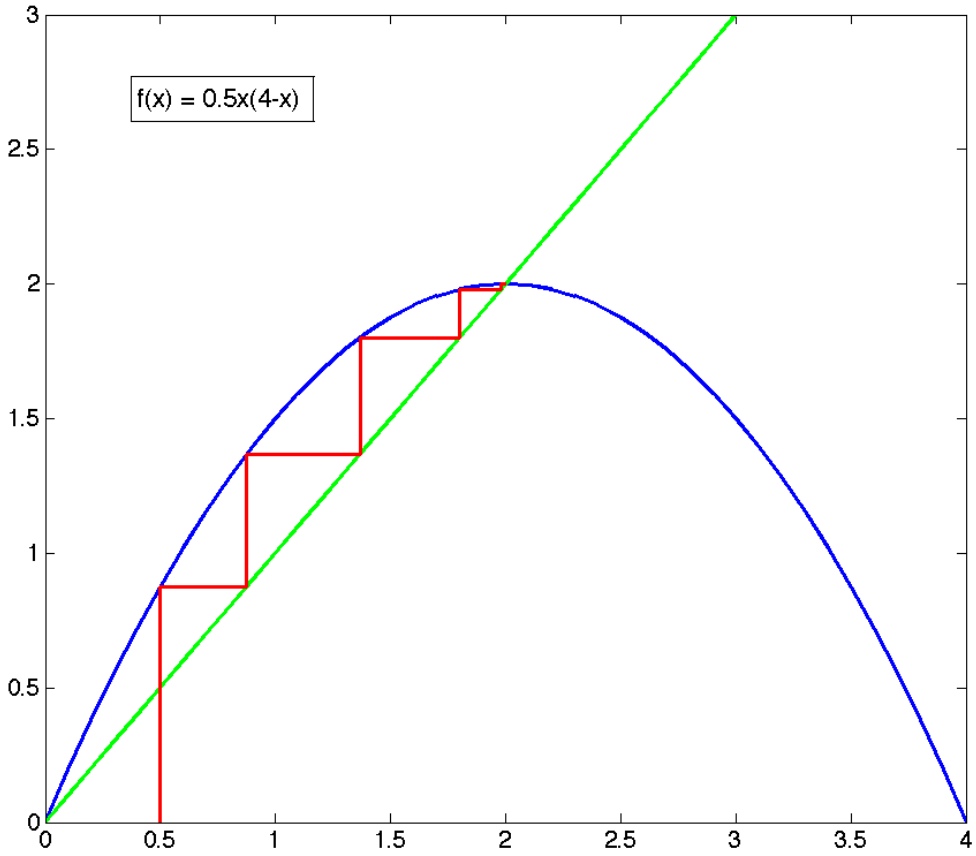


Figure 4.1: Cobweb plot of the logistic model with $r = 2$, showing a solution converging to the stable fixed point at the intersection of the graphs of the function and the identity line

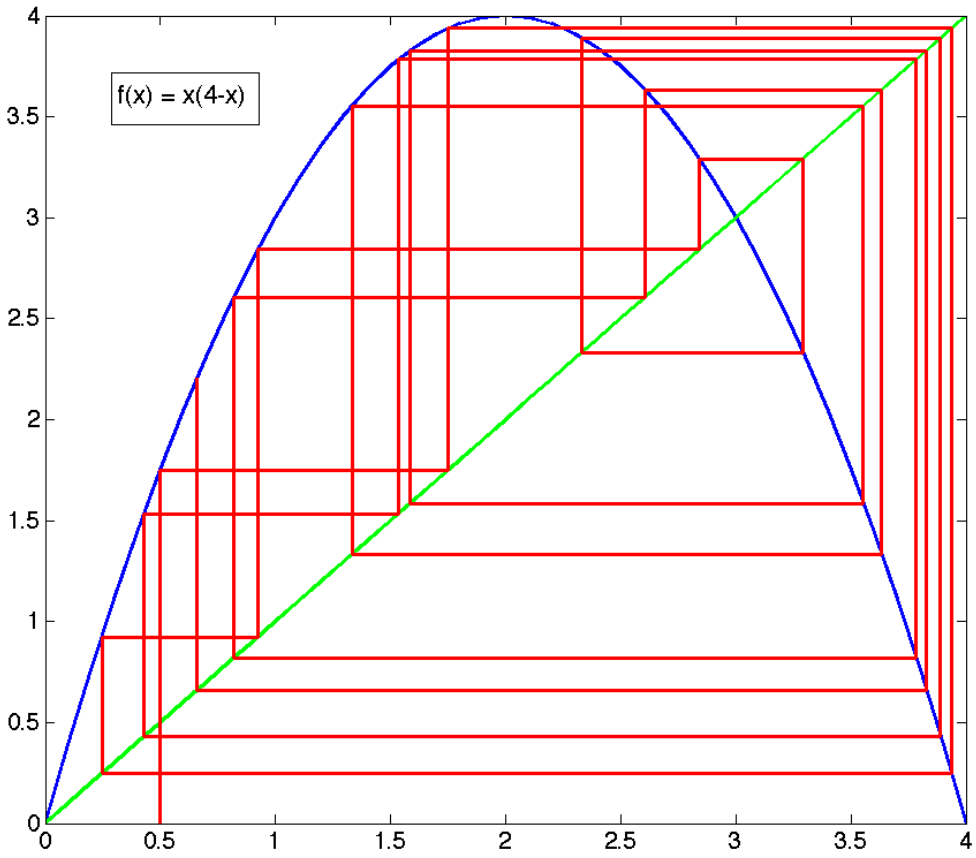


Figure 4.2: Cobweb plot of the logistic model with $r = 4$, showing a as solution bouncing around the unstable fixed point

4.1.1 chaos in discrete dynamical systems

In this chapter we learned to analyze the dynamics of solutions of nonlinear discrete-time dynamical systems without solving them on paper. In the last two sections we focused on the logistic difference equation as a simple nonlinear model with a rich array of dynamic behaviors. In this section we will summarize the analysis and draw conclusions for difference equation models in biology. This behavior was brought to the attention of biologists by John Maynard Smith (**smith_mathematical_1968?**) and Robert May (**may_bifurcations_1976?**).

Why does the logistic model behave so strangely in the second example above? We can use numerical simulations to plot the long-term solutions for the dependent variable for a range of parameter values, let us say between $2.5 < r < 4$. Then we plot the values to which the simulation converged (whether it is one, two, or many) on the y-axis, and the value of the parameter r on the x-axis. The resulting *bifurcation diagram* in shown in {numref}`fig-log-bifur`. The value of the parameter r is plotted on the horizontal axis, and the set of values that the dependent variable takes in the long run is shown on the vertical axis. There is only one stable fixed point for $r < 3$, then we see a 2-cycle appear for $3 < r < 3.45$. For values of r greater than about 3.45, a series of period-doubling bifurcations occur with shorter and shorter intervals of r. This is called a *period-doubling cascade*, which culminates at the value of $r \approx 3.57$, where the number of points in the cycle becomes essentially infinite. The sequence of values of r at which period-doubling occurs is approximately:

- period 2; $r_1 = 3$
- period 4; $r_2 \approx 3.449$
- period 8; $r_3 \approx 3.544$
- period 16; $r_4 \approx 3.564$
- period 32; $r_5 \approx 3.569$
- period ∞ (chaos); $r_\infty \approx 3.570$

For $r > r_\infty$, we observe a remarkable behavior found only in nonlinear dynamical systems, called *chaos*. Chaos is characterized by two qualities:

Characteristics of chaos

1. **Aperiodic behavior**: the dependent variable never repeats a value exactly, instead bouncing around an infinite set of values for all time
2. **Sensitive dependence on initial conditions**: no matter how close two initial conditions in a chaotic system, given enough time the two trajectories will diverge and lose any resemblance

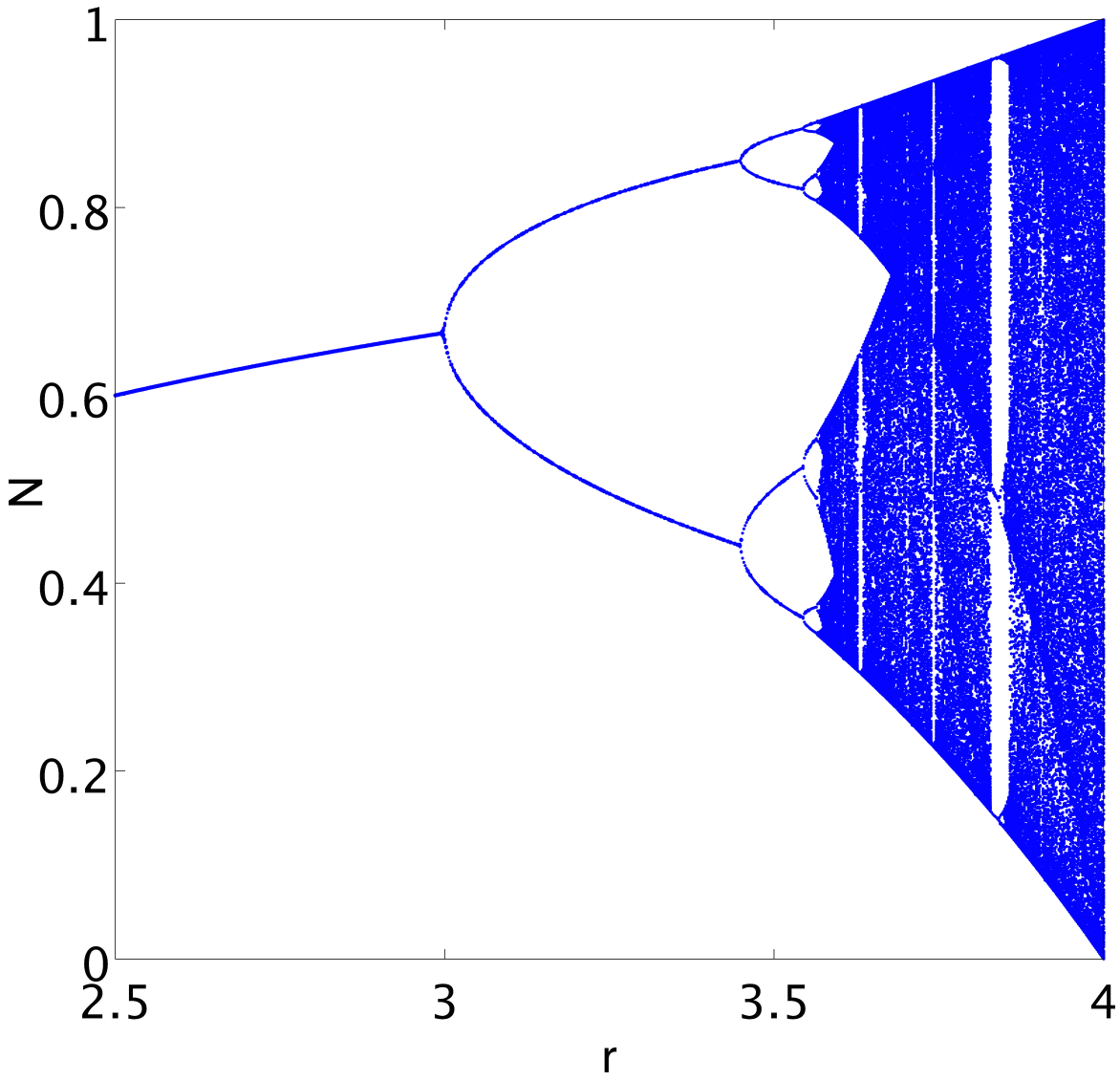


Figure 4.3: Bifurcation diagram for the logistic map $N(t+1) = r(1-N(t))N(t)$, with the parameter r on the horizontal axis and the vertical axis showing the values of the stable fixed point (for $r < 3$), then the values of the period two oscillation, the period four oscillation, etc., and for r greater than the critical value shows some of the values the solution chaotically jumps through.

What is especially surprising about chaos is that for a given initial condition a chaotic model gives a completely predictable and reproducible sequence of values of the dependent variable. However, given finite machine precision, or any error in initial conditions, a chaotic system is practically unpredictable and irreproducible. But there is a fundamental difference between deterministic chaos and a stochastic system, e.g. the model of coin tosses where knowing the previous result of the coin flip does not allow us to predict the next result, even under ideal conditions.

Notice in figure , that for $r > r_\infty$, chaotic behavior is observed only for some values of r. As you can see in figure , there are "bands of periodicity", where the attractor is a sequence of n numbers (and periods of 3,5, etc. are observed), alternating with bands of chaos. This illustrates that even the simplest nonlinear discrete dynamical systems can have incredibly complex behavior. When these results were first published by May in the 1970s, they revolutionized both the mathematical understanding of dynamical systems, and the field of theoretical biology. In one-variable dynamic systens, chaos occurs only in discrete-time dynamical models, but for three or more variables continuous time (ODE) dynamical systems also can behavechaotically.

As a mathematical side-note, if one looks at the differences between successive values of r_n, they behave like a geometric sequence, getting smaller and smaller by a constant fraction:

$$\delta_n = \frac{r_n - r_{n-1}}{r_{n+1} - r_n}$$

It is a remarkable fact that δ_n approaches a constant value when n gets large, 4.6692..., known as the Feigenbaum constant. It can be proven that this constant is the same for other iterated maps with the same shape as the downward parabola of the logistic map (e.g. $f(x) = \sin(x)$). Explaining why this deep mathematical fact is true is far outside the bounds of this course. (**strogatz_nonlinear_2001?**)

Chaos was a popular topic back in the 1980s and 90s, and even inspired popular books (**gleick_chaos:_1988?**). It is in fact remarkable that very simple difference equations can have solutions of apparently great complexity. This is intriguing because it appeals to a fairly universal human desire for simple explanations for complicated phenomena. The popular exposure to what was dubbed "chaos theory" (which is not an actual mathematical topic) spawned some inaccurate cliches, such as "a butterfly flapping its wings in South America can cause a hurricane to form and hit Florida". The image refers to the phenomenon of sensitive dependence on initial conditions, but of course it is utterly ridiculous to draw a causal arrow between a butterfly (one of an enumerable number of things changing the "initial conditions") and large-scale atmospheric phenomena. While there is some evidence that weather patters are complex systems that exhibit chaotic behavior, we lack the ability to isolate and control all influences that may perturb it, so pinning it on a butterfly is pretty unfair.

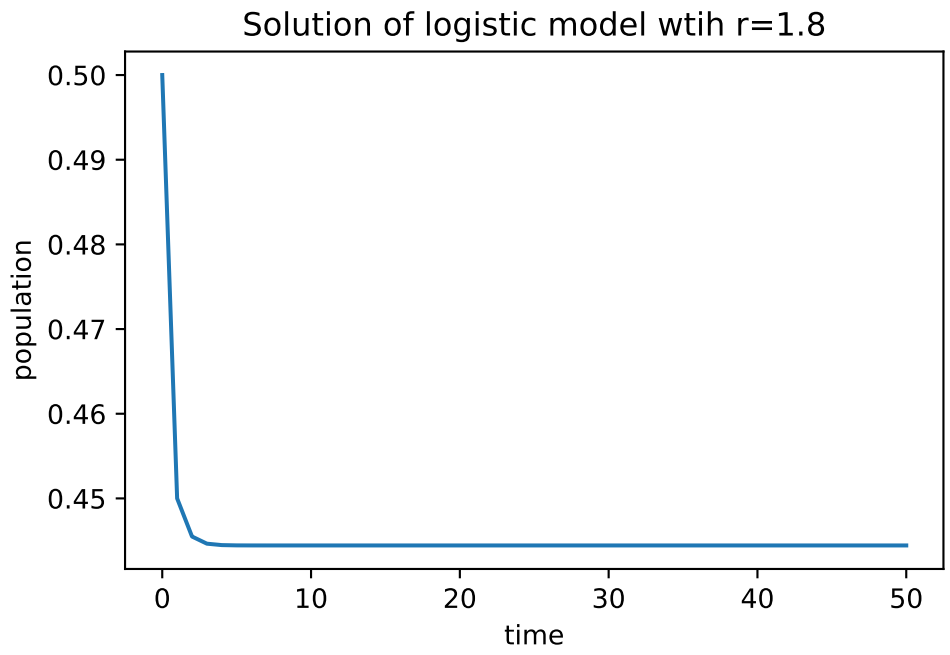
Despite the initial flurry of excitement, so-called chaos theory has failed to make a big impact on our understanding of complex biological systems. Although it is still quite fascinating

intellectually, a simple model like the logistic model is not an adequate model for any realistic population, particularly for large values of r where the chaotic behavior occurs. We now appreciate that the essential complexity of biological system requires multiple interacting variables which cannot be reduced to a single equation. However, there has been some successful observation of chaotic behavior in a population of flour beetles, which seemed to agree with predictions of a three-variable difference equation model (**costantino_chaotic_1997?**).

We have seen how graphical tools can be used to analyze and predict the behavior of a discrete-time dynamical system. We investigated the logistic model by finding the fixed points and analyzing their stability. Together with analysis of the graph of the updating function and making a cobweb plot, this allowed us to describe the dynamics of population growth in the logistic model, without doing any "math". Together, analytical and graphical analysis provide powerful tools for biological modelers.

Q3.1: For the logistic model with an initial population of 0.5 and $r = 1.1$, compute the first 50 iterations using the same for loop iteration you used above and plot the solution against time.

```
numsteps = 50 #set number of iterations
r = 1.8 #set parameter
N = np.zeros(numsteps+1) #initialize solution vector
N[0]=.5 #initial value
t = range(numsteps+1) #initialze time vector
a = -10
for i in range(numsteps):
    N[i+1] = r*N[i]*(1-N[i]) #logistic population model
plt.plot(t,N) #plot solution
plt.xlabel('time')
plt.ylabel('population')
plt.title('Solution of logistic model wtih r='+str(r))
plt.show()
```

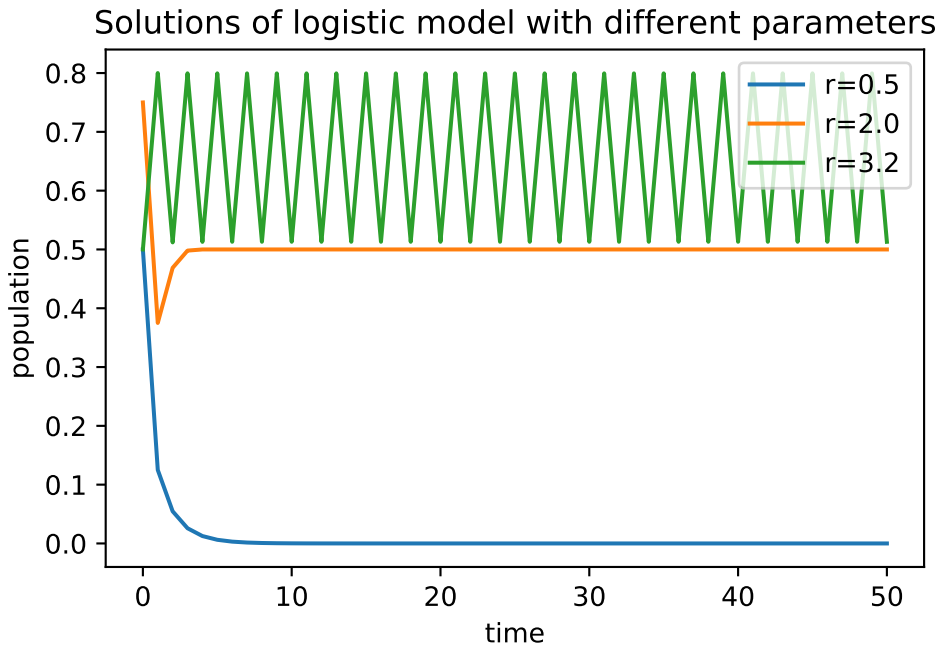


Q3.2: Change the parameter r to the following values: 0.5, 2.0, and 3.2, and in each case plot the solutions against time in separate figures. Describe each plot with a sentence.

```
numsteps = 50 #set number of iterations
r = .5 #set parameter
N = np.zeros(numsteps+1) #initialize solution vector
N[0]=.5 #initial value
t = range(numsteps+1) #initialze time vector
a = -10
for i in range(numsteps):
    N[i+1] = r*N[i]*(1-N[i]) #linear population model
plt.plot(t,N, label = 'r='+str(r)) #plot solution


r = 2.0 #set parameter
N = np.zeros(numsteps+1) #initialize solution vector
N[0]=0.75 #initial value
t = range(numsteps+1) #initialze time vector
a = -10
for i in range(numsteps):
    N[i+1] = r*N[i]*(1-N[i]) #linear population model
plt.plot(t,N, label = 'r='+str(r)) #plot solution
```

```
r = 3.2 #set parameter
N = np.zeros(numsteps+1) #initialize solution vector
N[0]=.5 #initial value
t = range(numsteps+1) #initialze time vector
a = -10
for i in range(numsteps):
    N[i+1] = r*N[i]*(1-N[i]) #linear population model
print(N)
plt.plot(t,N, label = 'r='+str(r)) #plot solution
plt.xlabel('time')
plt.ylabel('population')
plt.title('Solutions of logistic model with different parameters')
plt.legend()
plt.show()
```

```
[0.5        0.8        0.512      0.7995392  0.51288406 0.7994688
 0.51301899 0.79945762 0.51304043 0.79945583 0.51304386 0.79945554
 0.51304441 0.7994555  0.51304449 0.79945549 0.51304451 0.79945549
 0.51304451 0.79945549 0.51304451 0.79945549 0.51304451 0.79945549
 0.51304451 0.79945549 0.51304451 0.79945549 0.51304451 0.79945549
 0.51304451 0.79945549 0.51304451 0.79945549 0.51304451 0.79945549
 0.51304451 0.79945549 0.51304451 0.79945549 0.51304451 0.79945549
 0.51304451 0.79945549 0.51304451]
```

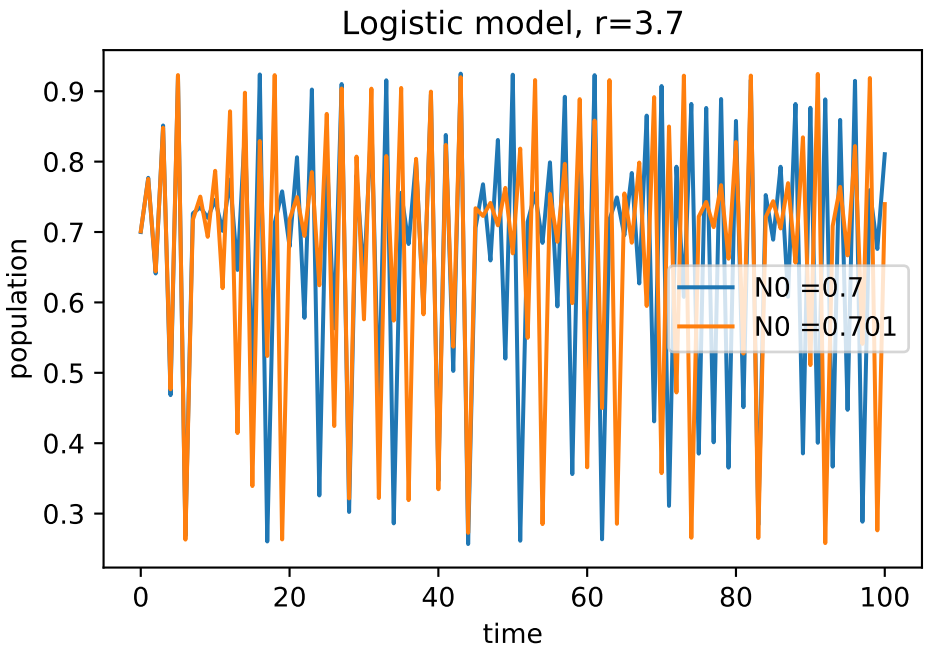


- The solution for r=0.5 decreases to zero
- The solution for r=2.0 stays at the fixed point of 0.5
- The solution for r=3.2 oscillates between two values indefinitely

Increase the parameter r further until you see strange, aperiodic behavior called chaos. Report at least one value of r at which you see chaotic dynamics.

```
numsteps = 100 #set number of iterations
r = 3.7 #set parameter
N = np.zeros(numsteps+1) #initialize solution vector
N[0]=.7 #initial value
t = range(numsteps+1) #initialze time vector
a = -10
for i in range(numsteps):
    N[i+1] = r*N[i]*(1-N[i]) #linear population model
plt.plot(t, N, label = 'N0 =' + str(N[0]))

numsteps = 100 #set number of iterations
r = 3.7 #set parameter
N = np.zeros(numsteps+1) #initialize solution vector
N[0]=.701 #initial value
t = range(numsteps+1) #initialze time vector
a = -10
```

```
for i in range(numsteps):
    N[i+1] = r*N[i]*(1-N[i]) #linear population model
plt.plot(t,N, label = 'N0 =' + str(N[0])) #plot solution

plt.xlabel('time')
plt.ylabel('population')
plt.title('Logistic model, r='+str(r))
plt.legend()
plt.show()
```



At $r = 3.7$ the solution bounces around without any apparent pattern, which is called chaos.

5 Discrete models of higher order

It is not unusual for biological systems to have multiple variables which influence each other, and thus need to be accounted in any model that aims to be useful. In this unit we will learn how to construct such models, and the methods for analyzing, solving, and numerically simulating them. We will see how models with two or more variables are used in a variety of biological fields: to describe population demographics, motility of cochlear cells, psychology of human relationships, gene regulation, and motion of molecular structures.

We will need new mathematical tools in order to analyze models with multiple variables. These methods are primarily from the realm of linear algebra. We will express multiple equations in terms of matrices and vectors, and learn how to operate on these objects. The dynamics of these models can be analyzed by doing calculations with the matrices, specifically finding special numbers and vectors known as eigenvalues and eigenvectors. These concepts, which will be introduced later, are absolutely central to all of applied mathematics, and to computational biology in particular.

In this chapter, the section on modeling is devoted to an old model of a population where individuals live for two generations, known as the Fibonacci model. We then describe how this model can be written down either as a single difference equation of second order, or as two equation of the first order, which may be represented in matrix form. We will learn to solve second order difference equations with an explicit formula, and then introduce some elementary matrix operations. In the computational section we will use the matrix notation to compute numerical solutions for higher order difference equations. Finally, in the synthesis section we will analyze two demographic population models, in which the population is broken into age groups. The matrix notation will be important for concisely representing different parameters for each age group.

In this chapter you will learn to:

- build higher order population models
- express age-structured models in matrix form
- analyze solutions of these models on paper
- use Python for matrix operations
- classify the behavior of solutions of these models

5.1 higher order difference equations

So far we have dealt with difference equations in which the value of the dependent variable at the next time step x_{t+1} depends solely on the variable at the present time x_t. These are known as *first order* difference equations because they only require one step from the present to the future. We will now examine difference equations where the future value depends not only on the present value x_t, but also on the past values: x_{t-1}, etc. The number of time steps that the equation looks into the past is the the order of the scheme.

5.1.1 the Fibonacci model and sequence

The Italian mathematician Leonardo Fibonacci, who lived in the late 12th - early 13th centuries, contributed greatly to the development of mathematics in the western world. For starters, he introduced the Hindu-Arabic numerals we use today, in place of the cumbersome Roman numerals. He also constructed an early model of population growth, which considered a population of individuals that lived for two generations. The first generation does not reproduce, but in the second generation each individual produces a single offspring (or each pair produces a new pair) and then dies. Then the total number of individuals at the next time step is the sum of the individuals in the previous two time steps ({numref}`fig-fib-rabbits`) This is described by the following second order difference equation:

$$N_{t+1} = N_t + N_{t-1}$$

(fibonacci)

The famous Fibonacci sequence is a solution of this equation. For a second-order equations, two initial conditions are required, and if we take $N_0 = 0$ and $N_1 = 1$, then the resulting sequence will look as follows:

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, ...$$

The Fibonacci sequence is famously found in many natural phenomena, including in phyllotaxis (arrangement of parts in plants), and spirals on some mollusk shells, e.g. *Nautilus pompilius* ({numref}`fig-fib-rabbits`). It may be observed by counting the number of spirals that can be drawn between plant units (such as seeds or petals), and observing that alternating the right-handed and left-handed spirals, while moving away from the center, often results in the Fibonacci sequence. The precise reason for this is unclear, although explanations exist, for instance that this pattern provides the most efficient packing of seeds.

[The shell of the *Nautilus pompilius* mollusk has the shape of a Fibonacci spiral, shown here filled with squares of the corresponding size http://mathforum.org/mathimages] (images/fibo_nautilus.jpg)

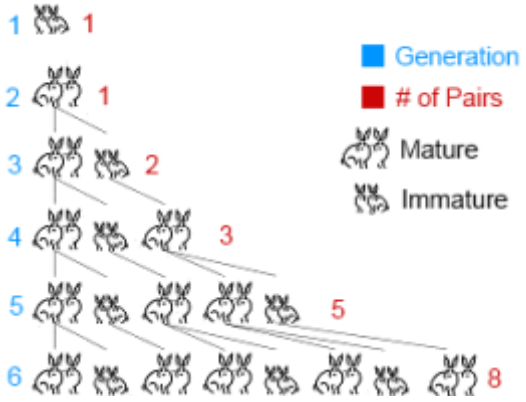


Figure 5.1: The Fibbonaci model with each pair of individuals waiting one generation before producing another pair each subsequent generation <https://artblot.wordpress.com/2013/05/10/rich-with-fibonacci-gold/

5.1.2 matrix representation of discrete time models

The Fibonacci model above can be represented by two equations instead of one, if we consider two dependent variables. Let us represent the number of rabbits in generation 1 (young) by x and in generation 2 (old) by$y$. The new generation at the next time $(t+1)$ is comprised of offspring of the young and old generations at time t, while the old generation at the next time is simply is young generation at the current time. This gives the following set of equations:

$$\begin{aligned} x_{t+1} &= x_t + x_{t-1} \\ x_t &= x_t \end{aligned}$$

The advantage of re-writing a single equation as two is that the new system is first order, that is, only relies on the values of the variables at the current time t. These equations can also be written in *matrix* form:

$$\begin{pmatrix} x_{t+1} \\ x_t \end{pmatrix} = \begin{pmatrix} x_t + x_{t-1} \\ x_t \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} x_t \\ x_{t-1} \end{pmatrix}$$

This representation is convenient and leads to a set of rules for matrix manipulation. We wrote the write-hand side as a product of a matrix containing the coefficients of x_t and y_t and the vector with the two variables. The product of the matrix and the vector is equal to the original vector.

5.2 Solutions for linear higher order difference equations

5.2.1 solutions of linear difference equations

Solutions for first order linear difference equations are exponential in form. The solutions for second order linear difference equations consist of a sum of two exponentials with different bases. For the following general linear second order difference equation:

$$x_{t+1} = ax_t + bx_{t-1}$$

The solution can be written as follows:

$$x_t = A\lambda_1^t + B\lambda_2^t$$

The solution for a second order difference equation is a sum of two terms that look like solutions to first order difference equations. There are two different types of constants in the solution: the bases of the exponential λ_1, λ_2 and the multiplicative constants A and B. They are different because the exponential parameters depend on the equation itself, but not on the initial conditions, while the multiplicative constants depend only on the initial conditions. Therefore, they can be determined separately:

Outline for solving a second order linear difference equation

1. Substitute the solution $x_t = \lambda^t$ into the difference equation. For the general difference equation, we obtain a the following quadratic relation by dividing everything by λ^{t-1}:
$$\lambda^{t+1} = a\lambda^t + b\lambda^{t-1} \Rightarrow \lambda^2 = a\lambda + b$$

2. Solve the quadratic equation for values of λ which satisfy the difference equation:

$$\lambda_{1,2} = \frac{a \pm \sqrt{a^2 + 4b}}{2}$$

If $a^2 + 4b > 0$, this gives two values of λ; if $a^2 + 4b = 0$, there is a single value, and if $a^2 + 4b < 0$, then no real values of λ satisfy the difference equation.

3. Once we have found the values λ_1 and λ_2, use the initial conditions (e.g. some values x_0 and x_1) to solve for the multiplicative constants:

$$x_0 = A + B;\ x_1 = A\lambda_1 + B\lambda_2$$

Use $A = x_0 - B$ to plug into the second equation: $x_1 = (x_0 - B)\lambda_1 + B\lambda_2$

4. The general solution for A and B is the following, provided $\lambda_2 \neq \lambda_1$:

$$B = \frac{x_1 - x_0\lambda_1}{\lambda_2 - \lambda_1};\ A = \frac{x_0\lambda_2 - x_1}{\lambda_2 - \lambda_1}$$

Let us apply this approach to solving the Fibonacci difference equation {eq}`fibonacci`:

$$\lambda^2 = \lambda + 1 \implies \lambda^2 - \lambda - 1 = 0$$

We find the solutions by the quadratic formula: $\lambda_{1,2} = (1 \pm \sqrt{5})/2$.

Now let us use the initial conditions $N_0 = 0$ and $N_1 = 1$. The two multiplicative constants must then satisfy the following:

$$0 = A + B;\ 1 = A\lambda_1 + B\lambda_2$$

By the formula we found above, the initial conditions are:

$$A = \frac{-1}{\lambda_2 - \lambda_1} = \frac{1}{\sqrt{5}};\ B = \frac{1}{\lambda_2 - \lambda_1} = \frac{-1}{\sqrt{5}}$$

The complete solution, which gives the t-th number in the Fibonacci sequence is:

$$N_t = \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^t - \frac{1}{\sqrt{5}}\left(\frac{1-\sqrt{5}}{2}\right)^t$$

There are several remarkable things about this formula. First is the fact that despite the abundance of irrational numbers, for each integer t the number N_t is an integer. One can check this by programming the formula in your favorite language, and plugging in any value of t.

Second, an important feature of the Fibonacci sequence is the ratio between successive terms in the sequence. Notice that of the two terms in the formula, $(\frac{1+\sqrt{5}}{2})^t$ grows as t increases, while $(\frac{1-\sqrt{5}}{2})^t$ decreases to zero, because the first number is greater than 1, while the second is less than 1. This means that for large t, the terms in the Fibonacci sequence are approximately equal to:

$$N_t \approx \frac{1}{\sqrt{5}}\left(\frac{1+\sqrt{5}}{2}\right)^t$$

Since each successive term is multiplied by $(1+\sqrt{5})/2$, the ratio between successive terms, $\phi = N_{t+1}/N_t$ approaches the value $\phi = (1+\sqrt{5})/2 \approx 1.618$ for increasing t.

This number $(1+\sqrt{5})/2$ is called the *golden ratio* or *golden section*, and was known from antiquity as the most aesthetically pleasing proportion in architecture and art, when used as a ratio between the height and width of the piece of art. Algebraically, the golden ratio is defined to be the number that is both the ratio between two quantities, e.g. a and b, and also the ratio between the sum of the two quantities $(a+b)$ and the larger of the quantities e.g. b ({numref}`fig-gold-ratio`). Geometrically, the golden ratio can be constructed as the ratio between two sides of a rectangle, a and b, which are also part of the larger rectangle with sides $a+b$ and a. This construction is shown in {numref}`fig-gold-rect`.

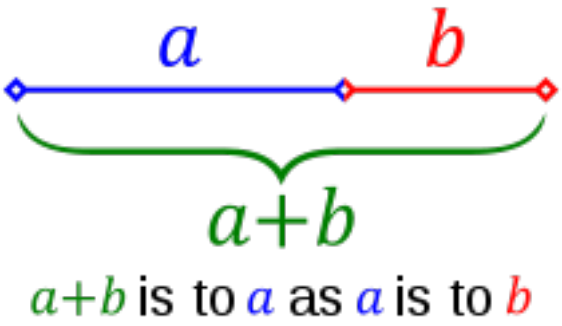


Figure 5.2: Line segments that are in golden proportion to each other http://en.wikipedia.org/wiki/Golden_ratio

To show that the geometric golden ratio is the same as the ratio that appears in the Fibonacci sequence, let us write down the algebraic condition stated above. Because we are interested in the ratio, let the smaller quantity be 1 and the larger one be ϕ; by the definition we obtain the following. $\phi = (\phi+1)/\phi$, thus $\phi^2 - \phi - 1 = 0$. This is the same quadratic equation that we derived for the exponential bases of the solution above. The solution to this equation (by the quadratic formula) is $\phi = (1 \pm \sqrt{5})/2$, and the positive solution is the golden ratio.

5.3 Matrices and vectors

One basic advantage of matrix notation is that it makes it possible to write any set of *linear equations* as a single matrix equation. By linear equations we mean those that contain only constants or first powers of the variables. The field of mathematics studying matrices and their generalizations is called *linear algebra*; it is fundamental to both pure and applied mathematics. In this section we will learn some basic facts about matrices and their properties.

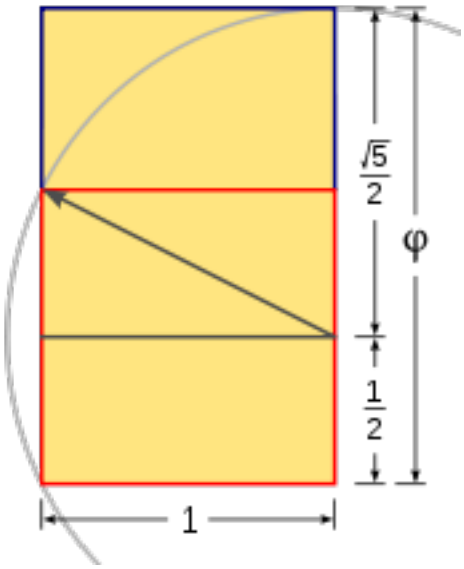


Figure 5.3: Construction of a rectangle with the golden ratio between its sides http://en.wikipedia.org/wiki/Golden_ratio

5.3.1 elementary matrix operations

Now is a good time to properly define what matrices are and how we can operate on them. We have already seen a matrix for the Fibonacci model, but just to make sure all of the terms are clear:

Definition

A matrix A is a rectangular array of *elements* A_{ij}, in which i denotes the row number (index), counted from the top, and j denotes the column number (index), counted from left to right. The *dimensions* of a matrix are defined by the number of rows and columns, so an n *by* m *matrix* contains n rows and m columns.

Definition

The elements of a matrix A which have the same row and column index, e.g. A_{33} are called the *diagonal elements*. Those which do not lie on the diagonal are called the *off-diagonal* elements.

For instance, in the 3 by 3 matrix below, the elements a, e, i are the diagonal elements:

$$A = \begin{pmatrix} a & b & c \\ d & e & f \\ g & h & i \end{pmatrix}$$

Matrices can be added together if they have the same dimensions. Then matrix addition is defined simply as adding up corresponding elements, for instance the element in the second row and first column of matrix A is added with the element in the second row and first column of matrix B to give the element in the second row and first column of matrix C. Recall from the previous chapter that rows in matrices are counted from top to bottom, while the columns are counted left to right.

5.3.2 matrix multiplication

Matrices can also be multiplied, but this operation is trickier. For mathematical reasons, multiplication of matrices $A \times B$ does not mean multiplying corresponding elements. Instead, the definition seeks to capture the calculation of simultaneous equations, like the one in the previous section. Here is the definition of matrix multiplication, in words and in a formula (**strang_linear_2005?**):

The *product of matrices A and B* is defined to be a matrix C, whose element c_{ij} is the **dot product of the i-th row of A and the j-th column of B**:

$$c_{ij} = a_{i1}b_{1j} + a_{i2}b_{2j} + ... + a_{iN}b_{Nj} = \sum_{k=1}^{q} a_{ik}b_{kj}$$

This definition is possible only if the length of the rows of A and the length of columns of B are the same, since we cannot compute the dot product of two vectors of different lengths. Matrix multiplication is defined only if A is n by q and B is q by m, for any integers n, q, and m and the resulting matrix C is a matrix with n rows and m columns. In other words, **the inner dimensions of matrices have to match** in order for matrix multiplication to be possible. This is illustrated in {numref}`fig-mat-mult`

Example. Let us multiply two matrices to illustrate how it's done. Here both matrices are 2 by 2, so their inner dimensions match and the resulting matrix is 2 by 2 as well:

$$\begin{pmatrix} 1 & 3 \\ 6 & 1 \end{pmatrix} \times \begin{pmatrix} 4 & 1 \\ 5 & -1 \end{pmatrix} = \begin{pmatrix} 1 \times 4 + 3 \times 5 & 1 \times 1 + 3 \times (-1) \\ 6 \times 4 + 1 \times 5 & 6 \times 1 + 1 \times (-1) \end{pmatrix} = \begin{pmatrix} 19 & -2 \\ 29 & 5 \end{pmatrix}$$

One important consequence of this definition is that **matrix multiplication is not commutative**. If you switch the order, e.g. $B \times A$, the resulting multiplication requires dot products

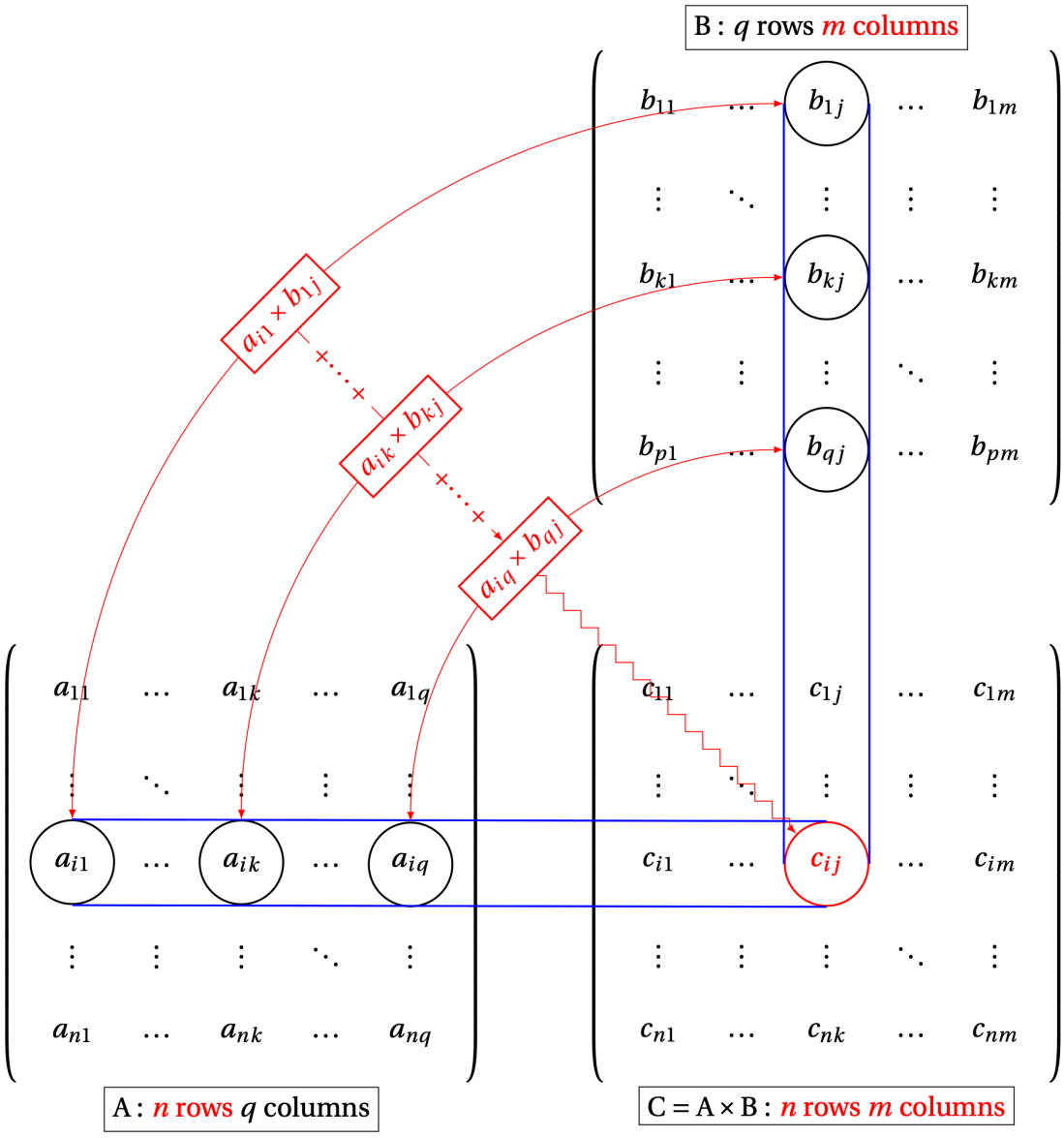


Figure 5.4: Multiplication of two matrices A and B results in a new matrix C

of the rows of B by the columns of A, and except in very special circumstances, they are not the same. In fact, unless m and n are the same integer, the product of $B \times A$ may not be defined at all.

In the example above of the matrix representation of the Fibonacci model, we implicitly used the conventional rules for multiplying matrices and vectors. Each row of the matrix

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$$

contains the numbers that multiply the two elements of the vector

$$\begin{pmatrix} x_t \\ x_{t-1} \end{pmatrix}$$

in order to generate the two equations $x_{t+1} = x_t + x_{t-1}$ and $x_t = x_t$.

Take the matrix equation for the Fibonacci difference equation above. Put the first two values 0 and 1 into the vector. Then perform the multiplication of the matrix and the vector:

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0+1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

We can take the resulting vector and apply the matrix again, to propagate the sequence for one more step:

$$\begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix} \times \begin{pmatrix} 1 \\ 1 \end{pmatrix} = \begin{pmatrix} 1+1 \\ 1 \end{pmatrix} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

Multiplying matrices and vectors is a basic operation that depends on the orientation of the vector. One can only multiply a square matrix by a column vector on the left, as we saw above, not on the right. By the same token, a row vector can only multiply a matrix on the right, and not the left, because we must use the *rows* of the matrix on the left to multiply the *columns* of the matrix on the right. This underscores the important fact that matrix multiplication is not commutative.

5.3.3 matrix inverses

Above we learned the rules of matrix multiplication, and we can write $C = A \times B$, so long as the number of columns in A matches the number of rows in B. However, what if we want to reverse the process? If we know the resulting matrix C, and one of the two matrices, e.g. A, how can we find B? Naively, we would like to be able to divide both sides by the matrix A, and find $B = C/A$. However, things are more complicated for matrices.

Properly speaking, we need to introduce the *inverse* of a matrix A. If we think about inverses of real numbers, a^{-1} is a number that when it multiplies a, results in one. In order to define the equivalent for matrices, we first need to introduce the unity of matrix multiplication.

> **Definition**
>
> The *identity* matrix is an n by n matrix that does not change another n by n matrix by multiplication:
>
> $$A \times I = I \times A = A$$
>
> The diagonal elements of the identity matrix are 1s and all off-diagonal elements are zero.

Example: Using the definition of matrix multiplication we can verify that this definition works for any 2 by 2 matrix:

$$\begin{pmatrix} -6 & -2 \\ 12 & -1 \end{pmatrix} \times \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} -6 \times 1 + -2 \times 0 & -6 \times 0 + -2 \times 1 \\ 12 \times 1 - 1 \times 0 & 12 \times 0 - 1 \times 1 \end{pmatrix} = \begin{pmatrix} -6 & -2 \\ 12 & -1 \end{pmatrix}$$

Now that we have specified the identity, we can define the matrix inverse:

> **Definition**
>
> A square matrix A has an *inverse matrix* A^{-1} if it satisfies the following:
>
> $$A^{-1} \times A = A \times A^{-1} = I$$

Finding the inverse of a matrix is not simple, and we will be content to let computers handle the dirty work. In fact, not every matrix possesses an inverse. There is a test for existence of an inverse of A, and it depends on the determinant (**strang_linear_2005?**):

A square matrix A possesses an inverse A^{-1} and is called *invertible* if and only if its determinant is not zero.

5.3.4 matrices transform vectors

In this section we will learn to characterize square matrices by finding special numbers and vectors associated with them. At the core of this analysis lies the concept of a matrix as an *operator* that transforms vectors by multiplication. To be clear, in this section we take as default that the matrices A are square, and that vectors $\vec{v}$ are column vectors, and thus will multiply the matrix on the right: $A \times \vec{v}$.

A matrix multiplied by a vector produces another vector, provided the number of columns in the matrix is the same as the number of rows in the vector. This can be interpreted as the matrix transforming the vector $\vec{v}$ into another one: $A \times \vec{v} = \vec{u}$. The resultant vector $\vec{u}$ may or may not resemble $\vec{v}$, but there are special vectors for which the transformation is very simple.

Example. Let us multiply the following matrix and vector (specially chosen to make a point):

$$\begin{pmatrix} 2 & 1 \\ 2 & 3 \end{pmatrix} \times \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 2-1 \\ 2-3 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

We see that this particular vector is unchanged when multiplied by this matrix, or we can say that the matrix multiplication is equivalent to multiplication by 1. Here is another such vector for the same matrix:

$$\begin{pmatrix} 2 & 1 \\ 2 & 3 \end{pmatrix} \times \begin{pmatrix} 1 \\ 2 \end{pmatrix} = \begin{pmatrix} 2+2 \\ 2+6 \end{pmatrix} = \begin{pmatrix} 4 \\ 8 \end{pmatrix}$$

In this case, the vector is changed, but only by multiplication by a constant (4). Thus the geometric direction of the vector remained unchanged.

Generally, a square matrix has an associated set of vectors for which multiplication by the matrix is equivalent to multiplication by a constant. This can be written down as a definition:

i Definition

An *eigenvector* of a square matrix A is a vector $\vec{v}$ for which matrix multiplication by A is equivalent to multiplication by a constant. This constant λ is called its *eigenvalue* of A corresponding the the eigenvector $\vec{v}$. The relationship is summarized in the following equation:

$$A \times \vec{v} = \lambda \vec{v}$$

(def-eigen)

Note that this equation combines a matrix (A), a vector ($\vec{v}$) and a scalar λ, and that both sides of the equation are column vectors. This definition is illustrated in {numref}`fig-eig-vec`, showing a vector (v) multiplied by a matrix A, and the resulting vector λv, which is in the same direction as v, due to scalar multiplying all elements of a vector, thus either stretching it if $\lambda > 1$ or compressing it if $\lambda < 1$. This assumes that λ is a real number, which is not always the case, but we will leave that complication aside for the purposes of this chapter.

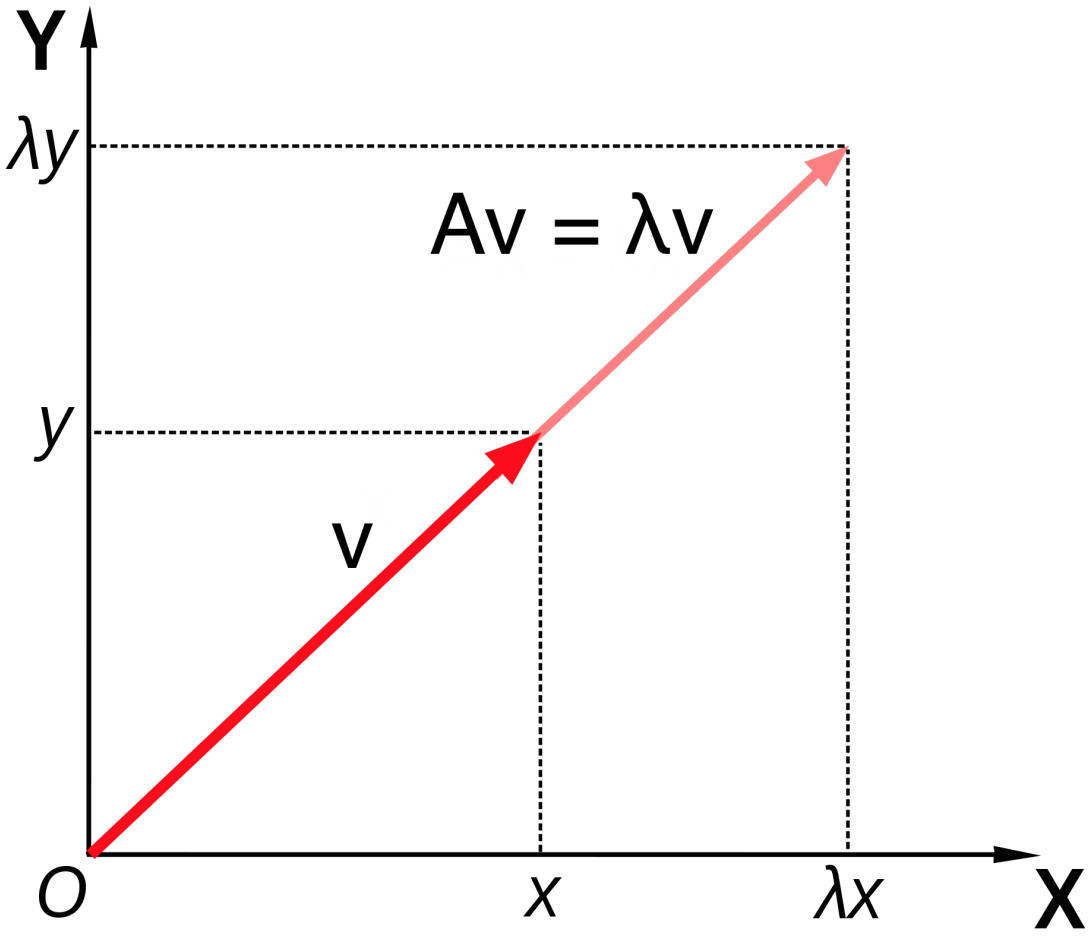


Figure 5.5: Illustration of the geometry of a matrix A multiplying its eigenvector v, resulting in a vector in the same direction λv. (Figure by Lantonov under CC BY-SA 4.0 via Wikimedia Commons)

The definition does not specify how many such eigenvectors and eigenvalues can exist for a given matrix A. There are usually as many such vectors $\vec{v}$ and corresponding numbers λ as the number of rows or columns of the square matrix A, so a 2 by 2 matrix has two eigenvectors and two eigenvalues, a 5x5 matrix has 5 of each, etc. One ironclad rule is that there cannot be more distinct eigenvalues than the matrix dimension. Some matrices possess fewer eigenvalues

than the matrix dimension, those are said to have a *degenerate* set of eigenvalues, and at least two of the eigenvectors share the same eigenvalue.

The situation with eigenvectors is trickier. There are some matrices for which any vector is an eigenvector, and others which have a limited set of eigenvectors. What is difficult about counting eigenvectors is that an eigenvector is still an eigenvector when multiplied by a constant. You can show that for any matrix, multiplication by a constant is commutative: $cA = Ac$, where A is a matrix and c is a constant. This leads us to the important result that if $\vec{v}$ is an eigenvector with eigenvalue λ, then any scalar multiple $c\vec{v}$ is also an eigenvector with the same eigenvalue. The following demonstrates this algebraically:

$$A \times (c\vec{v}) = cA \times \vec{v} = c\lambda\vec{v} = \lambda(c\vec{v})$$

This shows that when the vector $c\vec{v}$ is multiplied by the matrix A, it results in its being multiplied by the same number λ, so by definition it is an eigenvector. Therefore, an eigenvector $\vec{v}$ is not unique, as any constant multiple $c\vec{v}$ is also an eigenvector. It is more useful to think not of a single eigenvector $\vec{v}$, but of a **collection of vectors that can be interconverted by scalar multiplication** that are all essentially the same eigenvector. Another way to represent this, if the eigenvector is real, is that an eigenvector as a **direction that remains unchanged by multiplication by the matrix**, such as direction of the vector v in figure . As mentioned above, this is true only for real eigenvalues and eigenvectors, since complex eigenvectors cannot be used to define a direction in a real space.

To summarize, eigenvalues and eigenvectors of a matrix are a set of numbers and a set of vectors (up to scalar multiple) that describe the action of the matrix as a multiplicative operator on vectors. "Well-behaved" square n by n matrices have n distinct eigenvalues and n eigenvectors pointing in distinct directions. In a deep sense, the collection of eigenvectors and eigenvalues defines a matrix A, which is why an older name for them is characteristic vectors and values.

5.3.5 calculating eigenvalues

Finding the eigenvalues and eigenvectors analytically, that is on paper, is quite laborious even for 3 by 3 or 4 by 4 matrices and for larger ones there is no analytical solution. In practice, the task is outsourced to a computer. Nevertheless, it is useful to go through the process in 2 dimensions in order to gain an understanding of what is involved.

First, let us define two quantities that will be useful for this calculation:

> **i** Definition
>
> The *trace* τ of a matrix A is the sum of the diagonal elements: $\tau = \sum_i A_{ii}$

Definition

The *determinant* Δ of a 2x2 matrix A is given by the following: $\Delta = ad - bc$, where

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

For larger matrices, the determinant is defined recursively, in terms of 2x2 submatrices of the larger matrix, but we will not give the full definition here.

From the definition [def:eigen] of eigenvalues and eigenvectors, the condition can be written in terms of the four elements of a 2 by 2 matrix:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \times \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} av_1 + bv_2 \\ cv_1 + dv_2 \end{pmatrix} = \lambda \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

This is now a system of two linear algebraic equations, which we can solve by substitution. First, let us solve for v_1 in the first row, to get

$$v_1 = \frac{-bv_2}{a - \lambda}$$

Then we substitute this into the second equation and get:

$$\frac{-bcv_2}{a - \lambda} + (d - \lambda)v_2 = 0$$

Since v_2 multiplies both terms, and is not necessarily zero, we require that its multiplicative factor be zero. Doing a little algebra, we obtain the following, known as the *characteristic equation* of the matrix:

$$-bc + (a - \lambda)(d - \lambda) = \lambda^2 - (a + d)\lambda + ad - bc = 0$$

This equation can be simplified by using two quantities we defined at the beginning of the section: the sum of the diagonal elements called the trace $\tau = a + d$, and the determinant $\Delta = ad - bc$. The quadratic equation has two solutions, dependent solely on τ and Δ:

$$\lambda = \frac{\tau \pm \sqrt{\tau^2 - 4\Delta}}{2}$$

This is the general expression for a 2 by 2 matrix, showing there are two possible eigenvalues. Note that if $\tau^2 - 4\Delta > 0$, the eigenvalues are real, if $\tau^2 - 4\Delta < 0$, they are complex (have

real and imaginary parts), and if $\tau^2 - 4\Delta = 0$, there is only one eigenvalue. This situation is known as degenerate, because two eigenvectors share the same eigenvalue.

Example. Let us take the same matrix we looked at in the previous subsection:

$$A = \begin{pmatrix} 2 & 1 \\ 2 & 3 \end{pmatrix}$$

The trace of this matrix is $\tau = 2 + 3 = 5$ and the determinant is $\Delta = 6 - 2 = 4$. Then by our formula, the eigenvalues are:

$$\lambda = \frac{5 \pm \sqrt{5^2 - 4 \times 4}}{2} = \frac{5 \pm 3}{2} = 4, 1$$

These are the multiples we found in the example above, as expected.

A real matrix can have complex eigenvalues and eigenvectors, but whenever it acts on a real vector, the result is still real. This is because the complex numbers cancel each other's imaginary parts. For discrete time models, it is enough to consider the absolute value of a complex eigenvalue, which is defined as following: $|a+bi| = \sqrt{a^2 + b^2}$. As before, the eigenvalue with the largest absolute value "wins" in the long term.

5.3.6 calculation of eigenvectors on paper

The surprising fact is that, as we saw in the last subsection, the eigenvalues of a matrix can be found without knowing its eigenvectors! However, the converse is not true: to find the eigenvectors, one first needs to know the eigenvalues. Given an eigenvalue λ, let us again write down the defining equation of the eigenvector for a generic 2 by 2 matrix:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = \begin{pmatrix} av_1 + bv_2 \\ cv_1 + dv_2 \end{pmatrix} = \lambda \begin{pmatrix} v_1 \\ v_2 \end{pmatrix}$$

This vector equation is equivalent to two algebraic equations:

$$av_1 + bv_2 = \lambda v_1$$
$$cv_1 + dv_2 = \lambda v_2$$

Since we've already found λ by solving the characteristic equation, this is two linear equations with two unknowns (v_1 and v_2). You may remember from advanced algebra that such equations may either have a single solution for each unknown, but sometimes they may have none, or infinitely many solutions. Since there are unknowns on both sides of the equation, we can make both equations be equal to zero:

$$(a-\lambda)v_1 + bv_2 = 0$$
$$cv_1 + (d-\lambda)v_2 = 0$$

So the first equation yields the relationship $v_1 = -v_2 b/(a-\lambda)$ and the second equation is $v_1 = -v_2(d-\lambda)/c$, which we already obtained in the last subsection. We know that these two equations must be the same, since the ratio of v_1 and v_2 is what defines the eigenvector. So we can use either expression to find the eigenvector.

Example. Let us return to the same matrix we looked at in the previous subsection:

$$A = \begin{pmatrix} 2 & 1 \\ 2 & 3 \end{pmatrix}$$

The eigenvalues of the matrix are 1 and 4. Using our expression above, where the element $a = 2$ and $b = 1$, let us find the eigenvector corresponding to the eigenvalue 1:

$$v_1 = -v_2 \times 1/(2-1) = -v_2$$

Therefore the eigenvector is characterized by the first and second elements being negatives of each other. We already saw in the example two subsections above that the vector $(1, -1)$ is such as eigenvector, but it is also true of the vectors $(-1, 1)$, $(-\pi, \pi)$ and $(10^6, -10^6)$. This infinite collection of vectors, all along the same direction, can be described as the eigenvector (or eigendirection) corresponding to the eigenvalue 1.

Repeating this procedure for $\lambda = 4$, we obtain the linear relationship:

$$v_1 = -v_2 \times 1/(2-4) = 0.5v_2$$

Once again, the example vector we saw two subsections $(2, 1)$ is in agreement with our calculation. Other vectors that satisfy this relationship include $(10, 5)$, $(-20, -10)$, and $(-0.4, -0.2)$. This is again a collection of vectors that are all considered the same eigenvector with eigenvalue 4 which are all pointing in the same direction, with the only difference being their length.

5.4 Age-structured population models

It is often useful to divide a population into different groups by age in order to better describe the population dynamics. Typically, individuals at different life stages have distinct mortality and reproductive rates. The total population is represented as a vector, where each component denotes the size of the corresponding age group. The matrix A that multiplies this vector defines the dynamics of the higher order difference equation:

$$\vec{x}_{t+1} = A\vec{x}_t$$

We will now analyze two common *age-structured models* used by biologists and demographers.

5.4.1 Leslie models

One type of age-structured model used to describe population dynamics is called the *Leslie model* (**edelstein-keshet_mathematical_2005?**; **allman_mathematical_2003?**). In this model, there are several different age groups, and after a single time step, individuals in each one all either advance to the next oldest age group or die. This type of can be described in general using the following matrix (called a Leslie matrix):

$$L = \begin{pmatrix} f_1 & f_2 & ... & f_n \\ s_1 & 0 & ... & 0 \\ ... & ... & ... & ... \\ 0 & ... & s_{n-1} & 0 \end{pmatrix}$$

where f_i is the fecundity (number of offspring produced by an individual) of the i-th age group, and s_i is the survival rate of the i-th age group (the fraction of the group that survives and becomes the next age group). Population of the next generation is given by multiplying the age-structure vector of the previous generation: $\vec{x}_{t+1} = L\vec{x}_t$. Note that each age group proceeds straight to the next age group (multiplied by the survival rate) but no individuals stay in the same age group after one time step. Biologically, this assumes a clear, synchronized maturation of every age group in the population. Mathematically, this means that the *diagonal elements* of the matrix (those in the i-th row and i-th column) are 0.

Let us model a hypothetical population in which there are two age groups: a young age group which does not reproduce, with survival rate of 0.4 to become mature, and a mature age group which reproduces with mean fecundity of 2, and then dies. Let j_t be the population of the juveniles at time t, and m_t be the population of mature adults. Then the following Leslie matrix describes this model:

$$\begin{pmatrix} j_{t+1} \\ m_{t+1} \end{pmatrix} = \begin{pmatrix} 0 & 2 \\ 0.4 & 0 \end{pmatrix} \begin{pmatrix} j_t \\ m_t \end{pmatrix}$$

We can also express this model as a single difference equation, with the variable of total population. Because it takes two time steps for a young individual to reproduce, we need to consider the population in two previous time steps. The matrix equation above can be written as the following two equations:

$$j_{t+1} = 2m_t;\ m_{t+1} = 0.4j_t$$

This two-dimensional model can be turned into a second-order model by a simple substitution. The first equation can be written as $j_t = 2m_{t-1}$, and then substitute it into second one to, to obtain:

$$m_{t+1} = 0.8m_{t-1}$$

We can solve this equation using the tools from the analytical section. First, let us find the exponential bases λ:

$$\lambda^2 = 0.8 \Rightarrow \lambda = \pm\sqrt{0.8}$$

To solve the dynamical system completely, let us suppose we have the initial conditions m_0 and m_1. Then we have the following equations to solve:

$$A+B = m_0;\ A\sqrt{0}.8-B\sqrt{0}.8 = m_1 \Rightarrow (m_0-B)\sqrt{0}.8-B\sqrt{0}.8 = m_1 \Rightarrow B = m_0-\frac{m_1}{\sqrt{8}};\ A = \frac{m_1}{\sqrt{8}}$$

We have the following analytic solution of the difference equation:

$$m_t = \frac{m_1}{\sqrt{8}}\sqrt{8}^t - \left(m_0 - \frac{m_1}{\sqrt{8}}\right)\sqrt{8}^t = 2m_1\sqrt{8}^{t-1} - m_0\sqrt{8}^t$$

This solution can be used to predict the long-term dynamics of the population model. Since the bases of the exponentials are less than 1, the total number of individuals will decline to zero. This solution can be verified via a numerical solution of this model. {numref}`fig-leslie` shows the population over 20 time steps, starting with 10 individuals both for m_0 and m_1.

5.4.2 Usher models

Usher models are a modification of the Leslie model, where individuals are allowed to remain in the same age group after one time step. Thus, the form of an Usher matrix is:

$$U = \begin{pmatrix} f_1 & f_2 & ... & f_n \\ s_1 & r_2 & ... & 0 \\ ... & ... & ... & ... \\ 0 & ... & s_{n-1} & r_n \end{pmatrix}$$

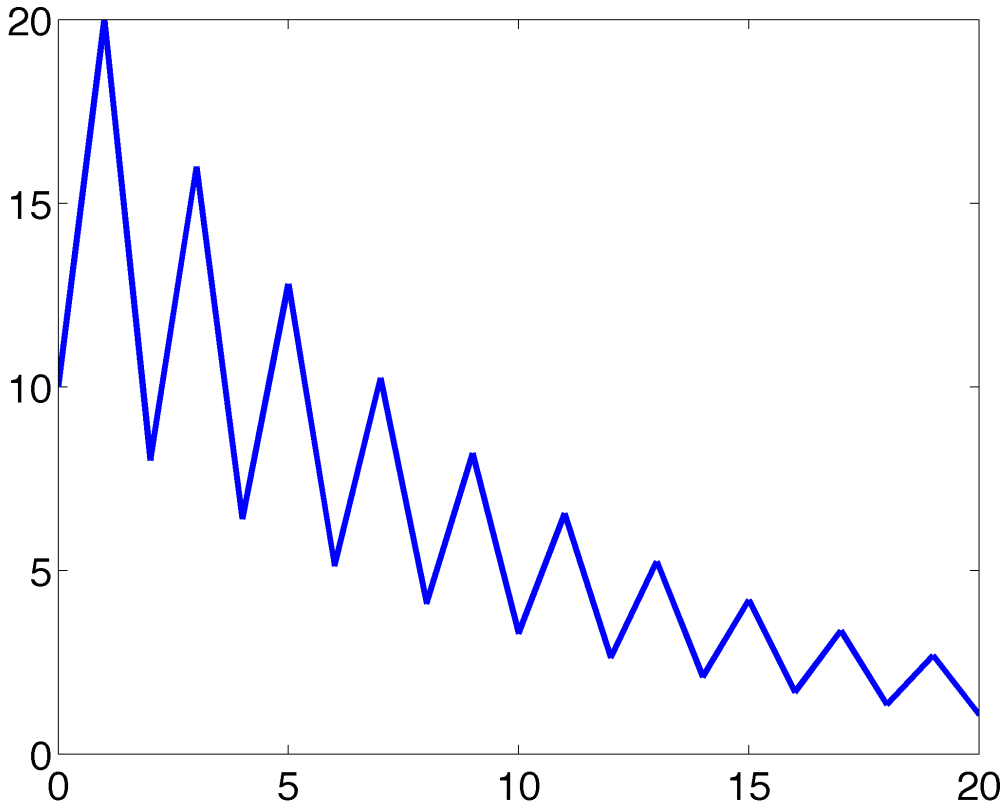


Figure 5.6: A plot of the total population in the Leslie model shown above, showing an oscillatory decay to 0

where r_i is the rate of remaining in the same age cohort.

For instance, it the population model above, we can introduce a rate of adults remaining adults (rather than dead) after a time step (let it be 0.2):

$$U = \begin{pmatrix} 0 & 2 \\ 0.4 & 0.2 \end{pmatrix}$$

$$j_{t+1} = 2m_t;\ m_{t+1} = 0.4j_t + 0.2m_t$$

Once again, we can find the solution for this model by recasting it as a single second-order equation. Let us substitute $2m_{t-1}$ for j_t to obtain the following:

$$m_{t+1} = 0.4(2m_{t-1}) + 0.2m_t$$

We can solve this second-order equation in the same fashion as above:

$$\lambda^2 = 0.2\lambda + 0.8 \Rightarrow \lambda = (0.2 \pm \sqrt{0.04 + 3.2})/2 = (0.2 \pm 1.8)/2 = 1, -0.8$$

The two exponential bases are 1 and -0.8, and therefore the solution has the general form $m_t = A + B(-0.8)^t$. The behavior of the solution over the long term is going to stabilize at some level A, determined by the initial conditions, because the term $B(-0.8)^t$, when raised to progressively larger powers, will decay to 0.

We can run a computer simulation to test this prediction, and see that the total population indeed approaches a constant. Starting with population of 10 individuals in the first two time steps, the time course of the population is plotted in {numref}`fig-usher`.

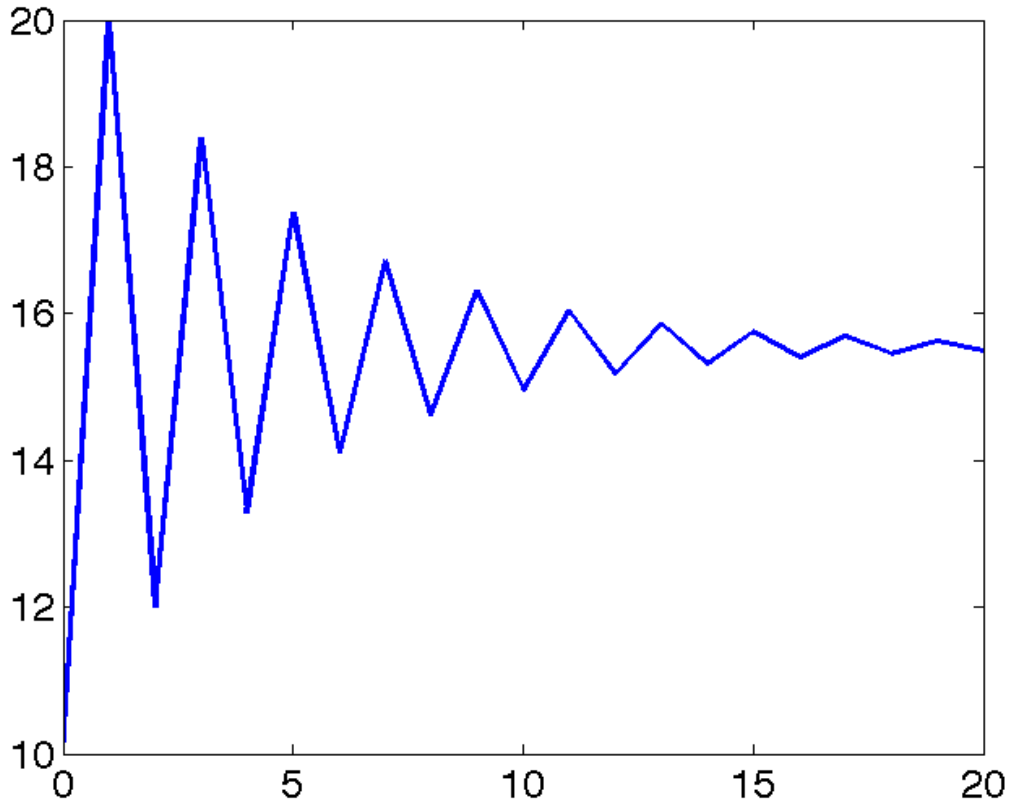


Figure 5.7: The total population of the Usher model shown above, showing oscillation and converging to a single value.

6 Matrix multiplication and population models

The solution of a difference equation can be found numerically using the matrix and vector form we introduced above. As we saw in the Fibonacci example, the next vector in the sequence can be obtained by multiplying the previous vector by the defining matrix. Let $\vec{x}_t$ be the vector containing n values of the dependent variables, and A be the defining matrix of dimension k by k. The solutions are obtained by repeated multiplication by the matrix A:

$$\vec{x}_{t+1} = A\vec{x}_t$$

We will now show how to implement this procedure in a program. For the simulation to be run, the program must set the following required components: the matrix defining the difference map, sufficient number of initial values, and the number of steps desired to iterate the solution. At each step, the current value of the vector of dependent variables is multiplied by the matrix A. In the following pseudocode I use two-dimensional arrays with two indices (row and column), and a colon in place of index indicates all of the elements in that dimension, e.g. $x[0, :]$ indicates the entire first row of array x. I assume that programming language has an operator for multiplying matrices, which is indicated by the multiplication symbol $\times$.

Solution of matrix discrete-time models

- set the number of variables k
- set age-structured matrix A
- set n to be the number of iterations (time steps)
- set the initial condition vector x_0
- initialize array x with n rows and $n + 1$ columns
- set the first column to x_0
- for i from 0 to $n - 1$
  - $x[:, i + 1] \leftarrow A \times x[:, i]$

This code produces a rectangular array x with k rows and $n + 1$ columns. The values of the variables at time j are stored in the vector $x[:, j]$. Conversely, in order to follow the dynamics of a particular variable over time, e.g. number i, through all n time steps, we can plot the vector $x[:, i]$.

Let us take the Fibonacci model again in the matrix form, with the matrix A and initial vector $\vec{x}_0$ as follows:

$$A = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix};\ \vec{x}_0 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

After iterating the matrix equation for 10 time steps, we obtain the following array X, with the fist and second row representing the population at the current and the previous time step, respectively, and the column representing time step:

$$\begin{array}{cccccccccc} 1 & 2 & 3 & 5 & 8 & 13 & 21 & 34 & 55 & 89 \\ 1 & 1 & 2 & 3 & 5 & 8 & 13 & 21 & 34 & 55 \end{array}$$

```
#Necessary imports
import numpy as np #package for work with arrays and matrices -- this week including some
import matplotlib.pyplot as plt #package with plotting capabilities
```

6.1 Matrix models in Python

In this section you will use Python's linear algebra library to compute the characteristic polynomial, eigenvalues, and eigenvectors of the models.

We saw in the section above that the found the eigenvalues by rewriting the equation for λ as a kth order polynomial, then found its roots. Python has a command for that, we can construct the characteristic polynomial of a matrix using the function poly(A), where A is a matrix. More specifically, we can find the coefficients for the characteristic polynomial.

Consider the Leslie population model from the example above:

$$\begin{pmatrix} j_{t+1} \\ m_{t+1} \end{pmatrix} = \begin{pmatrix} 0 & 2 \\ 0.4 & 0 \end{pmatrix} \times \begin{pmatrix} j_t \\ m_t \end{pmatrix}$$

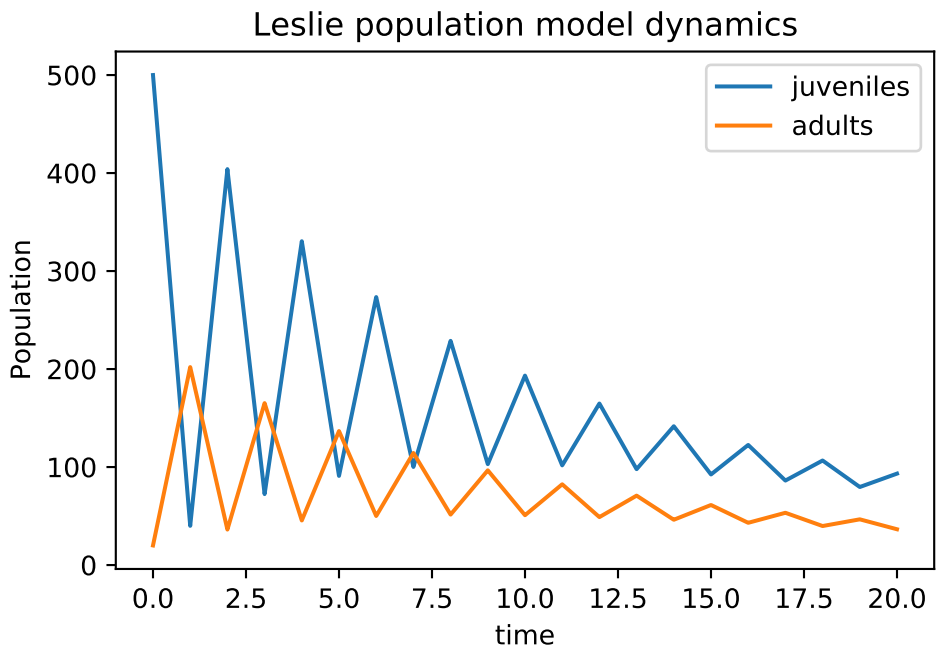
where j_t is the number of juveniles after t generations and m_t is the number of mature individuals after t generations. Propagation of this model requires multiplication of the matrix and the population vector. There is a special symbol in Python for this operation:

```
L=np.array([[0, 2], [0.4, 0]]) # define Leslie matrix
print(L)
pop = np.array([50, 10]) # define population column vector
print(pop)
new_pop = L@pop
print(new_pop)
```

```
[[0.  2. ]
 [0.4 0. ]]
[50 10]
[20. 20.]
```

This propagates the population by one time step only. To compute a numeric solution of this population over a number of time steps, use a for loop like in our last week's assignment. The only difference is that the solution is now a two-dimensional array instead of a one-dimensional one, with two rows for the two ages and numsteps+1 columns, and it needs to be pre-allocated before the for loop:

```
numsteps = 20; #number of time steps
L=np.array([[0, 2], [0.4, 0.1]]) # define Leslie matrix
pop = np.zeros([2, numsteps+1])
pop[:,0] = np.array([500,20]) #initialize the array with 50 juveniles and 10 adults
for i in range(numsteps):
    pop[:,i+1] = L@pop[:,i] #propagate the population vector for one step
plt.plot(pop[0,:],label='juveniles')
plt.plot(pop[1,:],label='adults')
plt.xlabel('time')
plt.ylabel('Population')
plt.title('Leslie population model dynamics')
plt.legend()
plt.show()
```



6.1.1 Eigenvalue and eigenvector analysis

The *eigenvalues* of the matrix L determine the dynamics of the population, the the *eigenvectors* determine the population structure. Python has a single function for finding eigenvalues and eigenvectors: `np.linalg.eig()`.

```
eVals, eVecs = np.linalg.eig(L)

print('Eigenvalues:')
print(eVals) #the order is flipped from the other method, but that's ok
print('Eigenvectors:')
print(eVecs)
```

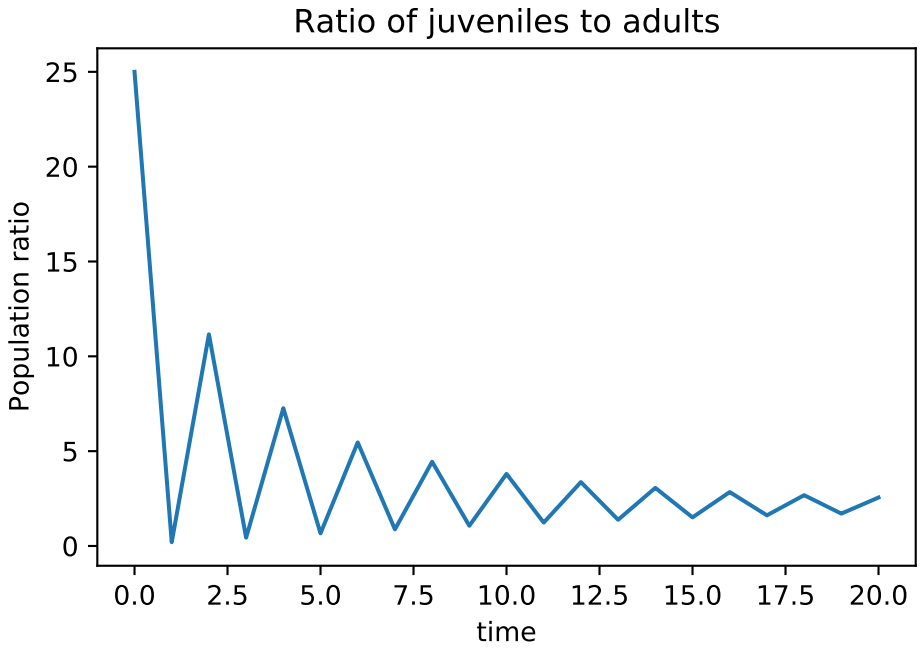
```
Eigenvalues:
[-0.84582364  0.94582364]
Eigenvectors:
[[-0.92102181 -0.90400779]
 [ 0.38951101 -0.42751597]]
```

Each column of the eVecs matrix corresponds to an eigenvaluein the eVals array (i.e. the first column of eigenvectors corresponds to the first element in eVals).

The largest (in absolute value) eigenvalue is the *dominant eigenvalue* and determines the long term behavior of the population. In this example, both eigenvalues are equal in absolute value and are less than 1, which predicts population decay.

The lack of a single dominant eigenvalue means that the population structure (ratio of juveniles and adults) does not converge to a stable fraction:

```
plt.plot(pop[0,:]/pop[1,:])
plt.xlabel('time')
plt.ylabel('Population ratio')
plt.title('Ratio of juveniles to adults')
plt.show()
```



6.1.2 Eigenvectors and population structure

Let us modify the Leslie matrix to allow the adults to survive with probability 0.2, which creates an Usher matrix with the following eigenvalues and eigenvectors. Below we also calculate the fraction of juveniles and adults in the long-term population:

```
U=np.array([[0, 2], [0.4, 0.2]]) # define Usher matrix
print(U)
```

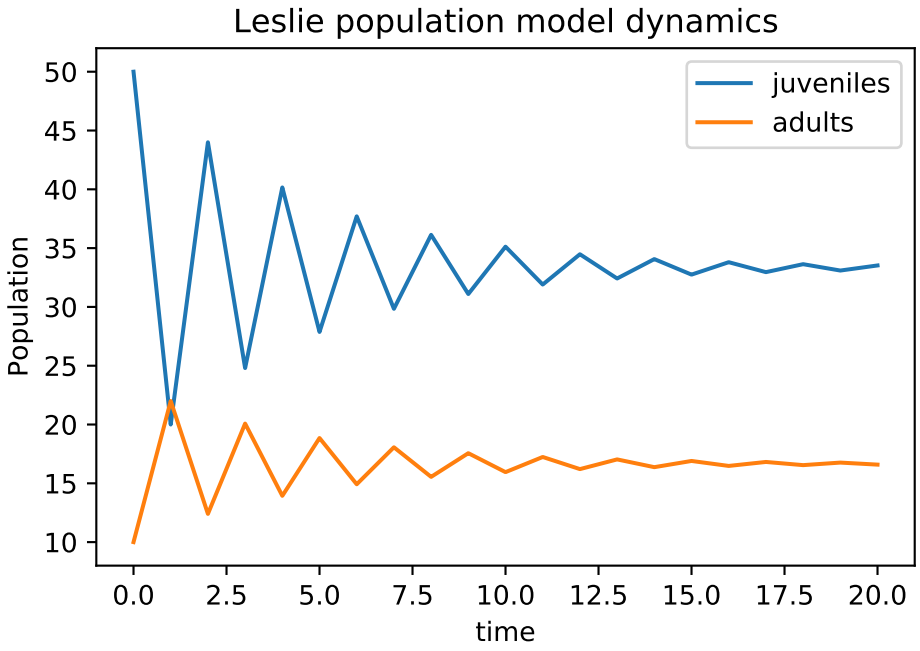
```
eVals, eVecs = np.linalg.eig(U)

print('Eigenvalues:')
print(eVals) #the order is flipped from the other method, but that's ok
print('Eigenvectors:')
print(eVecs)

print('The long term fractions of juveniles and adults are: ' + str(eVecs[:,1]/sum(eVecs[:
```

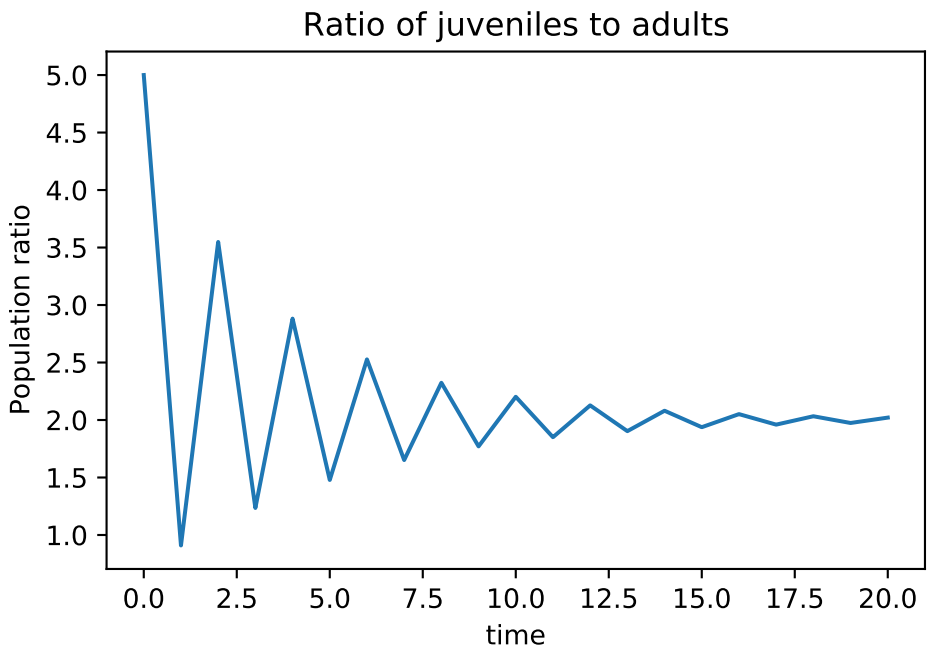
```
[[0.  2. ]
 [0.4 0.2]]
Eigenvalues:
[-0.8  1. ]
Eigenvectors:
[[-0.92847669 -0.89442719]
 [ 0.37139068 -0.4472136 ]]
The long term fractions of juveniles and adults are: [0.66666667 0.33333333]
```

Why did we use the second column (index 1)? Because it corresponds to the dominant eigenvalue 1, as you can check by looking at eVals. Notice that the population distribution remains stable in this population even as the total population declines, as can be seen by plotting the ratio of the juveniles to the adults:

```
numsteps = 20; #number of time steps
U=np.array([[0, 2], [0.4, 0.2]]) # define Leslie matrix
pop = np.zeros([2, numsteps+1])
pop[:,0] = np.array([50,10]) #initialize the array with 50 juveniles and 10 adults
for i in range(numsteps):
    pop[:,i+1] = U@pop[:,i] #propagate the population vector for one step
plt.plot(pop[0,:],label='juveniles')
plt.plot(pop[1,:],label='adults')
plt.xlabel('time')
plt.ylabel('Population')
plt.title('Leslie population model dynamics')
plt.legend()
plt.show()

plt.plot(pop[0,:]/pop[1,:])
plt.xlabel('time')
plt.ylabel('Population ratio')
plt.title('Ratio of juveniles to adults')
```

```
plt.show()
np.sqrt(3.24)
```






```
1.8
```

The juvenile/adult ratio converges to 2, as predicted by the leading eigenvector.

7 Linear regression

One of the most common ways of either fitting data, or if you want to put it in a fancier way, train a machine learning model, is called linear regression. This starts with a data set that has two different variables and pairs of observations of each, and produce a linear model that uses one variable (called explanatory) to predict the other (called response). Graphically speaking, the goal is to plot a line on a scatterplot that best fits the data (in one variable).

Though it is generally not possible to produce an exact fit for more than two observations, there is a method to calculate the closest linear model, called least-squares fitting. We will develop some fundamental tools from linear algebra to do this calculation, and then talk about the underlying assumptions and what they mean for applicability of linear regression.

7.0.1 List of terms and concepts

- Solving linear equations
- Matrix inverse
- Least-squares data fitting
- Explanatory vs. response variables and supervised learning
- Covariance and correlation
- Goodness of fit and R-squared
- Polynomial regression
- Residuals and assumptions of linear regression

7.1 Systems of linear equations

As one goes through life, sometimes one has to solve a set of linear equations, that have multiple variables (let's call them a and b) and the same number of equations that they need to satisfy with constant coefficients. For example, here is a system of two linear equations:

$$2a - b = -3a + b = 1$$

where we want to find a and b that satisfy both equations. This can be written as a matrix equation, with matrix M containing the coefficients on the left hand side and the vector $\vec{v}$

containing the two coefficients on the right hand side, and the vector $\vec{a}$ containing the unknown variables a and b:

$$\begin{pmatrix} 2 & -1 \\ 1 & 1 \end{pmatrix} \times \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} -3 \\ 1 \end{pmatrix} M\vec{a} = \vec{v}$$

Written as a single linear equation, it is tempting to "divide" both sides by M and thus solve for the vector $\vec{a}$, but matrices cannot be reciprocated like numbers. Linear algebra provides a way of doing this correctly.

In order to get rid of the matrix M on one side of the equation, one can multiply it by another matrix called its inverse.

Definition

For a square (n by n) matrix M the *inverse* matrix M^{-1} (also n by n) satisfies the following conditions: $M^{-1} \times M = M \times M^{-1} = I$, where I is the n by n identity matrix.

Example: For the matrix above, the inverse matrix is (check for yourself)

$$M^{-1} = \begin{pmatrix} 1/3 & 1/3 \\ -1/3 & 2/3 \end{pmatrix}$$

In general, finding the inverse of a matrix is best left to computers. However, for a 2 by 2 matrix, there is an explicit formula for an inverse:

$$M = \begin{pmatrix} \alpha & \beta \\ \gamma & \delta \end{pmatrix} M^{-1} = \frac{1}{\det(M)} \begin{pmatrix} \delta & -\beta \\ -\gamma & \alpha \end{pmatrix}$$

where the determinant $\det(M) = \alpha\delta - \beta\gamma$. Note that the division by the determinant of M in front of the matrix means every element of M is divided by determinant (as we see in the example above, where every element is divided by 3).

Once we have found the inverse of a matrix, we can solve the linear equation by multiplying both sides by the inverse:

$$M^{-1} \times M \times \vec{a} = V^{-1} \times \vec{v}\vec{a} = M^{-1} \times \vec{v}$$

In the example above, we multiply the vector $\vec{v}$ by the inverse and find the solution: $(a, b) = (-2/3, 5/3)$ (you can check that it works by plugging it into the original equations)

7.1.1 invertibility of matrices

It is useful to consider the geometric meaning of systems of linear equations. In two dimensions, as in the above example, each equation can be represented by a line in the plane. The solution to the two equations is the intersection of the two lines. The intersection is guaranteed to exist if the two lines are not parallel. If they are indeed parallel, then they either do not intersect at all, so there is no solution, or they overlap completely, in which case there are infinitely many solutions.

A similar geometric interpretation is true in higher dimensions. In three dimensions, each linear equation represents a plane, and as long as no two planes are parallel, there is only one point in which they intersect. But if two planes have the same direction, again, there is either no solution, or infinitely many (a line or plane of solutions). In higher dimensions, a solution is the intersection of n hyper-planes, and again, for a unique solutions to exist, no two hyper-planes can be parallel.

We saw the algebraic and geometric approach to solving systems of linear equations. In the algebraic solution, we can multiply by the inverse of the matrix, but we did not specify when it exists. Algebraically speaking, this can be determined from the determinant of the matrix, as in the formula for the inverse of a 2 by 2 matrix. This is the reason for the following fundamental result:

> **Theorem**
>
> Invertibility property: For a square (n by n) matrix M, an inverse matrix M^{-1} (also n by n) exists if any only if the determinant of M is not zero.

Geometrically speaking, a determinant of zero indicates that the intersection of the lines (or hyperplanes) is not a single point or speaking mathematically, they are not linearly independent. If that is the case, as we said above, there is not unique solution to the system of equations: there are either none, or infinitely many solutions.

7.2 Fitting a line to data

One of the most common questions in data science (or any science) is to describe a relationship between two numeric variables. Often, one is seen as the potential cause and the other as the effect, and they are called the explanatory and response variables, respectively. For example, {numref}`fig-cancer-risk` plots multiple data points of the cancer risk for different types of tissues plotted on the y-axis (response) as a function of the total number of cell divisions plotted on the x-axis (explanatory).

The question is: can the relationship between the variables be described by a linear function $y = ax + b$? And if so, how do you choose the best slope a and intercept b?

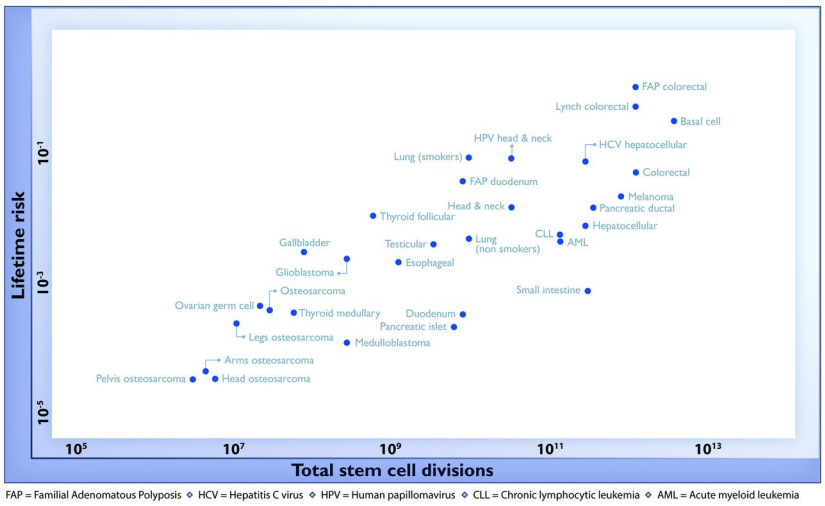


Figure 7.1: Cancer risk (response) as a function of number of cell divisions (explanatory); Figure from <https://www.science.org/doi/10.1126/science.1260825?)

The answer is straightforward if we only have two data points: we can use the exact solution that we described in the previous section. For example, if the two data points are $(-1,-2),(5,4)$, then the line that passes through both points must satisfy both equations below, with a and b being the slope and the intercept:

$$-a+b=-25a+b=4$$

To find the solution for a and b, we take the inverse of the matrix of coefficients M and multiply it by the vector $\vec{v}$ on the left hand side:

$$M=\begin{pmatrix}-1 & 1\\ 5 & 1\end{pmatrix};\vec{v}=\begin{pmatrix}-2\\ 4\end{pmatrix}M^{-1}\times\vec{v}=\frac{1}{-6}\begin{pmatrix}1 & -1\\ -5 & -1\end{pmatrix}\times\begin{pmatrix}-2\\ 4\end{pmatrix}=\begin{pmatrix}1\\ -1\end{pmatrix}$$

This means that a line with slope 1 and intercept -1 will pass through these two points.

But of course two data points is a very small amount of data to build a model. To make it just a bit more interesting, let's add one more data point, so our data set is: $(-1,-2),(5,4),(2,7)$. How can we find a line to fit those points?

Bad idea: Take two points and find a line, that is the slope and the intercept, that passes through the two. It should be clear why this is a bad idea: we are arbitrarily ignoring some of the data, while perfectly fitting two points.

So how do we use all the data? Let us write down the equations that a line with slope a and intercept b have to satisfy in order to fit our data points:

$$-a + b = -25a + b = 42a + b = 7$$

Let us write it in matrix form again:

$$\begin{pmatrix} -1 & 1 \\ 5 & 1 \\ 2 & 1 \end{pmatrix} \times \begin{pmatrix} a \\ b \end{pmatrix} = \begin{pmatrix} -2 \\ 4 \\ 7 \end{pmatrix} M \times \begin{pmatrix} a \\ b \end{pmatrix} = \vec{v}$$

This system has no exact solution, since there are three equations and only two unknowns. We need to find a and b such that they provide the *best fit* to the data, not the perfect solution. To do that, we need to define how to measure goodness of fit.

7.2.1 minimizing the sum of residuals

The most common approach to determine the goodness of fit is to subtract the predicted values of y from the data, as follows: $e_i = y_i - (mx_i + b)$. However, if we add it all up, the errors with opposite signs will cancel each other, giving the impression of a good fit simply if the deviations are symmetric. A more reasonable approach is to take absolute values of the deviations before adding them up. This is called the total deviation, for n data points with a line fit:

$$TD = \sum_{i=1}^{n} |y_i - mx_i - b|$$

Mathematically, a better measure of total error is a sum of squared errors, which also has the advantage of adding up nonnegative values, but is known as a better measure of the distance between the fit and the data (think of Euclidean distance, which is also a sum of squares):

$$SSE = \sum_{i=1}^{n} (y_i - mx_i - b)^2$$

To calculate the best-fit slope and intercept, we first need to define the variance and covariance of a data set:

> **Definition**
>
> The *variance* of a data set X with n data points is the following sum, where $\bar{X}$ is the mean of the data:

$$Var(X) = \frac{1}{n-1}\sum_{i=1}^{n}(\bar{X} - x_i)^2$$

The covariance of a data set of pairs of values (X, Y) is the sum of the products of the corresponding deviations from their respective means:

$$Cov(X, Y) = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{X})(y_i - \bar{Y})$$

Intuitively, this means that if two variables tend to deviate in the same direction from their respective means, they have a positive covariance, and if they tend to deviate in opposite directions from their means, they have a negative covariance. In the intermediate case, if sometimes they deviate together and other times they deviate in opposition, the covariance is small or zero. For instance, the covariance between two independent random variables is zero.

It should come as no surprise that the slope of the linear regression depends on the covariance, that is, the degree to which the two variables deviate together from their means. If the covariance is positive, then for larger values of x the corresponding y values tend to be larger, which means the slope of the line is positive. Conversely, if the covariance is negative, so is the slope of the line. And if the two variables are independent, the slope has to be close to zero. The actual formula for the slope of the linear regression is:

$$a = \frac{Cov(X, Y)}{Var(X)}$$

To find the intercept of the linear regression, we make use of one other property of the best fit line: in order for it to minimize the SSE, it must pass through the point $(\bar{X}, \bar{Y})$. Again, I will not prove this, but note that the point of the two mean values is the central point of the "cloud" of points in the scatterplot, and if the line missed that central point, the deviations will be larger. Assuming that is the case, we have the following equation for the line: $\bar{Y} = a\bar{X} + b$, which we can solve for b:

$$b = \bar{Y} - \frac{Cov(X, Y)\bar{X}}{Var(X)}$$

The parameters of the best-fit line can be calculated from the means, variances, and covariance of the two variable data set. But where did the formulas come from?

We want find the slope and intercept (a and b) which result in the lowest sum of squared errors. This approach is generally known as least squares fitting, and in the case of fitting a line, it is called linear *regression*. One way to find the values that minimize the sum of squared errors is to find the derivatives of SSE with respect to a and b and set them to 0:

$$\frac{\partial SSE}{\partial a} = \sum_{i=1}^{n} -2x_i(y_i - ax_i - b) = 0 \frac{\partial SSE}{\partial b} = \sum_{i=1}^{n} -2(y_i - ax_i - b) = 0$$

Re-write this with the y_is on the right hand side:

$$a\sum_{i=1}^{n} x_i^2 + b\sum_{i=1}^{n} x_i = \sum_{i=1}^{n} x_i y_i a \sum_{i=1}^{n} x_i + b\sum_{i=1}^{n} 1 = \sum_{i=1}^{n} y_i$$

This is now a linear system of equations, just as we started with. Turns out, there is compact way of representing this equation in matrix notation. Using the notation from the example above, let the matrix M contain a column of x values from the data, and a column of ones, and the vector $\vec{y}$ contain a column of y values of the data:

$$M = \begin{pmatrix} x_1 & 1 \\ ... & ... \\ x_n & 1 \end{pmatrix}; \vec{y} = \begin{pmatrix} y_1 \\ ... \\ y_n \end{pmatrix}$$

Then the equations above can be written as the following linear algebra equation, and solved using matrix inverse:

$$M^t \times M \times \begin{pmatrix} a \\ b \end{pmatrix} = M^t \times \vec{y} \begin{pmatrix} a \\ b \end{pmatrix} = (M^t \times M)^{-1} \times M^t \times \vec{y}$$

There is a linear algebra fact that the 2 by 2 matrix $M^t \times M$ is invertible so long as the columns of M are linearly independent. In this case this means as long as the x values of the data are not all the same, we can find a least-squares linear fit to a set of n data points. If you write down the solution for a and b as sums of all the components, you will obtain the formulas that were presented above.

One essential measure of the quality of linear regression is correlation, which is a measure of how much variation in one random variable corresponds to variation in the other. If this sounds very similar to the description of covariance, it's because they are closely related. Essentially, correlation is normalized covariance, made to range between -1 and 1. Here is the definition:

Definition

The (linear or Pearson) correlation of a data set of pairs of data values (X, Y) is:

$$r = \frac{Cov(X,Y)}{\sqrt{Var(X)Var(Y)}} = \frac{Cov(X,Y)}{\sigma_X \sigma_Y}$$

If the two variables are identical, $X = Y$, then the covariance becomes its variance $Cov(X, Y) = Var(X)$ and the denominator also becomes the variance, and the correlation is 1. This is also true if X and Y are scalar multiples of each other, as you can see by plugging in $X = cY$ into the covariance formula. The opposite case if X and Y are diametrically opposite, $X = -cY$, which has the correlation coefficient of -1. All other cases fall in the middle, neither perfect correlation nor perfect anti-correlation. The special case if the two variables are independent, and thus their covariance is zero, has the correlation coefficient of 0.

This gives a connection between correlation and slope of linear regression:

$$a = r\frac{\sigma_Y}{\sigma_X}$$

Whenever linear regression is reported, one always sees the values of correlation r and squared correlation r^2 displayed. The reason for this is that r^2 has the meaning of the the fraction of the variance of the dependent variable Y explained by the linear regression $Y = aX + b$.

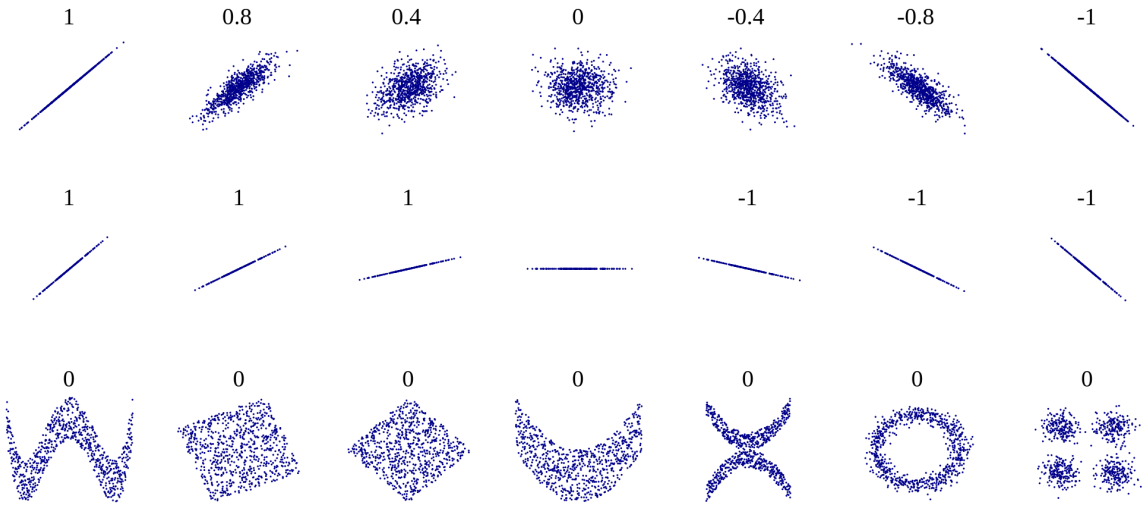


Figure 7.2: Correlation coefficient does not tell the whole story when it comes to describing the relationship between two variables; multiple scatterplots of generated data with correlation coefficient r shown above.http://en.wikipedia.org/wiki/File:Correlation_examples2.svg

There are, as usual, a couple of cautions about relying on the correlation coefficient First, just because there is no linear relationship, does not mean that there is no other relationship. {numref}`fig-corr-examples` shows some examples of scatterplots and their corresponding correlation coefficients. What it shows is that while a formless blob of a scatterplot will certainly have zero correlation, so will other scatterplots in which there is a definite relationship

(e.g. a circle, or a X-shape). The point is that **correlation is always a measure of the linear relationship between variables**.

Second cautionary tale is well known, as that is the danger of equating correlation with a causal relationship. There are numerous examples of scientists misinterpreting a coincidental correlation as meaningful, or deeming two variables that have a common source as causing one another. It cannot be repeated often enough that one must be careful when interpreting correlation: a weak one does not mean there is no relationship, and a strong one does not mean that one variable causes the variation in the other.

7.3 assumptions of linear regression

The simple formulas for slope, intercept, and standard deviation are only valid under certain conditions. The classic linear regression presented above relies on the following assumptions:

- the two variables have a linear relationship
- the measurements are all independent of each other
- there is no noise in the measurements of the independent variable
- the noise in the measurements of the dependent variable is normally distributed with the same variance

In reality, each data measurement has a random component, that we can call noise, resulting from experimental error, environmental variation, etc, and different measurements may have different levels of noise (standard deviation). One can estimate the error for a measurement, for instance by repeating the experiment several times, and estimating the standard deviation of the measurement random variable (we will not get into how to do this until the third quarter). It is important to account for this uncertainty in the data, since a measurement which is all over the place must carry less weight than one which is solid. A proper mathematical way of doing this is by defining a different function to measure the goodness of fit, known as the chi-squared function:

$$\chi^2 = \sum_{i=1}^{n} \frac{(y_i - ax_i - b)^2}{\sigma_i^2}$$

where σ_i is the standard deviation of the i-th data point. Given all this information, we can find a solution analogous to the one found in the previous section. The only modification is to divide the matrices by the standard deviation σ_i:

$$M = \begin{pmatrix} x_1/\sigma_i & 1/\sigma_1 \\ ... & ... \\ x_n/\sigma_n & 1/\sigma_n \end{pmatrix} \vec{y} = \begin{pmatrix} y_1/\sigma_1 \\ ... \\ y_n/\sigma_n \end{pmatrix}$$

Then the least squares solution is found by the same formula as above, but here we have accounted for the experimental uncertainty:

$$\begin{pmatrix} a \\ b \end{pmatrix} = (M^t \times M)^{-1} \times M^t \times \vec{y}$$

7.4 linear least squares for polynomial fitting

Fitting data sets is not restricted to linear functions. One simple extension is extension is to higher degree polynomials. Let us consider a quadratic function: $y = ax^2 + bx + c$. By analogy with the equations for fitting a linear function, we have a set of n equations, one for each data point:

$$ax_1^2 + bx_1 + c = y_1 ax_2^2 + bx_2 + c = y_2 .... ax_n^2 + bx_n + c = y_n$$

Thus, we can define the matrix M for the least-squares quadratic fit, along with the same vector $\vec{y}$ as follows:

$$M = \begin{pmatrix} x_1^2 & x_1 & 1 \\ ... & ... & ... \\ x_n^2 & x_n & 1 \end{pmatrix} ; \ \vec{y} = \begin{pmatrix} y_1 \\ ... \\ y_n \end{pmatrix}$$

and find the best fit parameters for the quadratic function:

$$\begin{pmatrix} a \\ b \\ c \end{pmatrix} = (M^t \times M)^{-1} \times M^t \times \vec{y}$$

It is straightforward to extend this to higher order polynomials, just by adding columns of higher powers of x data to the matrix M. The basic structure of the solution remains the same.

Another important concern is about the appropriate number of parameters in a fit for a particular data set. It is clear that adding more parameters results in better fit, but at some point the number of parameters is too large, and "over-fitting" becomes as issue. Obviously, if one uses the same number of parameters as data points, one can obtain a perfect fit that has little predictive power - it just matches the given data. Deciding at what point adding more parameters is not productive is a difficult question, which can be addressed by various statistical methods that are outside of the scope of the course.

8 Linear regression in Python

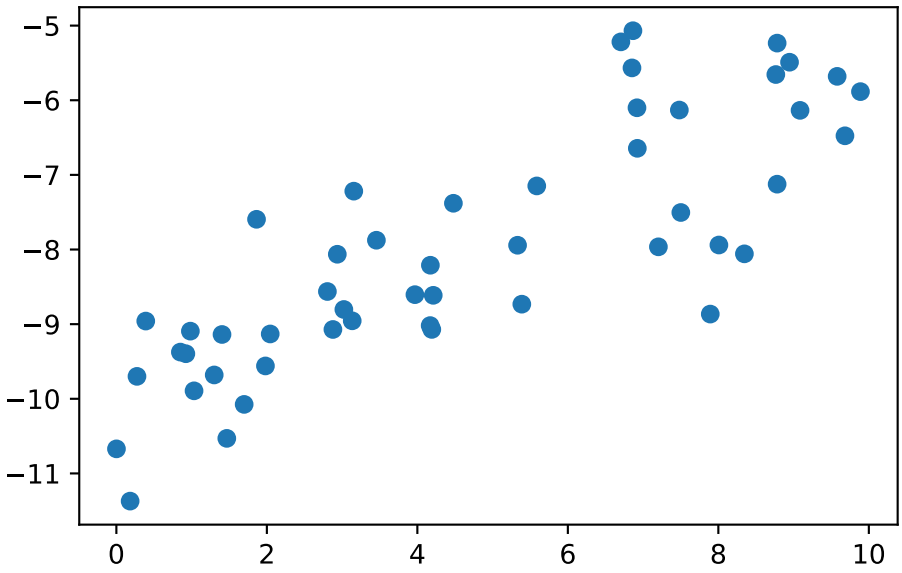
```
# Necessary imports
import numpy as np #package for work with arrays and matrices
import matplotlib.pyplot as plt #package with plotting capabilities
from scipy import stats
import pandas as pd
```

8.1 Linear regression on 2-variable data sets

Linear regression is a supervised learning method for predicting the value of a response variable (Y) based on a linear model of the explanatory variable (X). The following scripts illustrate it using a function from the sklearn package (code adopted from https://jakevdp.github.io/PythonDataScienceHandbook/05.06-linear-regression.html)

Let us generate a data set with y a linear function of x with known slope and intercept, plus added random noise:

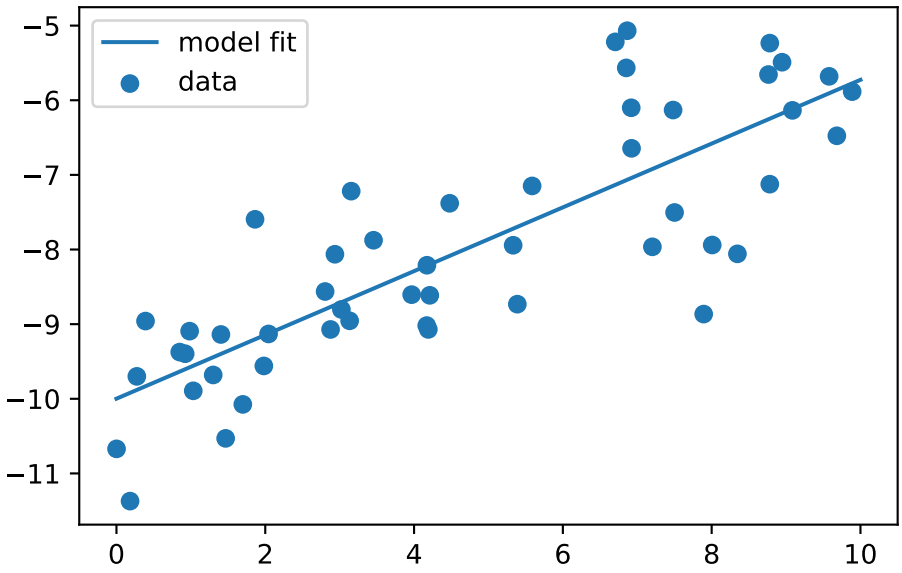
```
m = 0.4 # slope
b = -10 # intercept
rng = np.random.RandomState(1)
x = 10 * rng.rand(50)
y = m * x + b + rng.randn(50)
plt.scatter(x, y)
plt.show()
```



Ue the LinearRegression function to see whether it returns the correct slope and intercept and how well the line fits the data:

```
slope, intercept, r, p_value, std_err = stats.linregress(x,y)

print("Model slope:    ", slope)
print("Model intercept:", intercept)
print("R^2:            ", r**2)

xfit = np.linspace(0, 10, 1000)
yfit = xfit*slope + intercept

plt.scatter(x, y, label = 'data')
plt.plot(xfit, yfit, label = 'model fit')
plt.legend()
plt.show()
```

```
Model slope:     0.4272088103606956
Model intercept: -9.998577085553208
R^2:             0.6751620299329717
```



8.1.1 Example of baby mass data set

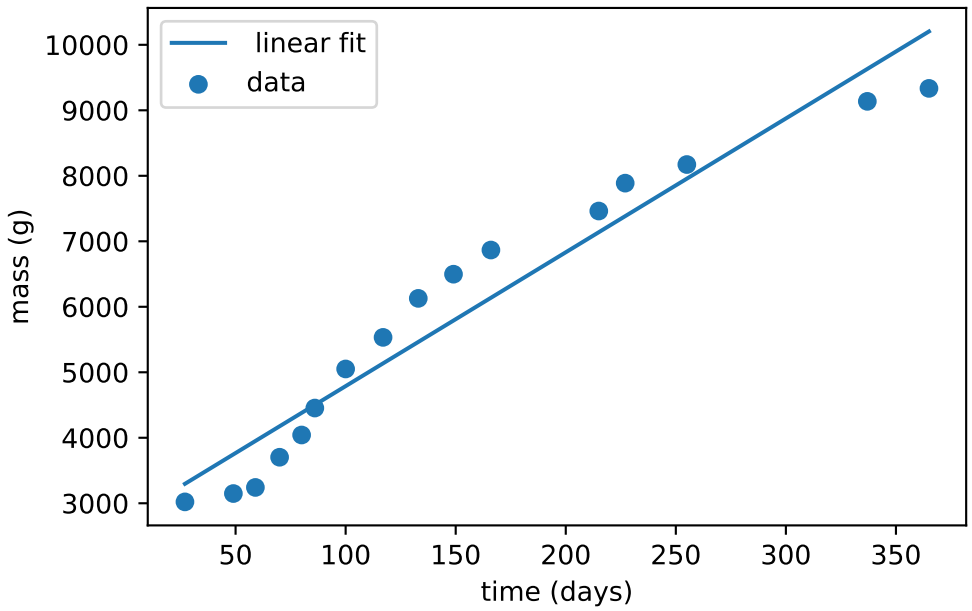
Load the data set `newborn_mass.csv` which contains two variables: days (in days after birth) and mass (in grams) using the numpy function loadtxt: https://docs.scipy.org/doc/numpy-1.15.0/reference/generated/numpy.loadtxt.html Note that you'll need to skip the first row and specify comma as the delimiter.

Use your linear regression function to find the best-fit line between the explanatory variable of time (days) and response variable of mass. Make a scatterplot with the regression line overlayed. Based on the determination coefficient, what fraction of the variance in the response variable is explained by the linear fit?

```
baby = pd.read_csv("data/newborn_mass.csv")
print(baby.head())

days = baby.days
mass = baby.grams

slope, intercept, r, p_value, std_err = stats.linregress(days, mass)
print("Model slope:    ", slope)
print("Model intercept:", intercept)
print("R^2:            ", r**2)

plt.scatter(days, mass, label = 'data')
```

```
new_mass = slope*days+intercept
plt.plot(days, new_mass, label = ' linear fit')

plt.xlabel('time (days)')
plt.ylabel('mass (g)')
plt.legend()
plt.show()
```

```
   days     grams
0    27  3021.150
1    49  3148.500
2    59  3241.675
3    70  3702.750
4    80  4042.350
Model slope:     20.43433410132227
Model intercept: 2745.1200914550172
R^2:             0.9367352637458148
```



9 Models with one variable in continuous time

In the previous chapters we have considered discrete time models, in which time is measured in integers. This worked well to describe processes that happen in periodic cycles, like cell division or heart pumping. Many biological systems do not work this way. Change can happen continuously, that is, at any point in time. For instance, the concentration of a biological molecule in the cell changes gradually, as does the voltage across the cell membrane in a neuron.

The models for continuously changing variables require their own set of mathematical tools. Differential equations use derivatives to describe how a variable changes with time. There is a tremendous amount of knowledge accumulated by mathematicians, physicists and engineers for analyzing and solving differential equations. There are many classes of differential equations for which it is possible to find analytic solutions, often in the form of "special functions." Differential equations courses for physicists and engineers are typically focused on learning about the variety of existing tools for solving a few types of differential equations. For the purposes of biological modeling, knowing how to solve a limited number of differential equations is of limited usefulness. We will instead focus on learning how to analyze the behavior of differential equations in general, without having to solve them on paper.

In this chapter we will analyze linear differential equations, which have mathematical solutions, and learn about numerical methods for computing and plotting these solution. Specifically, you will learn to:

- build differential equations of population size
- mathematically solve linear differential equations
- put together a model of membrane potential
- compute numeric solutions of these models
- compute and analyze the error in numeric solutions

9.1 Ordinary differential equations

We consider models with *continuous time*, for which it does not make sense to break time up into equal intervals. Instead of equations describing the increments in the dependent variable from one time step to the next, we will see equations with the instantaneous rate of the change (derivative) of the variable. For discrete time models, one formulation of the general difference equation was this:

$$x_{t+1} - x_t = g(x)$$

$g(x)$ is a function of the dependent variable, which may be as simple as 0 or ax, or can be horribly nonlinear and complicated.

For difference equations, the time variable t is measured in the number of time steps (Δt), whether the time step is 20 minutes or 20 years. In continuous time models, we express t in actual units of time, instead of counting time steps. Thus, what we wrote as $t+1$ for discrete time should be expressed as $t + \Delta t$ for continuous time. The left-hand-side of the equation above describes the change in the variable x over one time step Δt. We can write it as a Newton's quotient, and then take the limit of the time step shrinking to 0:

$$\lim_{\Delta t \to 0} \frac{x(t + \Delta t) - x(t)}{\Delta t} = \frac{dx}{dt} = g(x)$$

To take the limit of the time step going to 0 means that we allow the increments in time to be infinitesimally small, and therefore the time variable may be any real number. The equation above thus becomes a differential equation, because it involves a derivative of the dependent variable.

In general, an ordinary differential equation is defined as follows:

> **Definition**
>
> An ordinary differential equation is an equation that contains derivatives of the dependent variable (e.g. x) with respect to an independent variable (e.g. t).

For example:

$$\frac{dx^2}{dt^2} + 0.2\frac{dx}{dt} - 25 = 0$$

There are at least two good reasons to use differential equations for many applications. First, some events happen very frequently and non-periodically, so it is more reasonable to allow time to flow continuously instead of in steps. The second reason is mathematical: it turns out that dynamical systems with continuous time, described by differential equations, are better behaved than difference equations. This has to do with the essential "jumpiness" of difference equations. Even for simple nonlinear equations, the value of the variable after one time step can be far removed from its last value. This can lead to highly complicated solutions, as we saw in the logistic model in Chapter 1.

9.1.1 growth proportional to population size

We will now build up some of the most common differential equations models. First up, a simple population growth model with a constant growth rate. Suppose that in a population each individual reproduces with the average reproductive rate r. This is reflected in the following differential equation:

$$\frac{dx}{dt} = \dot{x} = rx$$

(linear_ode)

This expression states that the rate of change of x, which we take to be population size, is proportional to x with multiplicative constant r. We will frequenlty use the notation $\dot{x}$ for the time derivative of x for aesthetic reasons.

First, we apply dimensional analysis to this model. The units of the derivative are population per time, as can be deduced from the Newton's quotient definition. Thus, the units in the equation have the following relationship:

$$\frac{[population]}{[time]} = [r][population] = \frac{1}{[time]}[population]$$

This shows that as in the discrete time models, the dimension of the population growth rate r is inverse time, or frequency. The difference with the discrete time population models lies in the time scope of the rate. In the case of the difference equation, r is the rate of change per one time step of the model. In the differential equation, r is the *instantaneous rate of population growth*. It is less intuitive than the growth rate per single reproductive cycle, just like the slope of a curve is less intuitive than the slope of a line. The population growth happens continuously, so the growth rate of r individuals per year does not mean that if we start with one individual, there will be r after one year. In order to make quantitative predictions, we need to find the solution of the equation, which we will see in the next section.

9.1.2 chemical kinetics

Reactions between molecules in cells occur continuously, driven by molecular collisions and physical forces. In order to model this complex behavior, it is generally assumed that reactions occur with a particular speed, known as the *kinetic rate*. A simple reaction of conversion from one type of molecule (A) to another (B) can be written as follows:

$$A \to^k B$$

In this equation the parameter k is the kinetic rate, describing the speed of conversion of A into B, per concentration of A.

Chemists and biochemists use differential equations to describe the change in molecular concentration during a reaction. These equations are known as the *laws of mass action*. For the reaction above, the concentration of molecule A decreases continuously proportionally to itself, and the concentration of molecule B increases continuously proportionally to the concentration of A. This is expressed by the following two differential equations:

$$\begin{aligned} \dot{A} &= -kA \\ \dot{B} &= kA \end{aligned}$$

(lin_chem_kin)

Several conclusions can be drawn by inspection of the equations. First, the dynamics depend only on the concentration of A, so keeping track of the concentration of B is superfluous. The second observation reinforces the first: the sum of the concentrations of A and B is constant. This is mathematically demonstrated by adding the two equations together to obtain the following:

$$\dot{A} + \dot{B} = -kA + kA = 0$$

One of the basic properties of the derivative is that the sum of derivatives is the same as the derivative of the sum:

$$\dot{A} + \dot{B} = \frac{d(A+B)}{dt} = 0$$

This means that the sum of the concentrations of A and B is a constant. This is a mathematical expression of the law of conservation in chemistry: molecules can change from one type to another, but they cannot appear or disappear in other ways. In this case, a single molecule of A becomes a single molecule of B, so it follows that the sum of the two has to remain the same. If the reaction were instead two molecules of A converting to a molecule of B, then the conserved quantity is $2A+B$. The concept of conserved quantity is very useful for the analysis of differential equations. We will see in later chapters how it can help us find solutions, and explain the behavior of complex dynamical systems.

9.2 Analytic solutions of linear ODEs

9.2.1 concepts of ODEs

Let us define some terminology for ODEs:

:::{.callout-note}
Definition The *order* of an ODE is the highest order of the derivative of the dependent variable x. :::

For example, $\dot{x} = rx$ is a first order ODE, while $\ddot{x} = -mx$ is a second order ODE (double dot stands for second derivative). In this chapter we will restrict ourselves to first-order ODEs that can be generally written as follows:

Definition

A *first-order* ODE is one where the derivative dx/dt is equal to a *defining function* $f(x,t)$, like this:

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

(first-order-ode)

Note that the function may depend on both the dependent variable x and the independent variable t. This leads to the next definition:

Definition

An ODE is *autonomous* if the defining function f depends only on the dependent variable x and not on t.

For example, $\dot{x} = 5x - 4$ is an autonomous equation, while $\dot{x} = 5t$ is not. An autonomous ODE is also said to have *constant coefficients* (e.g. 5 and -4 in the first equation above).

Definition

An ODE is *homogeneous* if every term in the defining function involves either the dependent variable x or its derivative.

For example, $\dot{x} = x^2 + \sin(x)$ is homogeneous, while $\dot{x} = -x + 5t$ is not. Most simple biological models that we will encounter in the next two chapters are autonomous, homogeneous ODEs. However, inhomogeneous equations are important in many applications, and we will encounter them at the end of the present section.

9.2.2 solutions via separate-and-integrate

In contrast with algebraic equations, we cannot simply isolate x on one side of the equal sign and find the solutions as one, or a few numbers. Instead, solving ordinary differential equations is very tricky, and no general strategy for solving an arbitrary ODE exists. Moreover, a solution

for an ODE is not guaranteed to exist at all, or not for all values of t. We will discuss some of the difficulties later, but let us start with equations that we can solve.

> **Definition**
>
> The *analytic (or exact) solution* of an ordinary differential equation is a function of the independent variable that satisfies the equation. If no initial value is given, then the *general solution* function will contain an uknown *integration constant*. If an initial value is specified, the integration constant can be found to obtain a *specific solution*.

This means that the solution function obeys the relationship between the derivative and the defining function that is specified by the ODE. To verify that a function is a solution of a given ODE, take its derivative and check whether it matches the other side of the equation.

Example. The function $x(t) = 3t^2 + C$ is a general solution of the ODE $\dot{x} = 6t$, which can be verified by taking the derivative: $\dot{x}(t) = 6t$. Since this matches the right-hand side of the ODE, the solution is valid.

Example. The function $x(t) = Ce^{5t}$ is a general solution of the ODE $\dot{x} = 5x$. This can be verified by the taking the derivative: $\dot{x} = 5Ce^{5t}$ and comparing it with the right-hand side of the ODE: $5x = 5Ce^{5t}$. Since the two sides of the equation agree, the solution is valid.

In contrast with algebraic equations, we cannot simply isolate x on one side of the equal sign and find the solutions as one, or a few numbers. Instead, solving ordinary differential equations is very tricky, and no general strategy for solving an arbitrary ODE exists. Moreover, a solution for an ODE is not guaranteed to exist at all, or not for all values of t. We will discuss some of the difficulties later, but let us start with equations that we can solve.

The most obvious strategy for solving an ODE is integration. Since a differential equation contains derivatives, integrating it can remove the derivative. In the case of the general first order equation, we can integrate both sides to obtain the following:

$$\int \frac{dx}{dt} dt = \int f(x,t) dt \Rightarrow x(t) + C = \int f(x,t) dt$$

The constant of integration C appears as in the standard antiderivative definition. It can be specified by an initial condition for the solution $x(t)$. Unless the function $f(x,t)$ depends only on t, it is not possible to evaluate the integral above. Instead, various tricks are used to find the analytic solution. The simplest method of analytical solution of a first-order ODEs, which I call *separate-and-integrate* consists of the following steps:

Outline of separate-and-integrate method

1. use algebra to place the dependent and independent variables on different sides of the equations, including the differentials (e.g. dx and dt)
2. integrate both sides with respect to the different variables, don't forget the integration constant
3. solve for the dependent variable (e.g. x) to find the *general solution*
4. plug in $t = 0$ and use the initial value $x(0)$ to solve for the integration constant and find the *specific solution*

Example. Consider a very simple differential equation: $\dot{x} = a$, where $\dot{x}$ stands for the time derivative of the dependent variable x, and a is a constant. It can be solved by integration:

$$\int \frac{dx}{dt} dt = \int a dt \Rightarrow x(t) + C = at$$

This solution contains an undetermined integration constant; if an initial condition is specified, we can determine the complete solution. Generally speaking, if the initial condition is $x(0) = x_0$, we need to solve an algebraic equation to determine C: $x_0 = a \times 0 - C$, which results in $C = -x_0$. The complete solution is then $x(t) = at + x_0$. To make the example more specific, if $a = 5$ and the initial condition is $x(0) = -3$, the solution is $x(t) = 5t - 3$.

Example. Let us solve the linear population growth model in equation {eq}`linear_ode`: $\dot{x} = rx$. The equation can be solved by first dividing both sides by x and then integrating:

$$\int \frac{1}{x} \frac{dx}{dt} dt = \int \frac{dx}{x} = \int r dt \Longrightarrow \log |x| = rt + C \Longrightarrow x = e^{rt+C} = Ae^{rt}$$

We used basic algebra to solve for x, exponentiating both sides to get rid of the logarithm on the left side. As a result, the additive constant C gave rise to the multiplicative constant $A = e^C$. Once again, the solution contains a constant which can be determined by specifying an initial condition $x(0) = x_0$. In this case, the relationship is quite straightforward: $x(0) = Ae^0 = A$. Thus, the complete solution for equation {eq}`linear_ode` is:

$$x(t) = x_0 e^{rt}$$

As in the case of the discrete-time models, population growth with a constant birth rate has exponential form. Once again, please pause and consider this fact, because the exponential solution of linear equations is one of the most basic and powerful tools in applied mathematics. Immediately, it allows us to classify the behavior of linear ODE into three categories:

Classification of solutions of linear ODEs

For the solution of the ODE $\dot{x} = rx$

- $r > 0$: $x(t)$ grows without bound
- $r < 0$: $x(t)$ decays to 0
- $r = 0$: $x(t)$ remains constant at the initial value

The rate r being positive means that the birth rate is greater than the death rate in the population, leading to unlimited population growth. If the death rate is greater, the population will decline and die out. If the two are exactly matched, the population size will remain unchanged.

Example. The solution for the biochemical kinetic model in equation {eq}`lin_chem_kin` is identical except for the sign: $A(t) = A_0 e^{-kt}$. When the reaction rate k is positive, as it is in chemistry, the concentration of A decays to 0 over time. This is consistent with the arrow diagram of this model, since there is no back reaction, and the only chemical process is conversion of A into B. The concentration of B can be found by using the fact that the total concentration of molecules in the model is conserved. Let us call it C. Then $B(t) = C - A(t) = C - A_0 e^{-kt}$. The concentration of B increases to the asymptotic limit of C, meaning that all molecules of A have been converted to B.

9.2.3 solution of nonhomogeneous ODEs

ODEs that contain at least one term without the dependent variable are a bit more complicated. If the defining function is $f(x, t)$ is *linear* in the dependent variable x, they can be solved on paper using the same separate-and-integrate method, modified slightly to handle the constant term. Here are the steps to solve the generic linear ODE with a constant term $\dot{x} = ax + b$:

Tip

solution of linear autonomous ODEs
Consider an ODE of the form $\dot{x} = ax + b$

1. separate the dependent and independent variables on different sides of the equations, by dividing both sides by the right hand side $ax + b$, and multiplying both sides by the differential dt
2. integrate both sides with respect to the different variables, don't forget the integration constant!
3. solve for the dependent variable (e.g. x) to find the *general solution*

4. plug in $t = 0$ and use the initial value $x(0)$ to solve for the integration constant and find the *specific solution*

Example: Let us solve the following ODE model using separate and integrate with the given initial value:

$$\frac{dx}{dt} = 4x - 100;\ x(0) = 30$$

- Separate the dependent and independent variables:

$$\frac{dx}{4x - 100} = dt$$

- Integrate both sides:

$$\int \frac{dx}{4x - 100} = \int dt \Rightarrow \frac{1}{4}\int \frac{du}{u} = \frac{1}{4}\ln|4x - 100| = t + C$$

The integration used the substitution of the new variable $u = 4x - 100$, with the concurrent substitution of $dx = du/4$.

- Solve for the dependent variable:

$$\ln|4x - 100| = 4t + C \Rightarrow 4x - 100 = e^{4t}B \Rightarrow x = 25 + Be^{4t}$$

Here the first step was to multiply both sides by 4, and the second to use both sides as the exponents of e, removing the natural log from the left hand side, and finally simple algebra to solve for x as a function of t.

- Solve for the integration constant:

$$x(0) = 25 + B = 30 \Rightarrow B = 5$$

Here the exponential "disappeared" because $e^0 = 1$. Therefore, the specific solution of the ODE with the given initial value is

$$x(t) = 25 + 5e^{4t}$$

At this point, you might have noticed something about solutions of linear ODEs: they always involve an exponential term, with time in the exponent. Knowing this, it is possible to bypass the whole process of separate-and-integrate by using the following short-cut.

Important Fact

Any linear ODE of the form $\dot{x} = ax + b$ has an analytic solution of the form:

$$x(t) = Ce^{at} + D$$

with $D = -b/a$ and C determined by the initial value $x(0)$.

This can be verified by plugging the solution back into the ODE to see if it satisfies the equation. First, take the derivative of the solution to get the left-hand side of the ODE: $\frac{dx}{dt} = Cae^{at}$; then plug in $x(t)$ into the right hand side of the ODE: $ aCe^{at} + aD +b$. Setting the two sides equal, we get: $Cae^{at} = aCe^{at} + aD + b$, which is satisfied if $aD + b = 0$, which means $D = -b/a$. This is consistent with the example above, the additive constant in the solution was 25, which is $-b/a = -(-100)/4 = 25$.

Thus, if you want to solve a linear no ODE $\dot{x} = ax + b$, you can bypass the separate-and-integrate process, because the general solution always has the form in equation {eq}`sol-nonhom`. So the upshot is that all linear ODEs have solutions which are exponential in time with exponential constant coming from the slope constant a in the ODE. The dynamics of the solution are determined by the sign of the constant a: if $a > 0$, the solution grows (or declines) without bound; and if $a < 0$, the solution approaches an asymptote at $-b/a$ (from above or below, depending on the initial value).

9.2.4 model of drug concentration

Describing and predicting the dynamics of drug concentration in the body is the goal of *pharmacokinetics*. Any drug that humans take goes through several stages: first it is administered (put into the body), then absorbed, metabolized (transformed), and excreted (removed from the body) (**rosenbaum_basic_2011?**). Almost any drug has a dose at which it has a toxic effect, and most can kill a human if the dose is high enough. Drugs which are used for medical purposes have a *therapeutic range*, which lies between the lowest possible concentration (usually measured in the blood plasma) that achieves the therapeutic effect and the concentration which is toxic. One of the basic questions that medical practitioners need to know is how much and how frequently to administer a drug to maintain drug concentration in the therapeutic range.

The concentration of a drug is a dynamic variable which depends on the rates of several processes, most directly on the rate of administration and the rate of metabolism. Drugs can be administered through various means (e.g. orally or intravenously) which influences their rate of absorption and thus how the concentration increases. Once in the blood plasma, drugs are metabolized primarily by enzymes in the liver, converting drug molecules into compounds that can be excreted through the kidneys or the large intestine. The process of *metabolism proceeds at a rate that depends on both the concentration of the drug and on the enzyme that

catalyzes the reaction. For some drugs the metabolic rate may be constant, or independent of the drug concentration, since the enzymes are already working at full capacity and can't turn over any more reactions, for example alcohol is metabolized at a constant rate of about 1 drink per hours for most humans. {numref}`fig-alc-met` shows the time plots of the blood alcohol concentration for 4 males who ingested different amounts of alcohol, and the curves are essentially linear with the same slope after the peak. For other drugs, if the plasma concentration is low enough, the enzymes are not occupied all the time and increasing the drug concentration leads to an increase in the rate of metabolism. One can see this behavior in the metabolism of the anti-depressant drug bupropion in figure {numref}`fig-bupropion`, where the concentration curve shows a faster decay rate for higher concentration of the drug than for lower concentration. In the simplest case, the rate of metabolism is linear, or proportional to the concentration of the drug, with proportionality constant called the first-order metabolic rate.

Example: ODE model of drug kinetics

Let us build an ODE model for a simplified pharmacokinetics situation. Suppose that a drug is administered at a constant rate of M (concentration units per time unit) and that it is metabolized at a rate proportional to its plasma concentration C with metabolic rate constant k. Then the ODE model of the concentration of the drug over time $C(t)$ is:

$$\frac{dC}{dt} = M - kC$$

The two rate constants M and k have different dimensions, which you should be able to determine yourself. The ODE can be solved using the separate-and-integrate method:

- Divide both sides by the right hand side $M - kC$, and multiply both sides by the differential dt

$$\frac{dC}{M - kC} = dt$$

- Integrate both sides with respect to the different variables, don't forget the integration constant!

$$\int \frac{dC}{M - kC} = \int dt \Rightarrow -\frac{1}{k} \log |M - kC| = t + A$$

- Solve for the dependent variable $C(t)$

$$\exp(\log |M - kC|) = -\exp(kt + A) \Rightarrow M - kC = Be^{-kt} \Rightarrow$$

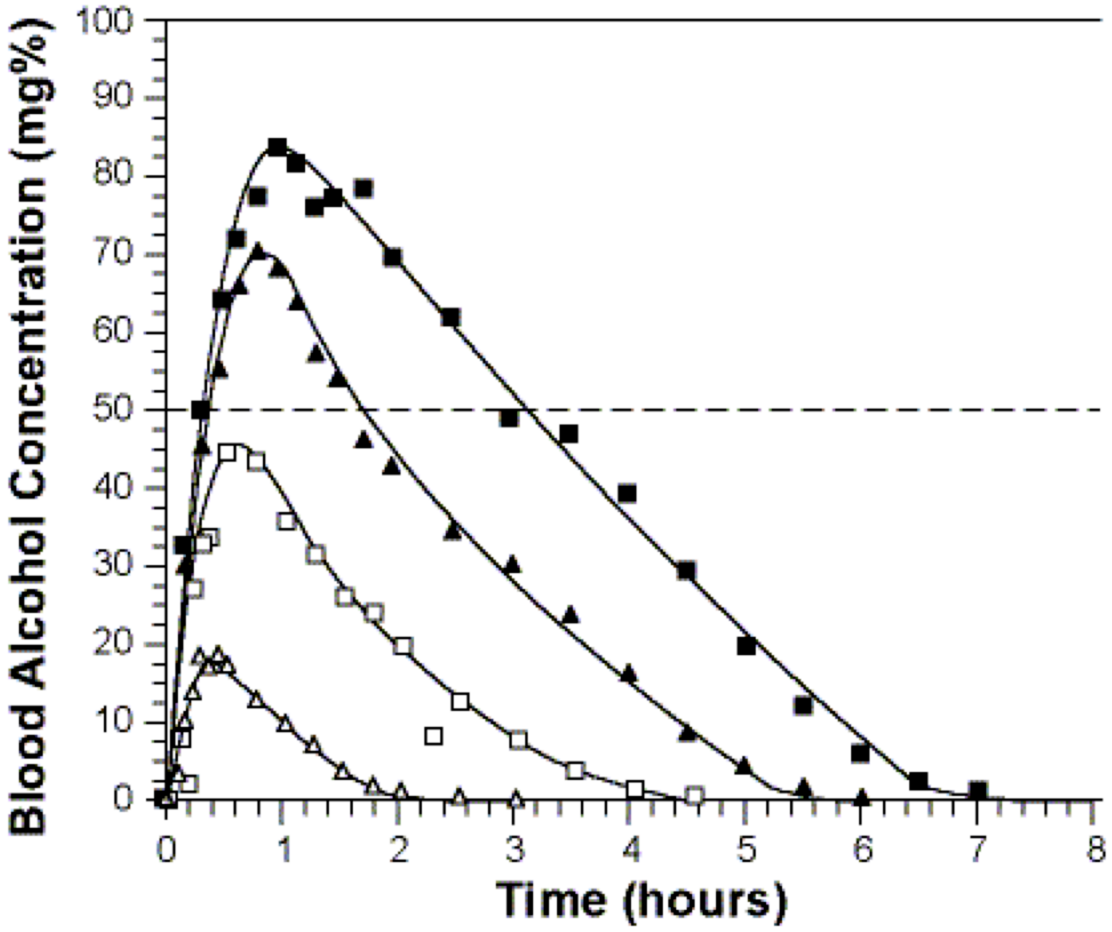


Figure 9.1: Blood alcohol content after ingesting different numbers of drinks, from 4 in the top curve to 1 in the bottom (figure from the National Institute on Alcohol Abuse and Alcoholism in public domain)

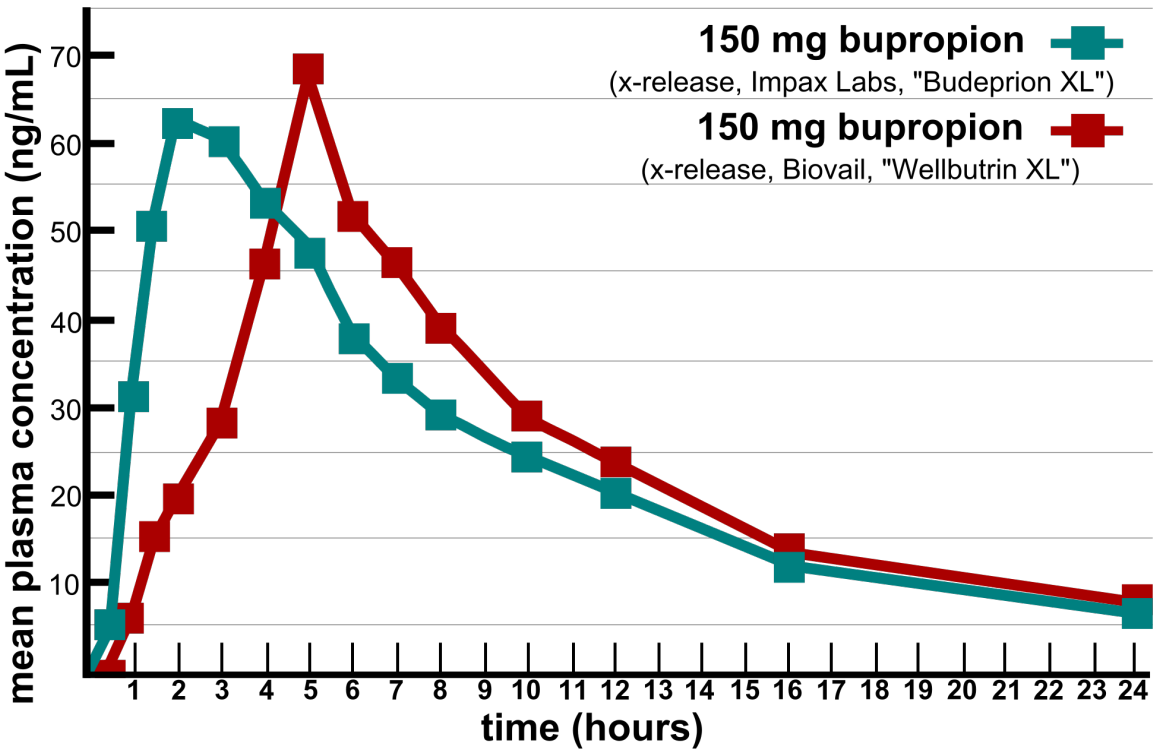


Figure 9.2: Blood concentration of bupropion for two different drugs in clinical trials (image by CMBJ based on FDA data under CC-BY 3.0 via Wikimedia Commons)

$$C(t) = \frac{M}{k} - Be^{-kt}$$

Notice that I changed the values of integration constants A and B during the derivation, which shouldn't matter because they have not been determined yet.

- Plug in $t = 0$ and use the initial value $x(0)$ to solve for the integration constant If we know the initial value $C(0) = C_0$, then we can plug it in and get the following algebraic expression:

$$C_0 = \frac{M}{k} - B \Rightarrow B = C_0 - \frac{M}{k}$$

Then the complete solution is:

$$C(t) = \frac{M}{k} - (C_0 - \frac{M}{k})e^{-kt}$$

The solution predicts that after a long time the plasma concentration will approach the value M/k, since the exponential term decays to zero. Notice that mathematically this is the same type of solution we obtained in equation {eq}`sol-nonhom` for a generic linear ODE with a constant term.

9.3 Membrane as electric circuit

In this example we will construct and analyze a model of electric potential across a membrane. The potential is determined by the difference in concentrations of charged particles (ions) on the two sides of the phospholipid bilayer, as shown in {numref}`fig-cell-mem`. The ions can flow through specific channels across the membrane, changing the concentration and thus the electric potential. K.S. Cole used principles of electrical circuits to devise the first quantitative model of the membrane voltage (**cole_dispersion_1941?**), which eventually led to more sophisticated models of Hodgkin and Huxley, and others.

To start, we will review the physical concepts and laws describing the flow of charged particles. The amount of charge (number of charged particles) is denoted by Q. The rate of flow of charge per time is called the current: $I = \frac{dQ}{dt}$. Current can be analogized to the flow of a liquid, and the difference in height that drives the liquid flow is similar to the electric potential, or voltage. The relationship between voltage and current is given by *Ohm's law:* $V = IR$ where R is the *resistance* of an electrical conductance, and sometimes we use the *conductance* $g = 1/R$ in the relationship between current and voltage:

$$gV = I$$

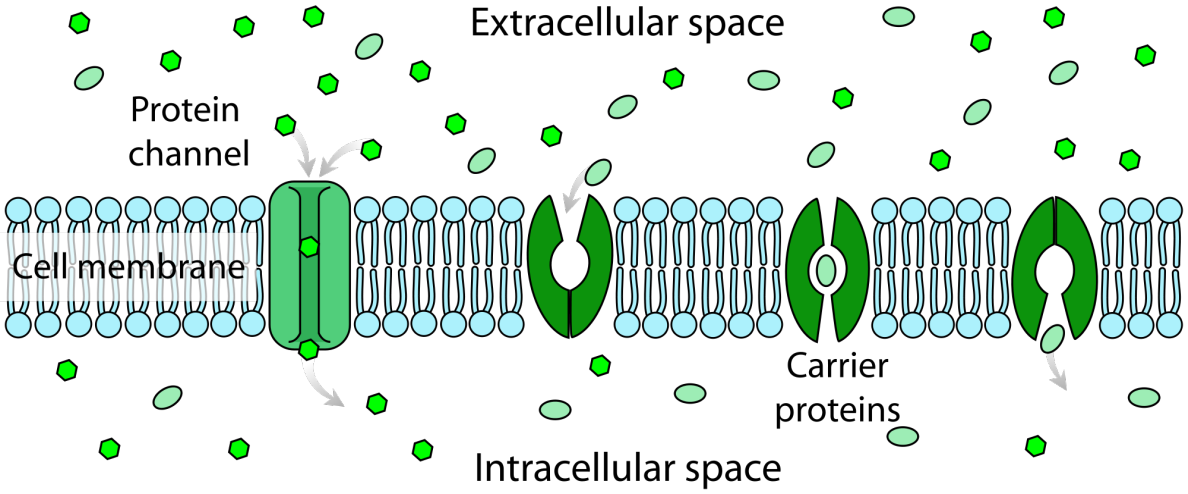


Figure 9.3: Illustration of a cell membrane with ion channels (image by LadyofHats in public domain via Wikimedia Commons)

There are devices known as capacitors, which can store a certain amount of electrical potential in two conducting plates separated by a dielectric (non-conductor). The voltage drop across a capacitor is described by the capacitor law:

$$V_C = \frac{Q}{C}$$

where C is the *capacitance* and Q is the charge of the capacitor.

Lipid bilayer membranes separate media with different concentrations of ions on the two sides, typically the extracellular and cytoplasmic sides. The differences in concentrations of different ions produce a membrane potential. The membrane itself can be thought of as a capacitor, with two charged layers separated by the hydrophobic fatty acid tails in the middle. In addition, there are ion channels that allow ions to flow from the side with higher concentration to that with lower (these are known as passive channels, as opposed to active pumps that can transport ions against the concentration gradient, which we will neglect for now.) These channels are often gated, which means that they conduct ions up to a certain voltage V_R, but then close and reverse direction at higher voltage. The channels are analogous to conducting metal wires, and therefore act as resistors with a specified conductance g. Finally, the electrochemical concentrations of ions act as batteries for each species, (Na^+, K^+, etc.) The overall electric circuit diagram of this model is shown in {numref}`fig-mem-circuit`.

Because the different components are connected in parallel, the total current has to equal the sum of the current passing through each element: the capacitor (membrane) and the gated

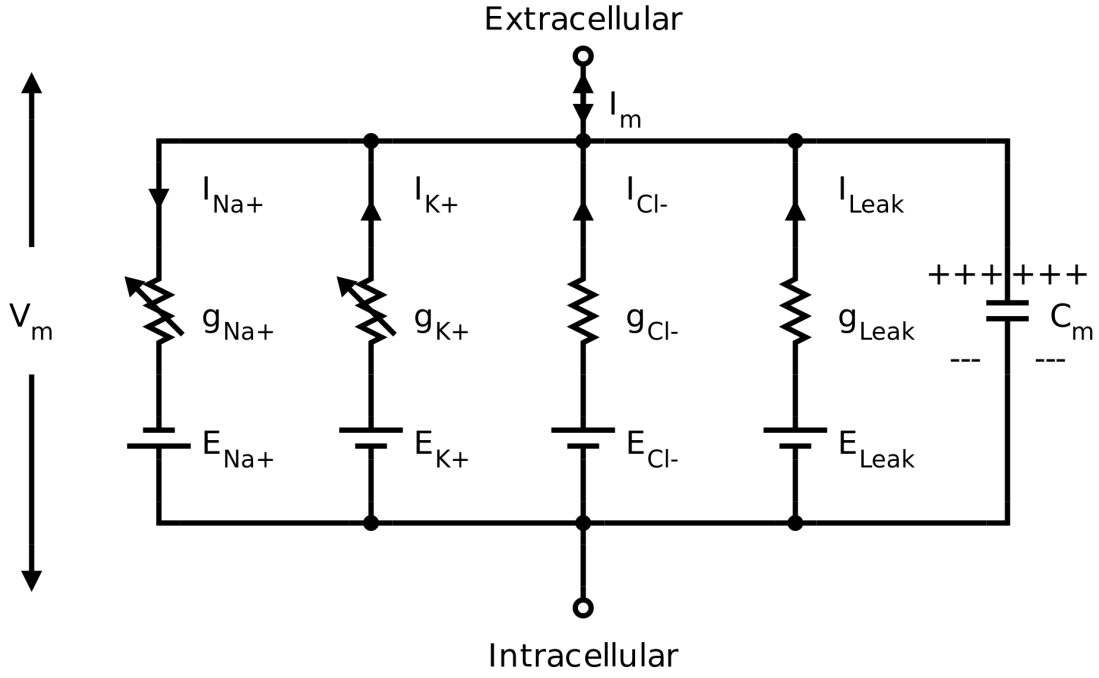


Figure 9.4: Model of ionic flow across a cell membrane as an electric circuit with ion channels as resistors and membrane as a capacitor, (image by NretsNrets under CC BY-SA 3.0 via Wikimedia Commons)

resistors (specific ion channels). The current flowing through a capacitor can be found by differentiating the capacitor law:

$$\frac{dQ}{dt} = I = C\frac{dV_C}{dt}$$

The current flowing through each ion channel is described by this relation: $I = g(V - V_R)$ Then, the total ion flow through the system is described as follows, where i denotes the different ionic species:

$$I_{app} = C\frac{dV}{dt} + \sum_i g(V - V_{Ri})$$

Let us reduce this model to a simple version, where there is no applied current $I_{app} = 0$ and only a single ionic species with reversal potential V_R. Then the differential equation looks like this:

$$\frac{dV}{dt} = -\frac{g}{C}(V - V_{Ri})$$

The Cole membrane potential ODE cam also be solved by the separate-and-integrate method. Dividing both sides by $V - V_R$ and multiplying through by the differential dt, we get:

$$\frac{dV}{V - V_R} = -\frac{g}{C}dt$$

Remember, that V_R is a constant, while $V(t)$ is the dependent variable. Performing the substitution $u = V - V_R$, and integrating, we get $\ln|V - V_R| = -\frac{g}{C}t + A$, where A is the integration constant. Exponentiating both sides and solving for $V(t)$, we obtain:

$$V(t) = V_R + Ae^{-\frac{g}{C}t}$$

So, if the voltage is V_0 at time 0, we have $V_0 = V_R + A \Rightarrow A = V_0 - V_R$. Thus, the specific solution is:

$$V(t) = V_R + (V_0 - V_R)e^{-\frac{g}{C}t}$$

This model predicts that if there is no applied current, then starting at a voltage V_0, the membrane potential will exponentially decay (or grow) to the channel resting (or reversal) potential.

Notice how similar the solutions of the two models are, even thought they are modeling different phenomena. This is an illustration of the power of mathematical modeling, which allows us

to use the same tools to draw general conclusions. This is another illustration of the fact that **solutions of linear ODEs are always exponential in their time dependence**. In cases with a constant rate term, the solutions also include a constant term, to which the solution converges, if the exponential term has a negative constant up in its power.

10 Numeric solutions of ODEs

Analytic solutions are very useful for a modeler because they allow prediction of the variable of interest at any time in the future. However, for many differential equations they are not easy to find, and for many others they simply cannot be written down in a symbolic form. Instead, one can use a numeric approach, which does not require an exact formula for the solution. The idea is to start at a given initial value (e.g. $x(0)$) and use the derivative from the ODE (e.g. dx/dt) as the rate of change of the solution (e.g. $x(t)$) to calculate the change or increment for the solution over a time step. Essentially, this means replacing the continuous change of the derivative with a discrete time step, thus converting the differential equation into a difference equation and then solving it. The solution of the difference equation is not the same as the solution of the ODE, so *numeric solutions* of ODEs are always approximate. I will use the notation $\hat{x}(t)$ to denote the numeric solution to distinguish it from the exact solution $x(t)$. The fundamental difference between them is that $\hat{x}(t)$ is not a formula that can be evaluated at any point in time, but instead is a sequence of numbers calculated every time step, which hopefully are close to the exact solution $x(t)$.

10.0.1 Forward Euler method

Let us introduce all the players: first, we need to pick the time step Δt, which is the length of time between successive values of $\hat{x}$. In the difference equation notation one can use $\hat{x}_i$ to mean $\hat{x}(i\Delta t)$, the value of the numeric solution after i time steps. Then we need to calculate the derivative, or the rate of change at a particular point in time. For any first-order ODE of the form

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

the rate of change depends (potentially) on the values of x and t. This rate of change based on the numeric solution after i time steps is $f(\hat{x}(i\Delta t), i\Delta t) = f(\hat{x}_i, t_i)$. Finally, to calculate the change of the dependent variable we need to multiply the rate of change by the time step. This should make sense in a practical context: if you drive for two hours (time step) at 60 miles per hour (rate of change), the total distance (increment) is $2 * 60 = 120$ miles. By the same token, we can write down how to calculate the next value of the numeric solution y_{i+1} based on the previous one:

$$\hat{x}_{i+1} = \hat{x}_i + \Delta t f(\hat{x}_i, t_i)$$

This method of computing a numeric solution of an ODE is called the *Forward Euler method*, after the famous mathematician who first came up with it. It is called a forward method because it uses the value of the dependent variable and its derivative at time step i to predict the value at the next time step $i + 1$. The method is *iterative*, so it needs to be repeated in order to calculate a set of values of the approximate solution $y(t)$. Here are a couple of simple examples of computing numeric solution using FE:

Example. Let us numerically solve the ODE $\dot{x} = -0.1$ using the Forward Euler method. This means the defining function in the formulation of FE above is $f(x, t) = -0.1$. We can calculate the numeric solution for a couple of steps and compare the values with the exact solution, since we now know that it is $x(t) = x_0 - 0.1t$. Let us pick the time step $\Delta t = 0.2$ and begin with the initial value $x(0) = 1$. Here are the first three steps using the FE method:

$$\begin{aligned}
\hat{x}(0.2) &= \hat{x}(0) + \Delta t f(\hat{x}(0)) = 1 + 0.2 \times (-0.1) = 0.98 \\
\hat{x}(0.4) &= \hat{x}(0.2) + \Delta t f(\hat{x}0.2)) = 0.98 + 0.2 \times (-0.1) = 0.96 \\
\hat{x}(0.6) &= \hat{x}(0.4) + \Delta t f(\hat{x}(0.4)) = 0.96 + 0.2 \times (-0.1) = 0.94
\end{aligned}$$

Since the rate of change in this ODE is constant, the solution declines by the same amount every time step. In this case, the numeric solution is actually exact and perfectly matches the analytic solution.

Example. Let us numerically solve the ODE $\dot{x} = -0.1x$ using the Forward Euler method. This means the defining function in the formulation of FE above is $f(x, t) = -0.1x$. We can calculate the numeric solution for a couple of steps and compare the values with the exact solution, since we now know that it is $x(t) = x_0 e^{-0.1t}$. Let us pick the time step $\Delta t = 0.2$ and begin with the initial value $x(0) = 100$. Here are the first three steps using the FE method:

$$\begin{aligned}
\hat{x}(0.2) &= \hat{x}(0) + \Delta t f(\hat{x}(0)) = 100 + 0.2 \times (-0.1 * 100) = 98 \\
\hat{x}(0.4) &= \hat{x}(0.2) + \Delta t f(\hat{x}(0.2)) = 98 + 0.2 \times (-0.1 * 98) = 96.04 \\
\hat{x}(0.6) &= \hat{x}(0.4) + \Delta t f(\hat{x}(0.4)) = 96.04 + 0.2 \times (-0.1 * 96.04) = \approx 94.12
\end{aligned}$$

In this case, the derivative is not constant and the numeric solution is not exact. The error in the numeric solution grows with time, which may be problematic. We will further investigate how to implement the computation of numeric solutions using R in the next section.

10.0.2 Error in numeric solutions

One of the main concerns of numerical analysis is to minimize the difference between the exact solution and the numeric solution, which is known as the *error*. There are at least two distinct sources of error in numeric solutions: a) *roundoff error* and b) *truncation error*. Roundoff error is caused by computers representing real numbers by a finite string of bits on a computer using what is known as a *floating point* representation. In many programming languages variables storing real numbers can be single or double precision, which typically support 24 and 53 significant binary digits, respectively. Any arithmetic operation involving floating point numbers is only approximate, with an error that depends on the way the numbers are stored in the memory. Truncation error is caused by approximations inherent in numeric algorithms. The most common class of numeric approximations for ODEs is known as *finite difference* methods, and Forward Euler is a very simple representative of that class. As the name suggests, these methods use difference equations to approximate a differential equation. There is inevitably a truncation error in such methods because they use a more or less clever scheme to approximate the instantaneous rate of change in an ODE, which can be thought as a truncation of the Taylor series after certain term.

A modeler has different controls over the roundoff error and truncation error. The first can be minimized by using more memory to store the numbers, e.g. by using double precision format for the variables. Further, there are techniques for minimizing the so-called loss of significance that occurs in certain arithmetic operations, like subtraction of two similar numbers. We will leave these considerations to numerical analysts (**press_numerical_2007?**); for the most part, roundoff error is not a significant issue on modern computers. Truncation error, however, is much more within our control, because it depends on the choice of the numerical algorithm. One can decrease the error in the case of finite difference methods by choosing smaller time steps, or by choosing an algorithm with a higher *order of accuracy*.

Returning specifically to the Forward Euler method, it is called a *first-order method* because the total error of the solution (after some number of time steps) depends linearly on the time step Δt. One can show this by using the Taylor expansion of the solution $\hat{x}(t)$ to derive the forward Euler method, with $\tau(\Delta t)$ representing the truncation error after one time step:

$$\hat{x}(t + \Delta t) = \hat{x}(t) + \Delta t \frac{d\hat{x}(t)}{dt} + \tau(\Delta t)$$

As you might have learned in calculus, the error remaining after the linear term in the Taylor series is proportional to the the square of the small deviation Δt. This only describes the error after 1 time step, but since the errors accumulate every time step, the total error after n time steps accumulates $n\tau(\Delta t)$. As we saw in the implementation above, for a given length of time, n is inversely proportional to Δt. Therefore, the total error is proportional to the Δt and so FE is a first-order method.

The exercise above shows that new errors in FE method accumulate in proportion with the time step. The next question is, what happens to these errors over time? Do they grow or dissipate with more iterations? This is known as the stability of a numerical method, and unlike the above question about the order of accuracy, the answer depends on the particular ODE that one needs to solve. Below I show an example of error analysis for a linear ODE:

Error in the FE scheme

To numerically solve the equation $\dot{x} = ax$, we substitute the function ax for the function $f(x,t)$, and obtain the FE approximation for this particular ODE:

$$\hat{x}_{i+1} = \hat{x}_i + \Delta t a \hat{x}_i = (1 + a\Delta t)\hat{x}_i$$

The big question is what happens to the truncation error: does it grow or decay? To investigate this question, let us denote the error at time t_i , that is the difference between the true solution $x(t_i)$ and the approximate solution $\hat{x}(t_i)$, by ϵ_i. It follows that $\hat{x}_i = x_i + \epsilon_i$. Then we can wrote the following difference equations involving the error:

$$\hat{x}_{i+1} = x_{i+1} + \epsilon_{i+1} = (x_i + \epsilon_i)(1 + a\Delta t) = x_i(1 + a\Delta t) + \epsilon_i(1 + a\Delta t)$$

Let us set aside the terms in the equation that involve x (since it is just the equation for forward Euler). The remaining difference equation for ϵ describes the change in the error:

$$\epsilon_{i+1} = \epsilon_i(1 + a\Delta t)$$

This states that the error in this numeric solution is repeatedly multiplied by the constant $(1 + a\Delta t)$. As we saw in section [sec:math14], this linear difference equation has an exponential solution $\epsilon_n = (1 + a\Delta t)^n \epsilon_0$, which decays to 0 if $|1 + a\Delta t| < 1$ or grows without bound if $|1 + a\Delta t| > 1$. The first inequality is called the stability condition for the FE scheme, since it guarantees that the old errors decay over time. Since $\Delta t > 0$, the only way that the left hand side can be less than 1 is if $a < 0$. Therefore, the condition for stability of the FE method for a linear ODE:

$$|1 + a\Delta t| < 1 \Rightarrow \Delta t < -2/a$$

Thus, if $a > 0$, the errors will eventually overwhelm the solution. If $a < 0$, if the time step is small enough (less than $-2/a$) then FE is stable.

Generally speaking, however, Forward Euler is about the worst method to use for practical numeric solutions of ODEs, due to its low accuracy and to its lack of stability under certain conditions.

10.0.3 Backward Euler method

More sophisticated numeric methods generally offer better stability than Forward Euler. For instance, there is a class of methods called *implicit* schemes which rely on evaluating the value of the derivative of x at a future time point. This may seem impossible, since we do not yet have the value of the dependent variable x in the future, only in the present. In fact, we can set up an algebraic relationship between the present value of x, the future value of x, and the derivative of x in the future. Then, depending on the form of the defining function $f(x)$, we may solve this relationship for the value of x at the future time.

To make the idea of implicit methods concrete, we will introduce a simple method called the Backward Euler. As suggested by the name, this method is essentially similar to the Forward Euler, but with the future value of x_{i+1} substituted in the defining function instead of the current value:

$$\hat{x}_{i+1} = \hat{x}_i + \Delta t \frac{d\hat{x}_{i+1}}{dt} = \hat{x}_i + \Delta t f(\hat{x}_{i+1})$$

How can we calculate the value of $f(\hat{x}_{i+1})$ if you don't know $\hat{x}_{i+1}$? Depending on the form of $f(x)$, it may be possible to algebraically solve for $\hat{x}_{i+1}$. If we can solve the implicit expression for y_{i+1}, we can program a numeric scheme that will compute the value $\hat{x}_{i+1}$ directly from $\hat{x}_i$. In other situations, the implicit expression may be impossible to solve algebraically. The practitioner may then use a method for solving such an expression numerically, using a numerical root-finding algorithm such as Newton's method that we will see later in this course.

Error in the BE scheme

Here is the implementation of the Backward Euler for the linear ODE $\dot{x} = ax$:

$$\hat{x}_{i+1} = \hat{x}_i + \Delta t a \hat{x}_{i+1}$$

For this particular ODE, the implicit equation can be solved for the future value $\hat{x}_{i+1}$:

$$(1 - a\Delta t)\hat{x}_{i+1} = \hat{x}_i \implies \hat{x}_{i+1} = \frac{1}{1 - a\Delta t}\hat{x}_i$$

Now we use the same stability analysis as we did for Forward Euler: assume the numerical solution $\hat{x}_i$ has total error ϵ_i, and substitute $\hat{x}_i = x_i + \epsilon_i$:

$$\hat{x}_{i+1} = x_{i+1} + \epsilon_{i+1} = \frac{1}{1 - a\Delta t}(x_i + \epsilon_i) = \frac{1}{1 - a\Delta t}x_i + \epsilon_i \frac{1}{1 - a\Delta t}$$

Again, let us comapare the numeric solution $\hat{x}$ with the exact solution x and investigate the behavior of the error, which is given by the difference equation:

$$\epsilon_{i+1} = \frac{1}{1 - a\Delta t}\epsilon_i$$

The error decays with time if the multiplicative constant $1/(1 - a\Delta t)$ is less than 1 in absolute value, which can be written as $|1- a t| >1 $. We need to consider two cases: positive a and negative a:

- If $a > 0$, then $|1 - a\Delta t|$ is greater than 1 provided that $\Delta t > 1/|a|$, so the Backward Euler scheme for the exponential growth ODE is stable when Δt is greater than a certain threshold. This appears counterintuitive, so it is worth investigating in the lab.
- If $a < 0$, then $|1 - a\Delta t|$ is greater than 1 for any value of Δt, so it is *unconditionally stable*. This is also worth investigating with numeric experimentation.

```
#Necessary imports
import numpy as np #package for work with arrays and matrices
import matplotlib.pyplot as plt #package with plotting capabilities
```

10.1 Implementation in Python

10.1.1 Forward Euler

We defined the Forward Euler method in the section above, and now we will implement is as a computational algorithm. Like any algorithm, one needs to be clear about its inputs and outputs. In this case, the inputs are the defining function $f(x, t)$, the initial value, the time step, and the total time. The output is the solution vector y, which contains a sequence of values that approximate the solution of the ODE, along with the vector of time values spaced by the time step. Notice that it is very similar to the script for numeric solution of a difference equation we saw in chapter 1 with the major difference being the presence of a time step, whereas in difference equations the time step is aways 1. There is one more important point for the implementation: usually one needs to solve the ODE for a particular length of time T with a specified time step Δt . This dictates that the required number of iterations be $T/\Delta t$; in other words, for a given time period the number of time steps is inversely proportional to the time step.

Outline (pseudocode) for the Forward Euler algorithm

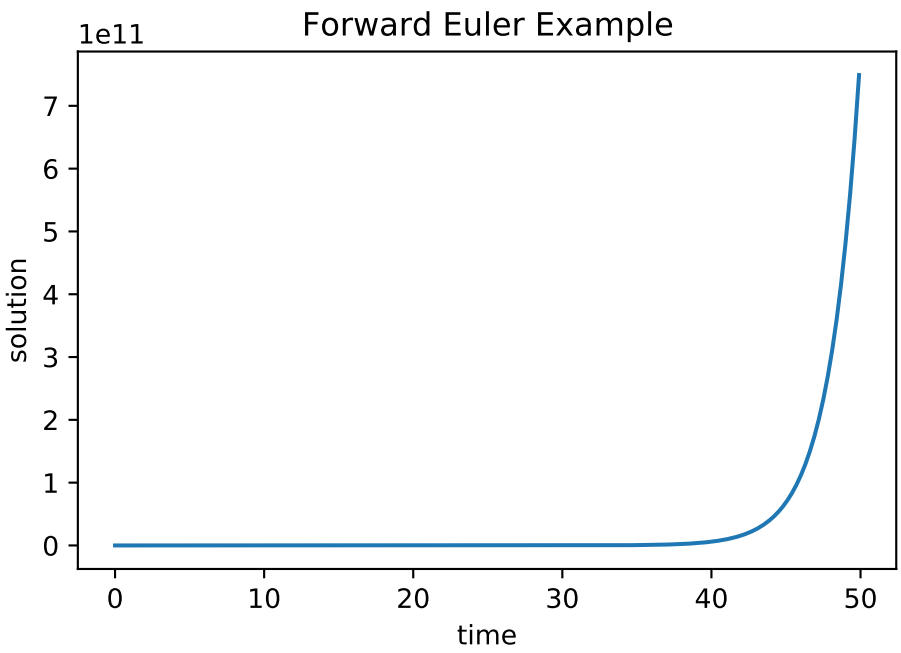
- Specify the defining function for the ODE $f(x)$
- Set the time step dt and the total length of time T
- Calculate the number of steps $n \leftarrow T/dt$
- Initialize the time array t with $n + 1$ elements
- Initialize the solution array x with $n + 1$ elements and initial value x_0

- Use a for loop to compute the next $x(i+1)$ based on the current $x(i)$ for n steps

Below we implement the Forward Euler method to solve the linear ODE

$$dx/dt = r * x$$

```
#Implementation of Forward Euler method to solve dx/dt = r*x

dt = 0.1 #set the time step
T = 50 #set the time duration
Niter = int(np.ceil(T/dt)) #determine the number of iterations
P = np.zeros(Niter) #preallocate the solution array
P[0] = 20 #set the initial value
t = np.arange(0,T,dt) #preallocate the time array
r = 0.5 #set the growth rate

#Do the Euler!
for i in np.arange(Niter-1):
    P[i+1] = P[i] + dt*r*P[i] #this is the FE step

plt.plot(t,P)
plt.xlabel('time')
plt.ylabel('solution')
plt.title('Forward Euler Example')
plt.show()
```



The plot should look like an exponential curve, which seems reasonable, but how accurate is it? Remember from the reading that we can define the error of FE at each point, t, as $|x(t) - \hat{x}(t)| = \epsilon(t)$. Also, we can define the algorithm as stable if the error at some point, t, does not grow so that $|x(t+1) - \hat{x}(t+1)| \leq \epsilon(t)$, where $x(t)$ is the exact solution.

10.1.2 Backward Euler

Now we'll turn to the second method introduced above, Backward Euler (BE). This time, instead of evaluating $f(x, t)$ at the present time for finding the future point, we use the future point itself! In order to do this, we set up an algebraic relationship between the present value, the future value, and the derivative of the future value such that

$$\hat{x}(t + \Delta t) = \hat{x}(t) + dt * f(\hat{x}(t + \Delta t)) + \epsilon(t)$$

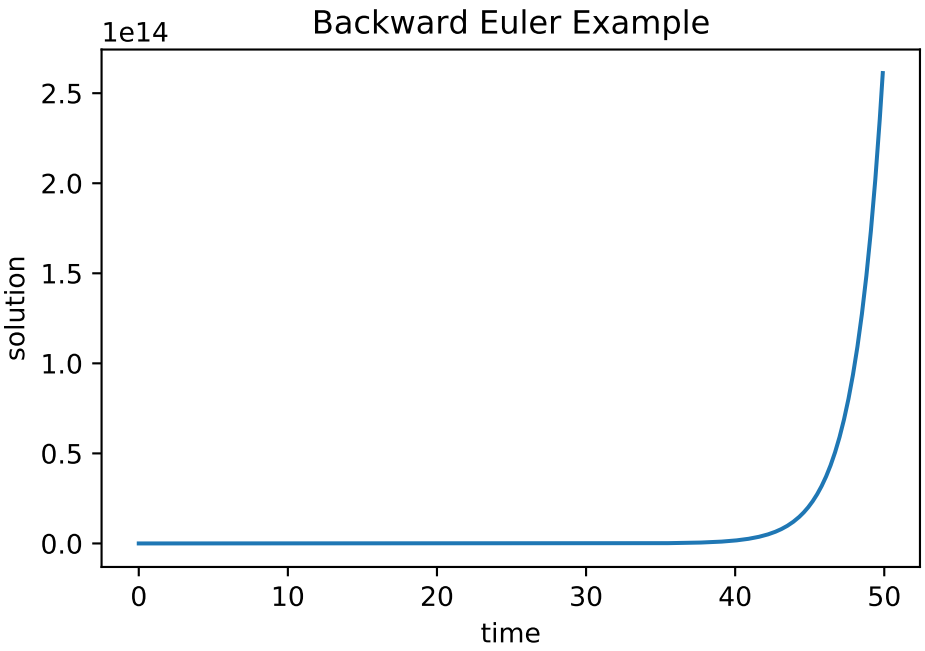
Then, we must solve for $\hat{x}(t + \Delta t)$. Sometimes this will be impossible to do algebraically, but it may be possible to solve the equation numerically. Once we solve for $x(t + 1)$, the steps for implementing the algorithm are similar to the ones for Forward Euler:

Outline (pseudocode) for the Backward Euler algorithm

- Specify the defining function for the ODE $f(x)$
- Set the time step dt and the total length of time T
- Calculate the number of steps $n \leftarrow T/dt$
- Initialize the time array t with $n+1$ elements
- Initialize the solution array x with $n+1$ elements and initial value x_0
- Use a for loop to compute the next $x(i+1)$ based on the current $x(i)$ for n steps

Below is an implementation of the Backward Euler scheme for the generic linear ODE:

```
#Implementation of Backward Euler method to solve dx/dt = r*x

dt = 0.1 #set the time step
T = 50 #set the time duration
Niter = int(np.ceil(T/dt)) #determine the number of iterations
x = np.zeros(Niter) #preallocate the solution array
x[0] = 2000 #set the initial value
t = np.zeros(Niter) #preallocate the time array
r = 0.5 #set the growth rate

#Do the Euler!
for i in range(Niter-1):
    x[i+1] = x[i]/(1-r*dt) #this is the BE step
    t[i+1] = t[i] + dt #add the current time to the time vector

plt.plot(t,x)
plt.xlabel('time')
plt.ylabel('solution')
plt.title('Backward Euler Example')
plt.show()
```


Backward Euler Example
1e14
2.5
2.0
1.5
1.0
0.5
0.0
solution
0
10
20
30
40
50
time

11 Graphical analysis of ordinary differential equations

We now proceed from linear ODEs to more complicated nonlinear equations. In contrast to linear differential equations, which can be solved in general, nonlinear differential equations may not be solvable even theoretically. Even though the solutions cannot be written down, they exist and can exhibit much more interesting behaviors than the exponential solutions we have seen. When solutions cannot be found on paper, we have two options: 1) use qualitative or graphical tools, such as finding equilibrium points and their stability, to predict the long-term dynamics of the solution; 2) construct numerical solutions that approximate the true solution. In this chapter we concentrate on the qualitative approach to analyzing ODEs, which allows one to predict the behavior of solutions of any autonomous ODE based on the graph of the defining function of the equation. In this chapter you will learn to do the following:

- find equilibrium values of an ODE
- analyze the stability of equilibria based on the graph of the defining function
- write down stability conditions analytically
- use graphical techniques to predict the behavior of the solution of a difference equation without solving it
- build basic compartment epidemiology models

11.1 Building nonlinear ODEs

The simple, linear population growth models we have seen in the last chapter assume that the per capita birth and death rates are constant, that is, they stay the same regardless of population size. The solutions for these models either grow or decay exponentially, but in reality, populations do not grow without bounds. It is generally true that the larger a population grows, the more scarce the resources, and survival becomes more difficult. For larger populations, this could lead to higher death rates, or lower birth rates, or both.

How can we incorporate this effect into a quantitative model? We will assume there are separate birth and death rates, and that the birth rate declines as the population grows, while

the death rate increases. Suppose there are inherent birth rates b and d, and the overall birth and death rates B and D depend linearly on population size P: $B = b - aP$ and $D = d + cP$.

To model the rate of change of the population, we need to multiply the rates B and D by the population size P, since each individual can reproduce or die. Also, since the death rate D decreases the population, we need to put a negative sign on it. The resulting model is:

$$\dot{P} = BP - DP = [(b - d) - (a + c)P]P$$

(logistic-ode)

The parameters of the model, the constants a, b, c, d, have different meanings. Performing dimensional analysis, we find that b and d have the dimensions of $1/[t]$, the same as the rate r in the exponential growth model. However, the dimensions of a (and c) must obey the relation: $[P]/[t] = [a][P]^2$, and thus,

$$[a] = [c] = \frac{1}{[t][P]}$$

This shows that the constants a and c have to be treated differently than b and d. Let us define the inherent growth rate of the population, to be $r_0 = b - d$ (if the death rate is greater than the birth rate, the population will inherently decline). Then let us introduce another constant K, such that $(a + c) = r_0/K$. It should be clear from the dimensional analysis that K has units of P, population size. Now we can write down the logistic equation in the canonical form:

$$\dot{P} = r\left(1 - \frac{P}{K}\right)P$$

This model can be re-written as $\dot{P} = aP - bP^2$, so it is clear that there is a *linear term* (aP) and a *nonlinear term* ($-bP^2$). When P is sufficiently small(and positive) the linear term is greater, and the population grows. When P is large enough, the nonlinear term wins and the population declines.

It should be apparent that there are two fixed points, at $P = 0$ and at $P = K$. The first one corresponds to a population with no individuals. On the other hand, K signifies the population at which the negative effect of population size balances out the inherent population growth rate, and is called the *carrying capacity* of a population in its environment (**otto_biologists_2007?**). We will analyze the qualitative behavior of the solution, without writing it down, in the next section of this chapter.

11.2 Qualitative analysis of ODEs

In this section we will analyze the behavior of solutions of an autonomous ODE without solving it on paper. Generally, ODE models for realistic biological systems are nonlinear, and most nonlinear differential equations cannot be solved analytically. We can make predictions about the behavior, or *dynamics* of solutions by considering the properties of the *defining function*, which is the function on the right-hand-side of a general autonomous ODE:

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

11.2.1 graphical analysis of the defining function

The defining function relates the value of the solution variable x to its rate of change dx/dt. For different values of x, the rate of change of $x(t)$ is different, and it is defined by the function $f(x)$. There are only three options:

- if $f(x) > 0$, $x(t)$ is increasing at that value of x
- if $f(x) < 0$, $x(t)$ is decreasing at that value of x
- if $f(x) = 0$, $x(t)$ is not changing that value of x

To determine for which values of x the solution $x(t)$ increases and decreases, it enough to look at the plot of $f(x)$. On the intervals where the graph of $f(x)$ is above the x-axis $x(t)$ increases, on the intervals where the graph of $f(x)$ is below the x-axis, $x(t)$ decreases. The roots (zeros) of $f(x)$ are special cases, they separate the range of x into the intervals where the solution grows and and where it decreases. This seems exceedingly simple, and it is, but it provides specific information about $x(t)$, without knowing how to write down its formula.

For an autonomous ODE with one dependent variable, the direction of the rate of change prescribed by the differential equation can be graphically represented by sketching the *flow on the line* of the dependent variable. The flow stands for the direction of change at every point, specifically increasing, decreasing, or not changing. The flow is plotted on the horizontal x-axis, so if x is increasing, the flow will be indicated by a rightward arrow, and if it is decreasing, the flow will point to the left. The fixed points separate the regions of increasing (rightward) flow and decreasing (leftward) flow.

Example. Consider a linear ODE the likes of which we have solved in section

$$\frac{dx}{dt} = 4x - 100$$

The defining function is a straight line vs. x, its graph is shown in figure **??**a. Based on this graph, we conclude that the solution decreases when $x < 25$ and increases when $x > 25$. Thus

we can sketch the solution $x(t)$ over time, without knowing its functional form. The dynamics depends on the initial value: if $x(0) < 25$, the solution will keep decreasing without bound, and go off to negative infinity; if $x(0) > 25$, the solution will keep decreasing without bound, and go off to positive infinity. This is shown by plotting numerical solutions of this ODE for several initial values in figure **??**b. The dotted line shows the location of the special value of 25 which separates the interval of growth from the interval of decline.

Example. Now let us analyze a nonlinear ODE, specifically the logistic model with the following parameters:

$$\frac{dP}{dt} = 0.3P\left(1 - \frac{P}{40}\right)$$

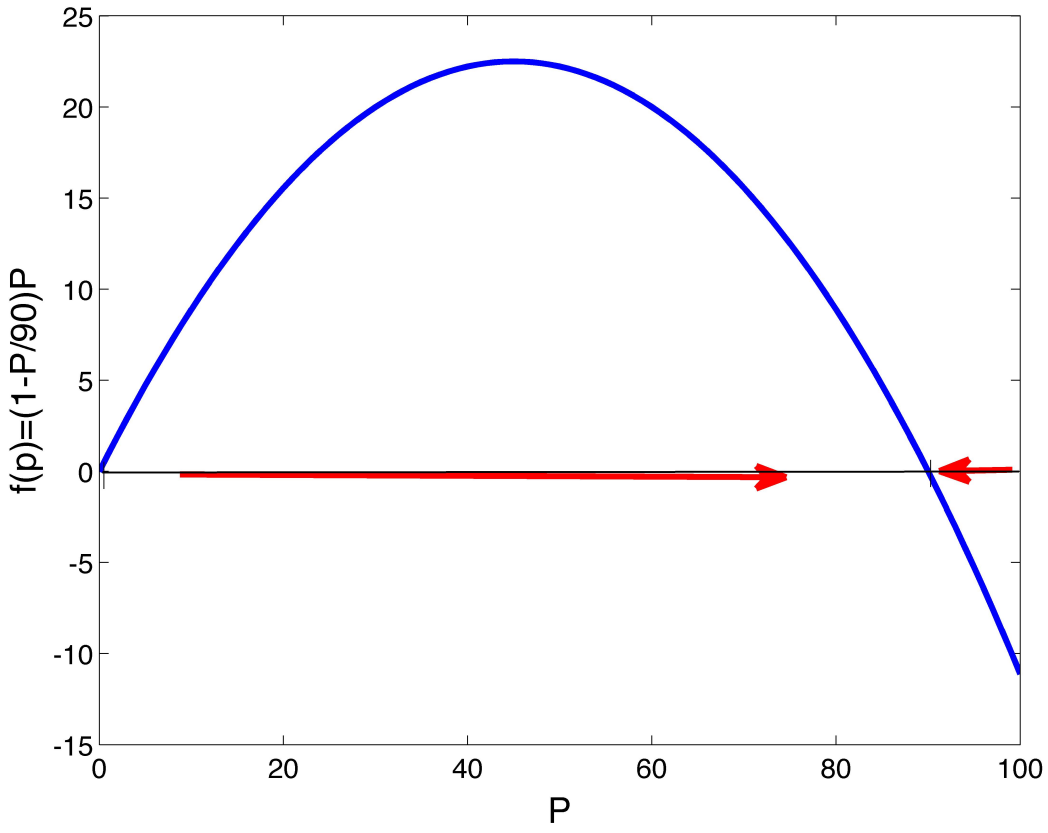


Figure 11.1: Flow diagram of the logistic model $\dot{P} = (1 - P/90)P$; red arrows indicate the direction field in the intervals separated by the fixed points

The defining function is a downward-facing parabola with two roots at $P = 0$ and $P = 90$, as shown in {numref}`fig-flow-log`. Between the two roots, the defining function is positive,

which means the derivative dP/dt is positive too, so the solution grows on that interval and approaches the value 90. For $P < 0$ the solution decreases without bound; for $P > 90$ the solution also decreases and converges to 90, since a solution cannot go through a value at which its derivative is 0.

Example: semi-stable fixed point. Let un analyze another nonlinear ODE

$$\frac{dx}{dt} = -x^3 + x^2$$

The flow of the solutions is plotted in {numref}`fig-flow-semi`, showing two fixed points at $x = 0, 1$. The red arrows on the x-axis show the direction of the flow in the three different regions separated by the zeros of $f(x)$. For $x < 0$, solutions decrease without bound; for $0 < x < 1$ solutions increase and approach 1 and for $x > 1$ solutions decrease and approach 1.

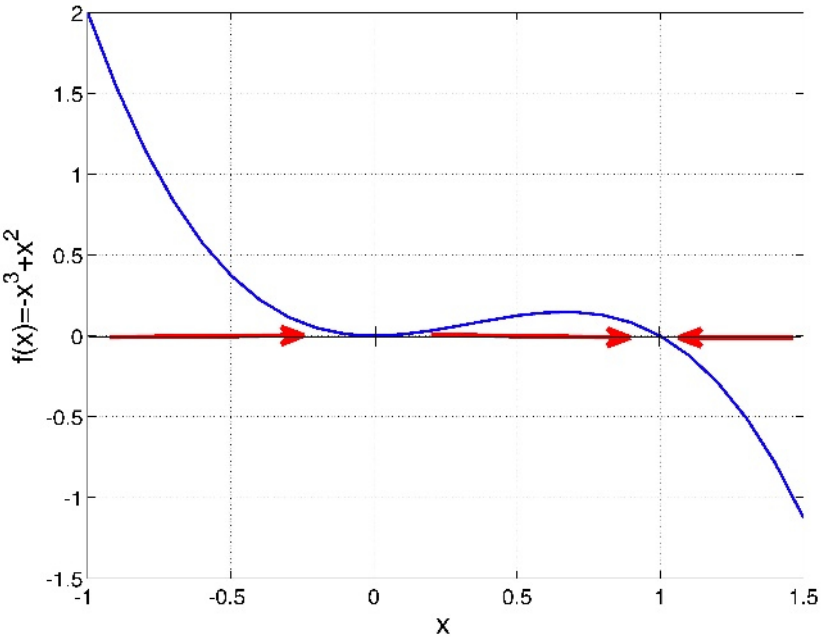


Figure 11.2: Flow diagram of the nonlinear ODE $\dot{x} = -x^3 + x^2$; red arrows indicate the direction field in the intervals separated by the fixed points

To summarize, the defining function of the ODE determines the rate of change of the solution $x(t)$ depending on the value of x. The graphical approach to finding areas of right and left flow is based on graphing the function $f(x)$, and dividing the x-axis based on the sign of $f(x)$. In the areas where $f(x) > 0$, its graph is above the x-axis, and the flow is to the right; conversely, when $f(x) < 0$, its graph is below the x-axis, and the flow is to the left. The next subsection puts this approach in a more analytic framework.

11.2.2 fixed points and stability

We have seen that the dynamics of solutions of differential equations depend on the initial value of the dependent variable: for some values the solution increases, for others it decreases, and for intermediate values it remains the same. Those special values separating intervals of increase and decrease are called fixed points (or equilibria), and the first step to understanding the dynamics of an ODE is finding its fixed points. A fixed point is a value of the solution at which the dynamical system stays constant, thus, the derivative of the solution must be zero. Here is the formal definition:

> **i** Definition
>
> For an ordinary differential equation $\dot{x} = f(x)$, a point x^* which satisfies $f(x^*) = 0$ is called a *fixed point* or *equilibrium*, and the solution with the initial condition $x(0) = x^*$ is constant over time $x(t) = x^*$.

Example. The linear equation $\dot{x} = rx$ has a single fixed point at $x^* = 0$. For a more interesting example, consider a logistic equation: $\dot{x} = x - x^2$. Its fixed points are the solutions of $x - x^2 = 0$, therefore there two fixed points: $x^* = 0, 1$. We know that if the solution has either of the fixed points as the initial condition, it will remain at that value for all time.

Locating the fixed points is not sufficient to predict the global behavior of the dynamical system, however. What happens to the solution of a dynamical system if the initial condition is very close to an equilibrium, but not precisely at it? Put another way, what happens if the equilibrium is perturbed? The solution may be attracted to the equilibrium value, that is, it approaches it ever-closer, or else it is not. In the first case, this is called a stable equilibrium, because a small perturbation does not dramatically change the long-term behavior of the solution. In the latter case, the equilibrium is called unstable, and the solution perturbed from the equilibrium never returns. These concepts are formalized in the following definition

> **i** Definition
>
> A fixed point x^* of an ODE $\dot{x} = f(x)$ is called a *stable* fixed point (or sink) if for a sufficiently small number ϵ, the solution $x(t)$ with the initial condition $x_0 = x^* + \epsilon$ approaches the fixed point x^* as $t \to \infty$. If the solution $x(t)$ does not approach x^* for all nonzero ϵ, the fixed point is called an *unstable* fixed point (or source).

To determine whether a fixed point is stable analytically we use the approach called *linearization*, which involves replacing the function $f(x)$ with a linear approximation. Let us define $\epsilon(t)$ to be the deviation of the solution $x(t)$ from the fixed point x^*, so we can write $x(t) = x^* + \epsilon(t)$. Assuming that $\epsilon(t)$ is small, we can write the function $f(x)$ using Taylor's formula:

$$f(x^* + \epsilon(t)) = f(x^*) + f'(x^*)\epsilon(t) + ... = f'(x^*)\epsilon(t) + ...$$

The term $f(x^*)$ vanished because it is zero by definition **??** of a fixed point. The ellipsis indicates all the terms of order $\epsilon(t)^2$ and higher, which are very small if $\epsilon(t)$ is small, and thus can be neglected. Thus, we can write the following approximation to the ODE $\dot{x} = f(x)$ near a fixed point:

$$\dot{x} = \frac{d(x^* + \epsilon(t))}{dt} = \dot{\epsilon}(t) = f'(x^*)\epsilon(t)$$

Thus we replaced the complicated nonlinear ODE near a fixed point with a linear equation, which approximates the dynamics of the deviation $\epsilon(t)$ near the fixed point x^*; note that the derivative $f'(x^*)$ is a constant for any given fixed point. In section **??** we classified the behavior of solutions for the general linear ODE $\dot{x} = rx$, and now we apply this classification to the behavior of the deviation $\epsilon(t)$. If the multiple $f'(x^*)$ is positive, the deviation $\epsilon(t)$ is growing, the solution is diverging away from the fixed point, and thus the fixed point is unstable. If the multiple $f'(x^*)$ is negative, the deviation $\epsilon(t)$ is decaying, the solution is converging to the fixed point, and thus the fixed point is stable. Finally, there is the borderline case of $f'(x^*) = 0$ which is inconclusive, and the fixed point may be either stable or unstable. The derivative stability analysis is summarized in the following:

Important Fact

For an autonomous ODE in the form $\dot{x} = f(x)$, a fixed point x^* can be classified as follows:

- $f'(x^*) > 0$: the slope of $f(x)$ at the fixed point is positive, then the fixed point is **unstable**.
- $f'(x^*) < 0$: the slope of $f(x)$ at the fixed point is negative, then the fixed point is **stable**.
- $f'(x^*) = 0$: stability cannot be determined from the derivative.

Therefore, knowing the derivative or the slope of the defining function at the fixed point is enough to know its stability. If the derivative has the discourtesy of being zero, the situation is tricky, because then higher order terms that we neglected make the difference. We will mostly avoid such borderline cases, but they are important in some applications (**strogatz_nonlinear_2001?**).

Warning

The derivative of the defining function $f'(x)$ is not the second derivative of the solution $x(t)$. This is a common mistake, because the function $f(x)$ is equal to the time derivative of $x(t)$. However, the derivative $f'(x)$ is not with respect to time, it is with respect to x, the dependent variable. In other words, it reflects the slope of the graph of the defining

function $f(x)$, not the curvature of the graph of the solution $x(t)$.

11.2.3 outline of qualitative analysis of an ODE

To summarize, here is an outline of the steps for analyzing the behavior of solutions of an autonomous one-variable ODE. These tasks can be accomplished either by plotting the defining function $f(x)$ and finding the fixed points and their stability based on the plot, or by solving for the fixed points on paper, then finding the derivative $f'(x)$ and plugging in the values of the fixed points to determine their stability. Either approach is valid, but the analytic methods are necessary when dealing with models that have unknown parameter values, which makes it impossible to represent the defining function in a plot.

Analyzing the flow and stability of solutions

- find the fixed points by setting the defining function $f(x) = 0$ and solving for values of x^*
- divide the domain of x into intervals separated by fixed points x^*
- determine on which interval(s) the solution $x(t)$ is increasing and on which it is decreasing
- use derivative stability analysis (graphically or analytically) to determine which fixed points are stable
- sketch the solutions $x(t)$ starting at different initial values, based on the stability analysis and whether the solution is increasing or decreasing in a particular interval

ODE analysis example

Analysis of linear ODE $dx/dt = 4x - 100$

- find the fixed points by setting the defining function to 0: $0 = 4x - 100$, so there is only one fixed point $x^* = 25$
- divide the domain of x into intervals separated by fixed points x^*: the intervals are $x < 25$ and $x > 25$
- the solution is decreasing on the interval $x < 25$ because $f(x) < 0$ there, and the solution is increasing on the interval $x > 25$ because $f(x) > 0$
- the derivative $f'(x)$ at the fixed point is 4, so the fixed point is *unstable*

- solutions $x(t)$ behave as follows: solutions with initial values below $x^* = 25$ decreasing, and those with initial values above $x^* = 25$ increasing.

ODE analysis example

Analysis of the logistic ODE $dP/dt = 0.3P(1 - P/40)$

- find the fixed points by setting the defining function to 0: $0 = 0.3P(1-P/40)$. The two solutions are $P^* = 0$ and $P^* = 40$.
- divide the domain of P into intervals separated by fixed points P^*: the intervals are $P < 0$; $0 < P < 40$; and $P > 40$
- the solution is decreasing on the interval $P < 0$ because $f(P) < 0$ there, the solution is increasing on the interval $0 < P < 40$ because $f(P) > 0$, and the solution is decreasing for $P > 40$ because $f(P) < 0$ there
- the derivative is $f'(P) = 0.3 - 0.3P/20$; since $f'(0) = 0.3 > 0$, the fixed point is *unstable*; since $P'(40) = -0.3 < 0$, the fixed point is *stable*
- solutions $P(t)$ behave as follows: solutions with initial values below $P^* = 0$ decreasing, those with initial values between 0 and 40 are increasing and asymptotically approaching 40, and those with initial values above 40 decreasing and asymptotically approaching 40.

This can be done more generally using the derivative test: taking the derivative of the function on the right-hand-side (with respect to P), we get $f'(P) = r(1 - 2\frac{P}{K})$. Assuming $r > 0$ (the population is viable), $f'(0) = r$ is positive, and the fixed point is therefore unstable. This makes biological sense, since we assumed positive inherent population growth, so given a few individuals, it will increase in size. On the other hand, $P'(K) = r(1 - 2) = -r$, so this fixed point is stable. Thus, according to the logistic model, a population with a positive inherent growth rate will not grow unchecked, like in the exponential model, but will increase until it reaches its carrying capacity, at which it will stay (if all parameters remain constant).

ODE analysis example

Analysis of the ODE $dx/dt = -x^3 + x^2$ that has a semi-stable fixed point

- find the fixed points by setting the defining function to 0: $0 = -x^3 + x^2$. The two fixed points are $x^* = 0$ and $x^* = 1$.
- divide the domain of x into intervals separated by fixed points x^*: the intervals are $x < 0$; $0 < x < 1$; and $x > 1$

- the solution is increasing on the interval $x < 0$ because $f(x) > 0$ there, the solution is increasing on the interval $0 < x < 1$ because $f(x) > 0$, and the solution is decreasing for $x > 1$ because $f(x) < 0$ there
- the derivative is $f'(x) = -3x^2 + 2x$; since $f'(0) = 0$, the fixed point is *undetermined*; since $f'(1) = -1 < 0$, the fixed point is *stable*.
- the solutions $x(t)$ with initial values below 0 are increasing and asymptotically approaching 0, those with initial values between 0 and 1 are increasing and asymptotically approaching 1, and those with initial values above 1 are decreasing and asymptotically approaching 1.

This example shows how graphical analysis can help when derivative analysis is undetermined. The red arrows on the x-axis of {numref}`fig-flow-semi` show the direction of the flow in the three different regions separated by the fixed points. Flow is to the right for $x < 1$, to the left for for $x > 1$; it is clear that the arrows approach the fixed point from both sides, and thus the fixed point is stable, as the negative slope of $f(x)$ at $x = 1$ indicates. One the other hand, the fixed point at $x = 0$ presents a more complicated situation: the slope of $f(x)$ is zero, and the flow is rightward on both sides of the fixed point. This type of fixed point is sometimes called *semi-stable*, because it is stable when approached from one side, and unstable when approached from the other.

11.3 Modeling the spread of infectious disease

The field of *epidemiology* studies the distribution of disease and health states in populations. Epidemiologists describe and model these issues with the goal of helping public health workers devise interventions to improve the overall health outcomes on a large scale. One particular topic of interest is the the spread of infectious disease and how best tor respond to it.. Because epidemiology is concerned with large numbers of people, the models used in the field do not address the details of an individual disease history. One approach to modeling this is to put people into categories, such as *susceptible* (those who can be infected but are not), *infectious* (those who are infected and can spread the disease), and *recovered* (those who cannot be infected or spread disease, shown in {numref}`fig-sir-model`. This type of models is called a *compartment model* and they are they commonly used to represent infectious disease on a population level both for deterministic models (e.g. ODEs) and stochastic models (e.g. Markov models). Dividing people into categories involves the assumption that everyone in a particular category behaves in the same manner: for instance, all susceptible people are infected with the same rate and all infected people recover with the same rate.

Let us construct an ODE to describe a two-compartment epidemiology model. There are two dependent variables to be tracked: the number of susceptible (S) and infected (I) individuals in the population. The susceptible individuals can get infected, while the infected ones can

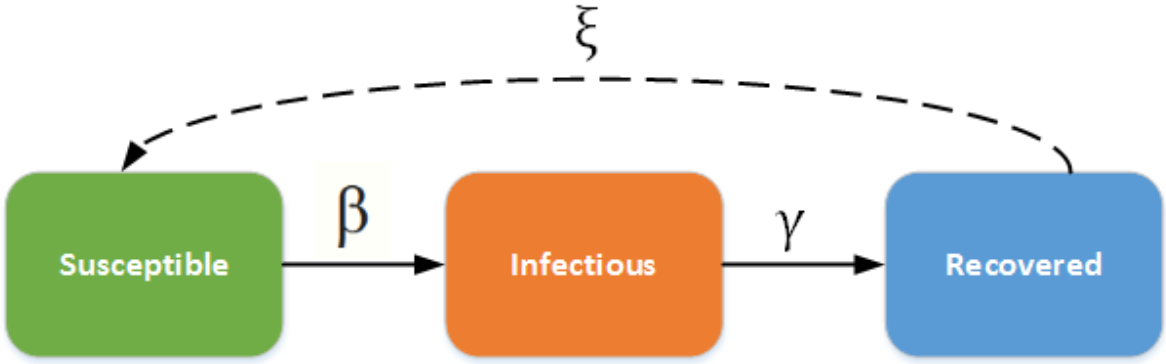


Figure 11.3: Diagram for the classic SIR model, with β the infectivity rate (per encounter between one S and one I) and γ is the recovery rate (per I) https://idmod.org/docs/emod/generic/model-sir.html

recover and become susceptible again. The implicit assumption is that there is no immunity, and recovered individuals can get infected with the same ease as those who were never infected. There are some human diseases for which this is true, for instance the common cold or gonorrhea. Transitions between the different classes of individuals can be summarized by the scheme in figure

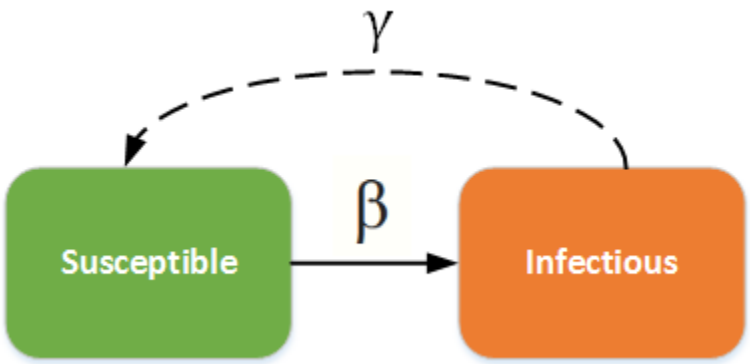


Figure 11.4: Diagram for the SIS model, where there is no possibility of recovery with immunity. As in the SIR model, β is the infectivity rate (per encounter between one S and one I) and γ is the recovery rate (per I) https://idmod.org/docs/emod/generic/model-sir.html

Here β is the individual rate of infection, also known as the transmission rate, and γ is the individual rate of recovery. There is an important distinction between the processes of infection and recovery: the former requires an infected individual and a susceptible individual, while the latter needs only an infected individual. Therefore, it is reasonable to suppose that the

rate of growth of infected individuals is the product of the individual transmission rate β and the product of the number of infected and susceptible individuals. The overall rate of recovery is the individual recovery rate γ multiplied by the number of the infected. This leads to the following two differential equations:

$$\begin{aligned} \dot{S} = & \quad -\beta IS + \gamma I \\ \dot{I} = & \quad \beta IS - \gamma I \end{aligned}$$

Note that, as in the chemical kinetics models, the two equations add up to zero on the right hand side, leading to the conclusion that $\dot{S} + \dot{I} = 0$. Therefore, the total number of people is a conserved quantity N, which does not change. This makes sense since we did not consider any births or deaths in the ODE model, only transitions between susceptible and infected individuals.

We can use the conserved quantity N to reduce the two equations to one, by the substitution of $S = N - I$:

$$\dot{I} = \beta I(N - I) - \gamma I$$

This model may be analyzed using qualitative methods that were developed in this chapter, allowing prediction of the dynamics of the fraction of infected for different transmission and recovery rates. First, let us find the fixed points of the differential equation. Setting the equation to zero, we find:

$$0 = \beta I(N - I) - \gamma I \Rightarrow I^* = 0;\ I^* = N - \gamma/\beta$$

This means that there are two equilibrium levels of infection: either nobody is infected ($I^* = 0$) or there is some persistent number of infected individuals ($ I^* = N - / $). Notice that the second fixed point is only biologically relevant if $N > / $.

Use the derivative test to check for stability. First, find the general expression for derivative of the defining function: $f'(I) = -2 I + N - $.

The stability of the fixed point $I^* = 0$ is found by plugging in this value into the derivative formula: $f'(0) = \beta N - \gamma$. We learned in section **??** that a fixed point is stable if the derivative of the defining function is negative. Therefore, $I^* = 0$ is stable if $\gamma - \beta N > 0$, and unstable otherwise. This gives us a *stability condition* on the values of the biological parameters. If the recovery rate γ is greater than the rate of infection for the population (the transmission rate multiplied by the population size) βN, then the no-infection equilibrium is stable. This predicts that the infection dies out if the recovery rate is faster than the rate of infection, which makes biological sense.

Similarly, we find the stability of the second fixed point $I^* = N - \gamma/\beta$ by substituting its value into the derivative, to obtain $f'(N - \gamma/\beta) = \gamma - \beta N$. By the same logic, as above, this fixed point is stable if $\gamma - \beta N < 0$, or if $\gamma < \beta N$. This is a complementary condition for the fixed point at 0, that is, only one fixed point can be stable for any given parameter values. In the biological interpretation, if the transmission rate βN is greater than the recovery rate γ, then the epidemic will persist.

We can use our graphical analysis skills to illustrate the situation. Consider a situation in which $\gamma < \beta N$. As predicted by stability analysis, the zero infection equilibrium should be unstable, and the equilibrium at $N - \gamma/\beta$ should be stable. In order to plot the function $f(I) = I (N - I) - I$, we choose the specific parameter values and plot the defininining function in {numref}`fig-sis-beta`. The direction of the flow on the I-axis prescribed by the defining function $f(I)$ shows that solutions approach the fixed point at $N - \gamma/\beta$ from both directions, which make it a stable fixed point, while diverging from $I = 0$.

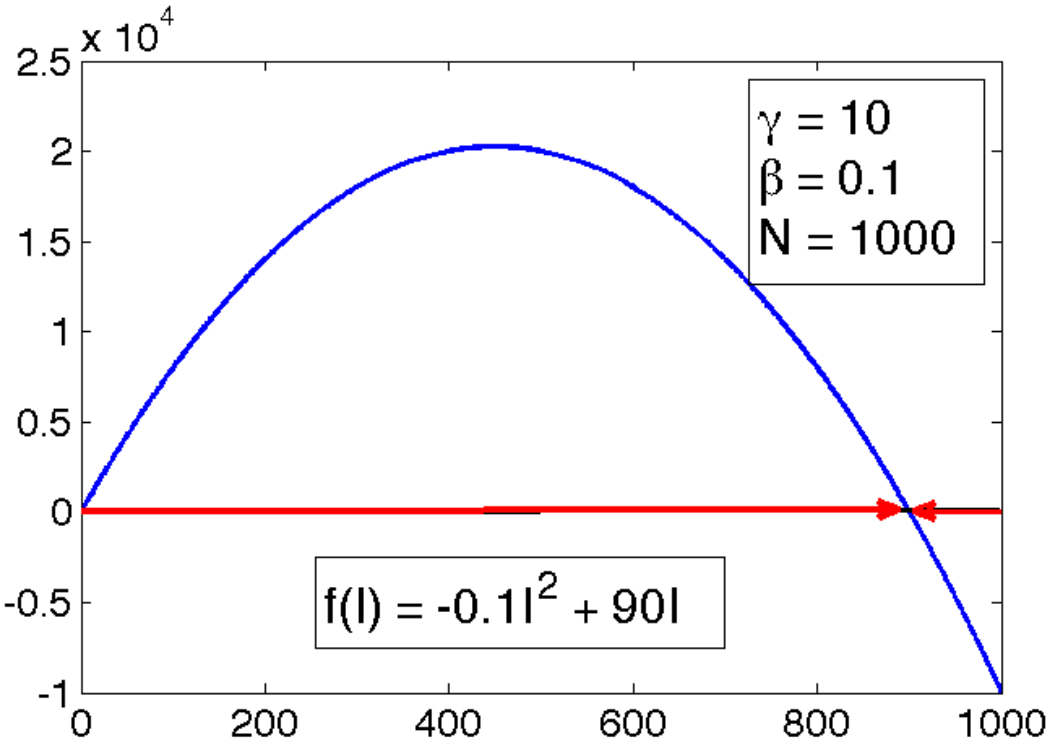


Figure 11.5: Graphical analysis of the SIS model with population $N = 1000$ individuals and $\gamma < \beta N$. The plots show the graph of the defining function of the variable I (blue) and the flow on the I-axis (red.)

On the other hand, if $\gamma > \beta N$, stability analysis predicts that the no-infection equilibrium ($I = 0$) is stable. {numref}`fig-sis-gamma` shows the plot of the defining function where the

effective infection rate βN is greater than the recovery rate γ. The flow on the I-axis is toward the zero equilibrium, therefore it is stable. Note that the second equilibrium at $I^* = N - \gamma/\beta$ is negative, and thus has no biological significance. The solutions, if the initial value is positive, all approach 0, so the infection inevitably dies out.

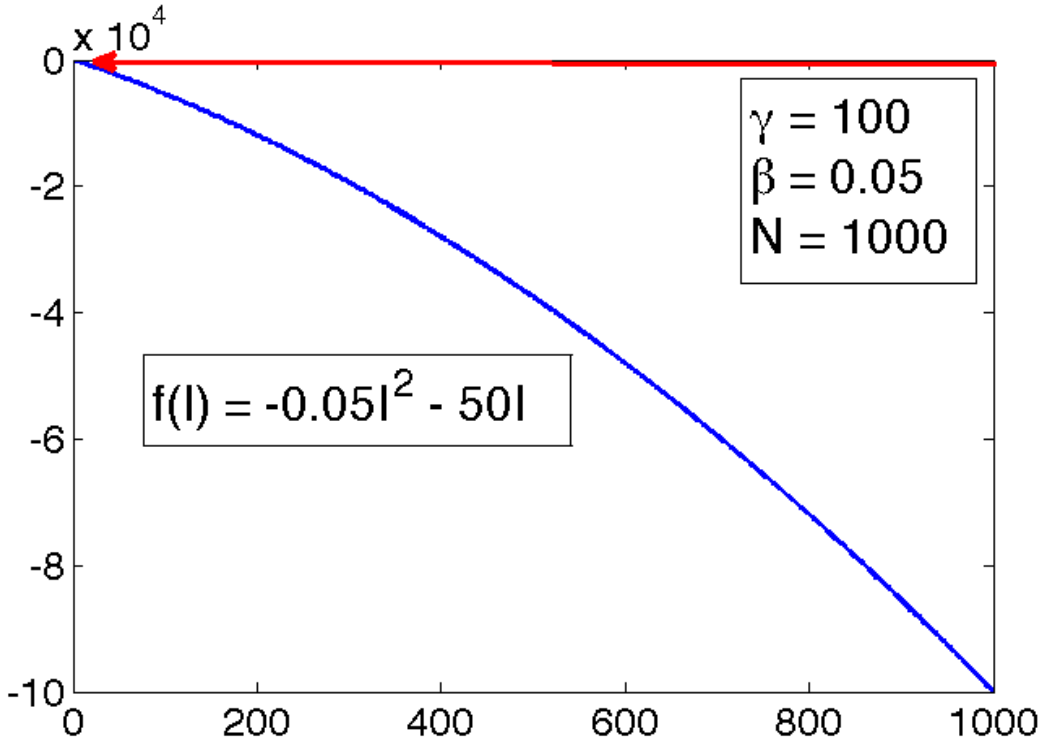


Figure 11.6: Graphical analysis of the SIS model with population $N = 1000$ individuals and $\gamma < \beta N$. The plots show the graph of the defining function of the variable I (blue) and the flow on the I-axis (red.)

Mathematical modeling of epidemiology has been a success story in the last few decades. Public health workers routinely estimate the parameter called the *basic reproductive number* R_0 defined to be the average number of new infections caused by a single infected individual in a susceptible population. This number comes out of our analysis above, where we found $R_0 = N\beta/\gamma$ to determine whether or not an epidemic persisted (**brauer_mathematical_2011?**). This number is critical in more sophisticated models of epidemiology.

Mathematical models are used to predict the time course of an epidemic, called the *epidemic curve* and then advise on the public health interventions that can reduce the number of affected individuals. In reality, most epidemic curves have the shape similar to the data from the Ebola virus epidemic in {numref}`fig-ebola1` and {numref}`fig-ebola2`. Most such curves show an initial increase in infections, peaking, and the declining to low levels, which is fun-

damentally different than the solution curves we obtained from the two-compartment model. To describe dynamics of this nature, models with more than two variables are needed, such as classic three-compartment SIR models (susceptible-infected-recovered) models and their modifications (**brauer_mathematical_2011?**). Being able to predict the future of an epidemic based on R_0 and other parameters allows public health officials to prepare and deploy interventions (vaccinations, quarantine, etc.) that have the best shot at minimizing the epidemic.

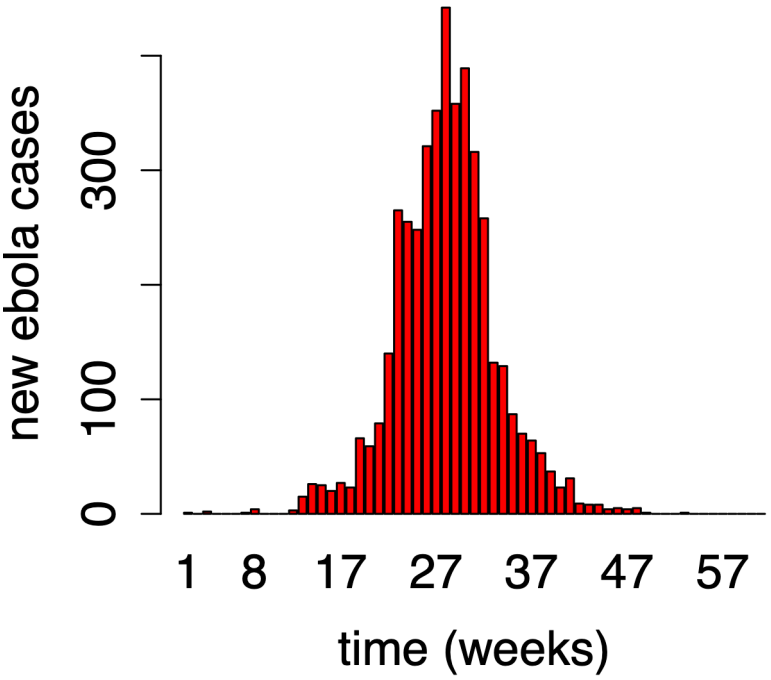


Figure 11.7: Number of new cases of ebola virus infections per week in Liberia time ranging from March 17, 2014 (week 1) until May 20, 2015 (week 61). Data from http://apps.who.int/gho/data/node.ebola-sitrep

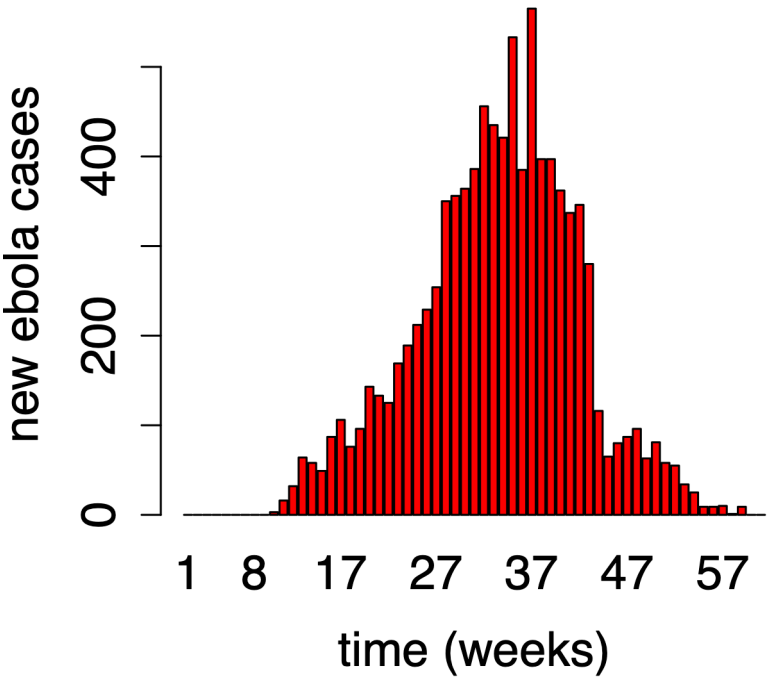


Figure 11.8: Number of new cases of ebola virus infections per week in Sierra Leone time ranging from March 17, 2014 (week 1) until May 20, 2015 (week 61). Data from http://apps.who.int/gho/data/node.ebola-sitrep

12 Linear ODEs with two variables

In chapter 3 we introduced and analyzed discrete-time models with multiple variables representing different demographic groups. In those models, the populations at the next time step depend on the population at the current time step in a linear fashion. More generally, any model with only linear dependencies can be represented in matrix form. In this chapter we will learn how to analyze the behavior of these models, and identify all possible classes of linear dynamical systems.

The main concept of this chapter are the special numbers and vectors associated with a matrix, called eigenvalues and eigenvectors. Any matrix can be thought of as an operator acting on vectors, and transforming them in certain ways. Loosely speaking, this transformation can be expressed in terms of special directions (eigenvectors) and special numbers that describe what happens along those special directions. Finding the eigenvalues and eigenvectors of a matrix allows us to understand the dynamics of biological models by classifying them into distinct categories.

In the modeling section, we will develop some intuition for modeling activators and inhibitors of biochemical reactions. We will then learn how to draw the flow of two-dimensional dynamical systems in the plane. In the analytical section, we will define eigenvectors and eigenvalues, and use this knowledge to find the general solution of linear multi-variable systems. In the computational section, numerical solutions of eigenvalues and eigenvectors will be applied to classifying all linear multi-dimensional systems, and to plotting the solutions, both over time and in the plane. Finally, in the synthesis section we will use a light-hearted model of relationship dynamics to illustrate how to analyze linear dynamical systems.

12.1 Flow in the phase plane

12.1.1 activators and inhibitors in biochemical reactions

Suppose two gene products (proteins) regulate each others' expression. Activator protein A binds to the promoter of the gene for I and activates its expression, while inhibitor protein I binds to the promoter of the gene for A and inhibits its expression (here the variables stand for concentrations of the two proteins in the cell):

$$\begin{aligned}\dot{A} &= -\alpha I \\ \dot{I} &= \beta A\end{aligned}$$

α and β are positive rate constants. They represent the rate of inhibition of A by I, and of activation of I by A, respectively. Let us now complicate the model by adding self-inhibition. It is common for regulatory proteins to inhibit their own production. Then, we have the following system of equations:

$$\begin{aligned}\dot{A} &= -\gamma A - \alpha I \\ \dot{I} &= \beta A - \delta I\end{aligned}$$

Here we have added two rates of self-inhibition γ and δ. This is a system of two coupled ODEs, and we will learn how to analyze these models both analytically and graphically.

12.1.2 phase plane portraits

Before we learn about the analytical tools of linear algebra, let us think intuitively about the effect of the variables on each other. The best way to describe this is through plotting the geometry of the *flow* prescribed by the differential equations. As we saw, for one-dimensional ODEs the direction of the change of the dependent variable (also known as the flow) could be shown as arrows on a line. A single variable can only increase, decrease, or stay the same (at a fixed point). In two dimensions there is more freedom. The flow is plotted on the *phase plane*, where for any combination of the two variables (say x, y) the ODE gives the derivatives of x and y. This vector gives the flow, or the rate of change at the particular point in the plane. Intuitively, the flow describes the direction in which the system is pulling the 2-dimensional solution. If we plot the progress of a solution of ODE (all the values of x and y starting with the initial condition) we will obtain a *trajectory* in the phase plane. The arrows of the flow are tangent to any trajectory curve, since they plot the derivatives of x, y.

Example: positive relationship between the variables Consider the following system of differential equations:

$$\begin{aligned}\dot{x} &= x + y \\ \dot{y} &= x + 2y\end{aligned}$$

(eq-ode1)

This is system is coupled, with x having an effect on y and vice versa. Specifically, the signs of the constants mean that positive values of x have the effect of increasing y (and vice versa), while negative values of x have the effect of decreasing y (and vice versa). For any pair of values of (x, y), there is a flow prescribed by the ODEs. E.g., when $x = 1, y = 1$, the derivatives

are $\dot{x} = 2, \dot{y} = 3$. This means that the flow at that point is given by the vector $(2, 3)$, and can be plotted in the x, y phase plane. This can be done for any pair of values of x and y, and plotted to give the phase plane portrait in figure {numref}`fig-ode1`.

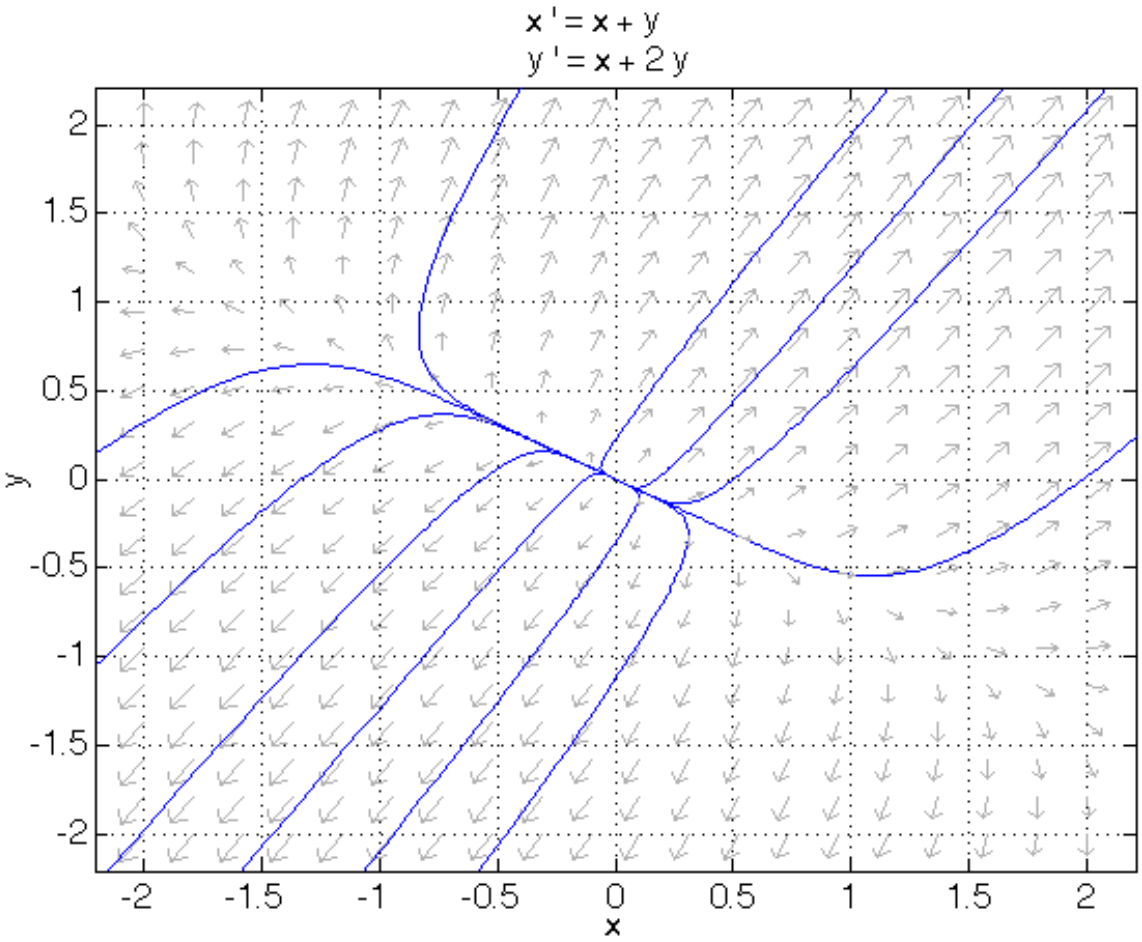


Figure 12.1: Phase plane flow for the system in {eq}`eq-ode1`

Observe that the overall dynamics of the systems are directed outward from the origin, as we expect from the ODEs. The blue lines on the plot are some sample trajectories. The solution over time for both x and y will either grow toward positive infinity, or decay to negative infinity.

Example: negative relationship between the variables Consider the following system of differential equations, where y has an effect on $\dot{x}$ opposite of its own sign. That is, negative values of y contribute to the growth of x, and vice versa.

$$\begin{aligned} \dot{x} &= -y \\ \dot{y} &= x \end{aligned}$$

(eq-ode2)

As above, the flow at any one point is given by the ODEs. E.g. at $(0, 1)$ the two derivatives prescribe flow in the $(1, 0)$ (up) direction, while at $(1, 0)$ the flow is in the $(0, -1)$ direction. Figure {numref}`fig-ode2` shows the arrows of flow in the phase plane around the origin. Note that the arrows go around in a circular pattern around the origin - this shows oscillatory flow of solutions.

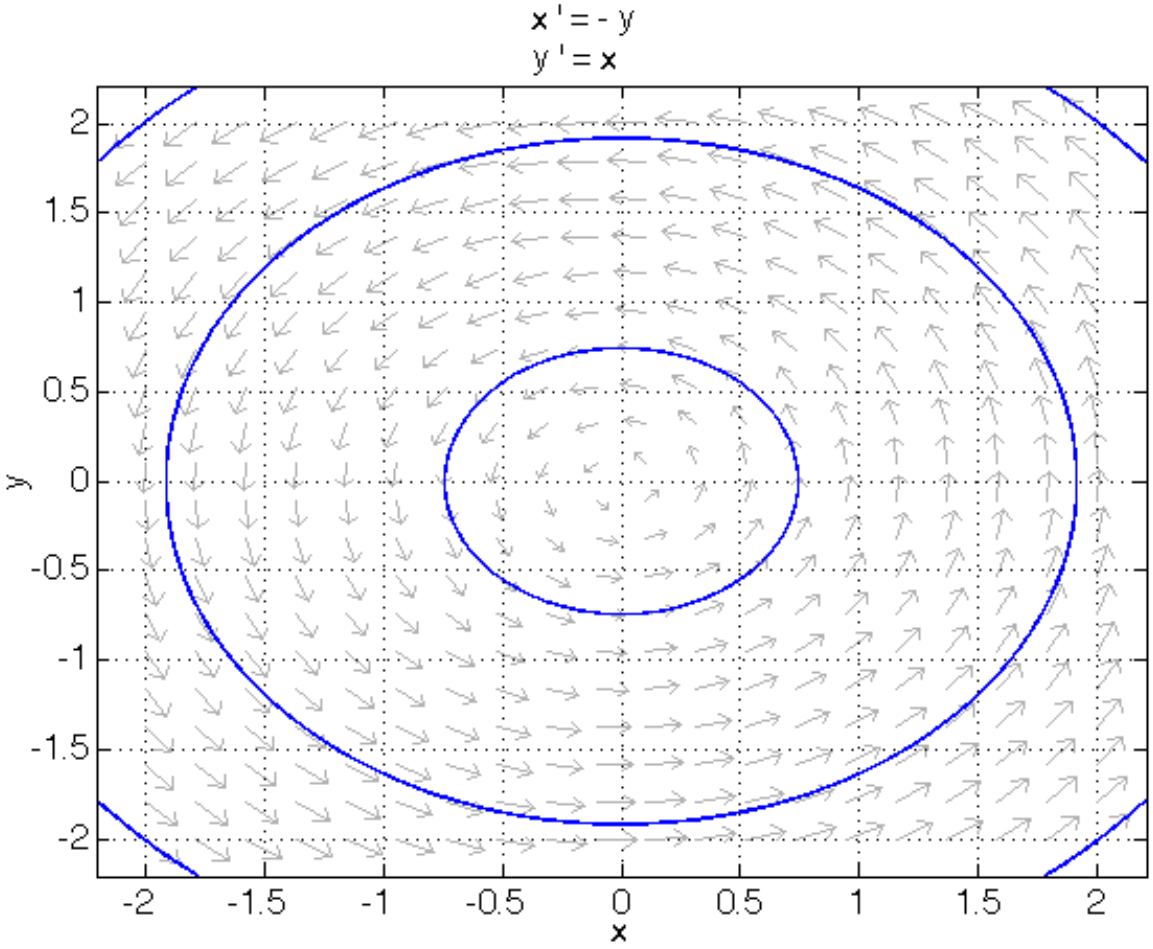


Figure 12.2: Phase plane flow for the system in {eq}`eq-ode2`

Let us consider the trajectories of x and y in time. The blue curves in the phase plane plot demonstrate the solutions go around the origin and return to the same point. This means

that the behavior of the solutions over time is *periodic*, with oscillations going from positive to negative numbers and back forever.

12.2 Solutions of linear two-variable ODEs

Let us start by considering two variable ODEs that do not affect each other:

Example: two uncoupled ODEs In general, for a two-variable system, the value of one variable affects the other. In the equations above, the terms with the constants b and c provide what is known as *coupling* between the two variables. Let us look at the primitive situation where the two variables are uncoupled, as an illustration of solving two-dimensional ODEs. If we set the coupling constants b and c to 0, we get:

$$\begin{aligned} \dot{x} &= ax \\ \dot{y} &= dy \end{aligned}$$

Using our knowledge of 1D linear ODE, we can solve the two equations independently to get the following: $x(t) = x_0 e^{at}$ and $y(t) = y_0 e^{dt}$. The solutions can be written in vector form:

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = x_0 e^{at} \begin{pmatrix} 1 \\ 0 \end{pmatrix} + y_0 e^{dt} \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

This is another way of writing that the dynamics of variable x is exponential growth (or decay) with rate a, and ditto y, with rate d. Given the initial conditions (x_0, y_0), we can divide the behavior of the solutions into a sum of two vectors, each growing or decaying at its own rate.

Linear algebra allows us to find the solution for two-dimensional ODEs where the variables are interdependent using the same idea. The general (homogeneous) ODE with two dependent variables can be written as follows:

$$\begin{aligned} \dot{x} &= ax + by \\ \dot{y} &= cx + dy \end{aligned}$$

We can write this in matrix form like this:

$$\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Let us call the matrix A, and represent the vector (x, y) as $\vec{x}$, then the general linear equation can be written like this:

$$\dot{\vec{x}} = A\vec{x}$$

(gen-lin-mult)

This notation is intended to make plain the similarity with the linear 1D ODE: $\dot{x} = ax$. This similarity is deep and substantial, in that linear equations in multiple dimensions share the same basic exponential form. In general, all solutions of linear equations can be written as a sum of exponentials multiplying different vectors:

General solution of linear 2-variable ODEs

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = C_1 e^{\lambda_1 t} \begin{pmatrix} x_1 \\ y_1 \end{pmatrix} + C_2 e^{\lambda_2 t} \begin{pmatrix} x_2 \\ y_2 \end{pmatrix}$$

(gen-sol-2var)
The constants C_1, C_2 are determined by the initial conditions, while the constants λ_1, λ_2 are the eigenvalues and the vectors (x_1, y_1) and (x_2, y_2) are the eigenvectors of the matrix A. We will now consider the application of this general result using computational tools.

12.3 Classification of linear systems

We have seen that linear algebra allows us to write down the solution of a multivariable dynamical system into a sum of exponential terms. In this section we use computational techniques to find the eigenvalues and eigenvectors of a system, and then produce the *phase portraits* of the linear systems. There are only a few different types of flow possible for linear systems, and we will classify them.

12.3.1 real eigenvalues

Let us consider the fixed points of the linear system: since both $\dot{x} = 0$ and $\dot{y} = 0$ must be zero, the only fixed point is the origin $(0, 0)$. We will see that the stability of the fixed point depends on the sign of the real part of the eigenvalue.

Suppose we have a positive real eigenvalue. The solution in the direction of the corresponding eigenvector is then described by $Ce^{\lambda t}$, $\lambda > 0$, which is exponential growth. The means that the solution is going to grow in the direction of the eigenvector away from the origin, and thus the origin is an unstable fixed point (in this direction). This type of fixed point is called an *unstable node*.

On the other hand, if $\lambda < 0$ for both eigenvalues, the solution decays exponentially and thus approaches the origin, so the fixed point is stable. This type of fixed point is called a *stable node*.

Since there are two different eigenvalues, one may be positive while another is negative. In this case, the fixed point is is called a *saddle point* for geometric reasons: solutions flow toward it in one direction, like a marble along the forward-backward axis of a saddle on a horse and flow away from it along the sideways direction on a saddle. Then, the fixed point is stable when approached along one eigenvector, but unstable along the other. What happens if the initial condition is not on either eigenvector? I will use a fact of linear algebra that given any two (non-colinear) 2D vectors, any vector in the plane can represented as a sum (with some coefficients) of these two. Thus, the general solution can be written as follows:

$$\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = C_1 e^{at} \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} + C_2 e^{-bt} \begin{pmatrix} u_1 \\ u_2 \end{pmatrix}$$

where $a, b > 0$. Then we see that the component in the direction of the first eigenvector will grow, while the component along the second eigenvector will decay. Thus, as $t \to \infty$, all solutions will approach the vector with the unstable eigenvalue, except those with initial conditions right on the eigenvector corresponding to the stable eigenvalue. This means that the fixed point is essentially unstable, because only trajectories which start exactly along the stable direction approach the fixed point in the long run, while others, may approach the fixed point for a finite time, flow away when the unstable component with the positive eigenvalue takes over, as shown in figure .

[Phase plane flow for a linear system with a saddle point] (images/week6_pp1.png)

12.3.2 complex eigenvalues

If the argument of the square root is negative, eigenvalues may be complex numbers, which we can write like this: $a + bi$. Using Euler's formula, we can write down the time-dependent part of the solutions as the following:

$$e^{(a+bi)t} = e^{at} e^{bit} = e^{at}(\cos(bt) + i \sin(bt))$$

The behavior of these solutions combines exponential growth or decay from the real part, with the oscillations produced by the imaginary part. This describes either exponentially growing or decaying oscillations, which look like decaying waves in time, or as a spiral in the phase plane:

Thus we see that the stability of the fixed point with complex eigenvalues depends on the sign of the real part. Purely imaginary eigenvalues produce periodic oscillations, which keep the same amplitude, as we saw in the example in the modeling section.

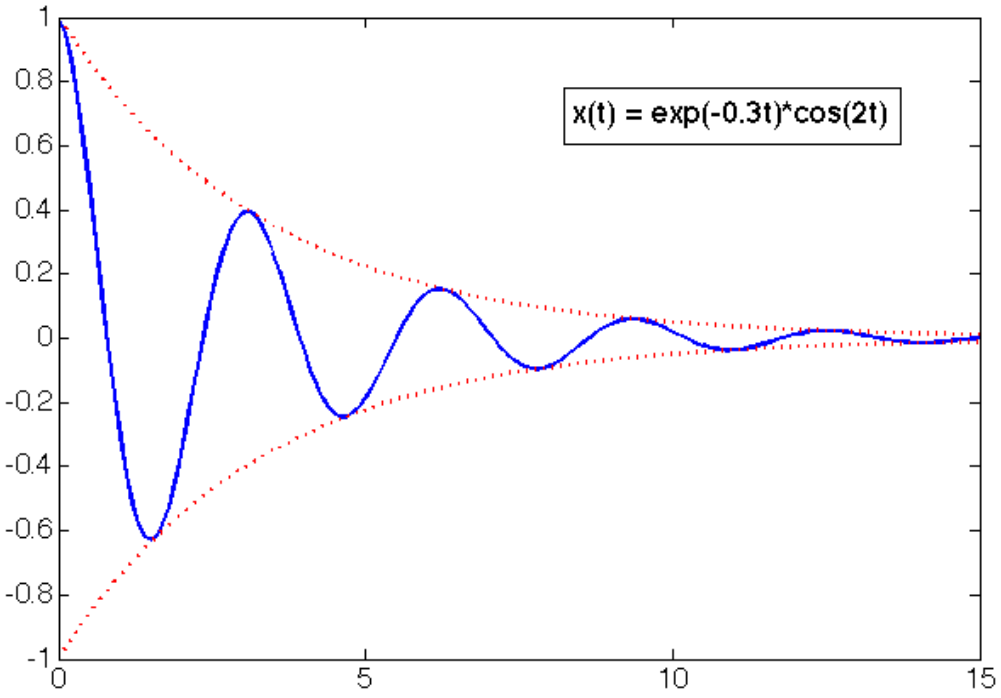


Figure 12.3: Exponentially decaying oscillations in the time plot of the solution $x(t)$

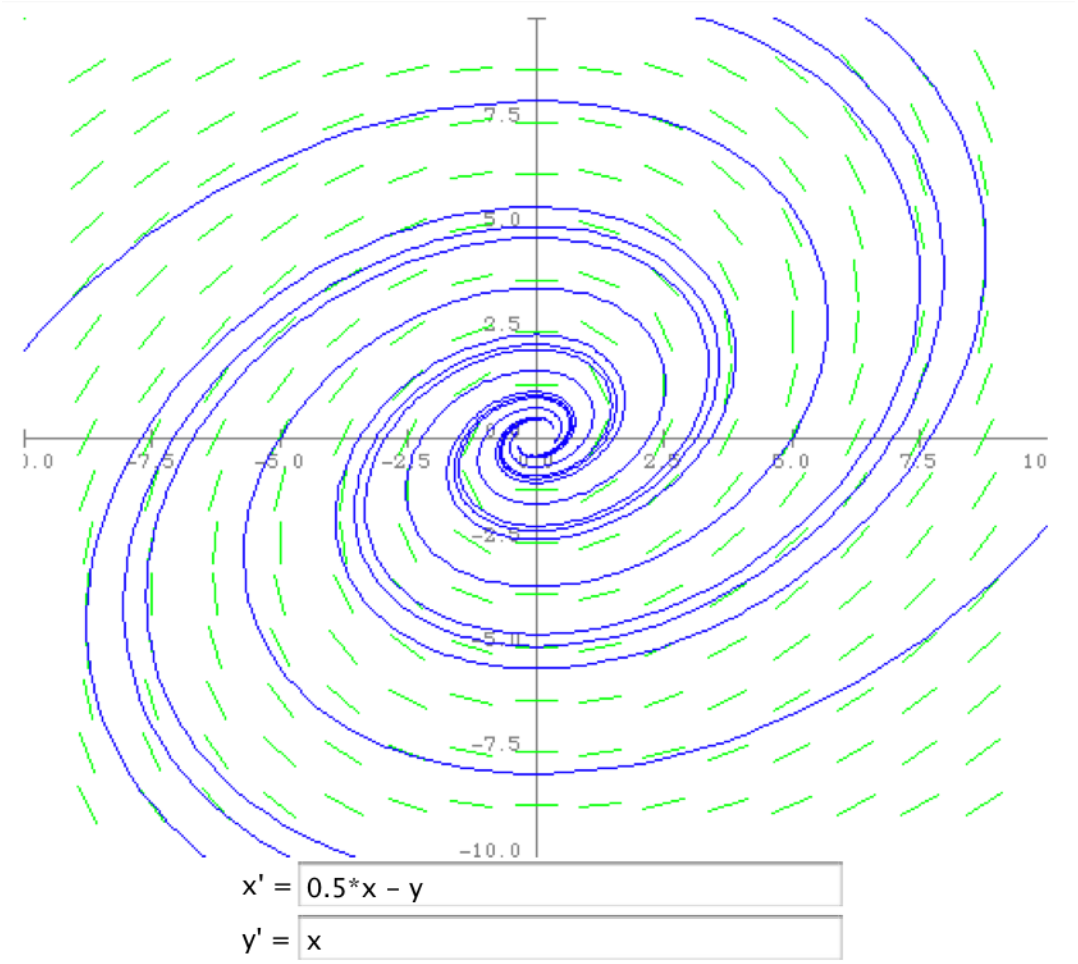


Figure 12.4: Phase plane portrait of a stable spiral

12.3.3 classification of linear systems

Stability	positive real parts	negative real parts	one positive, one negative	zero real part
real:	unstable node	stable node	saddle point	fixed line
complex:	unstable spiral	stable spiral	N/A	center point

Eigenvalues of linear ODEs define type of phase plane

The table above summarizes all the different types of flows in the phase plane possible for linear systems, in terms of the behavior of solutions relative to the fixed point at the origin. If the eigenvalues are real, the solutions will be exponential in nature. There are three possibilities for nonzero eigenvalues: *stable node* (both eigenvalues are negative), *unstable node* (both eigenvalues are positive), and a *saddle point* (mixed signs). If one of the eigenvalues is zero, this means that there is not flow along one direction, so there is a *line of fixed points* in the direction of the corresponding eigenvector (if both eigenvalues are zero, there is no flow at all.)

For complex eigenvalues, there are three possibilities: if the real part is positive, the solution will grow and oscillate (oscillations with exponentially increasing amplitude), if the real part is negative, the solution will decay and oscillation (oscillations with exponentially increasing amplitude), and if the real part is zero (pure imaginary eigenvalues) the solution will oscillate with constant amplitude. The first type is called an *unstable spiral*, the second a *stable spiral*, and the third a *center*. It is not possible for complex eigenvalues of two-dimensional systems to have different signs of real parts, because as the formula shows, the real part is the same for both and is equal to the trace divided by two.

12.4 Dynamics of romantic relationships

We examine a model, taken from (**strogatz_nonlinear_2001?**), that applies dynamical systems modeling to a pressing concern for many humans: the prediction of dynamics of a romantic relationship. There are several unrealistic assumptions involved in the following model: first, that love or affection can be quantified, second, that any changes in relationship depend only on the emotions of the two people involved, and third, that the rate of change of the two love variables depend linearly on each other.

If we can give those assumptions the benefit of the doubt (which is how all relationships begin), we can write down a system of ODEs to describe a romantically involved couple. Here X and Y are dynamic variables that quantify the emotional states of the two lovers:

$$\begin{aligned}\dot{X} &= aX + bY \\ \dot{Y} &= cX + dY\end{aligned}$$

Let us denote positive feelings (love) with positive values of X, Y, while negative values signify negative feelings (hate.) The significance of the parameters can be interpreted as follows: a, d describe the response of the two people to their own feelings, while b, c correspond to the effect the other person's feeling has on their own. For example, a person whose feeling grow as the other person's affection increases can be modeled with a positive value of b (or c). On the other hand, a person whose own feelings are dampened by the other one's excessively positive emotions, can be decribed by a negative value of b or c. Their own feelings can also play a role, either positive or negative, reflected in the sign of the constants a and d.

Using mathematical modeling, we can answer the following basic questions:

1. Given a set of values for parameters a, b, c, d, predict the dynamic behavior of the model relationship.
2. Find conditions for stability and existence of oscillations in the dynamical system, expressed as a function of the parameters.

To address the first question, here are some simplified scenarios for our two lovers in the model.

Detached lovers: Let the emotional state of the two lovers depend only on their own emotions, for example:

$$\begin{aligned}\dot{X} &= X \\ \dot{Y} &= -Y\end{aligned}$$

To classify the behavior of the model, we find the eigenvalues of the system. In this case, they are the diagonal elements of the matrix, 1 and -1. This mean that the origin is a saddle point, and therefore it is unstable. In the X direction, the emotions are going to grow without bound, either in the love or hate direction, while in the Y direction, the emotions are going to decay to zero (indifference). This should be no surprise, that since the two equations are independent, the lovers have no emotional effect on each other.

Lovers with no self-awareness: Here is an alternate situation: suppose two lovers were not influenced by their own emotions, but were instead attuned to the emotional state of the other. Then we might have the following model, in which lover X reacts in the opposite way by emotions of lover Y, but lover Y is, contrariwise, spurred by the love or hate of X in the same direction:

$$\begin{aligned}\dot{X} &= -Y \\ \dot{Y} &= X\end{aligned}$$

We find the eigenvalues of the system by using the expression in equation [eg:2D_eig]: $\lambda = (0 \pm \sqrt{-4})/2 = \pm i$. Pure imaginary eigenvalues tell us that the origin is a center point, with the solutions periodic orbits around the origin. Psychologically, we can interpret this scenario as cycles of love and hate, never growing and never decaying. The magnitude of these oscillations depends on the initial state of the system, that is, the feelings the lovers had at the beginning of the relationship.

We can now address the second question, and find under what circumstances different types of dynamic behaviors occur. We consider the general model, and ask what kinds of eigenvalues are possible for different parameter values. First, we write down the general expression for the eigenvalues, from equation [eg:2D_eig]:

$$\lambda = \frac{a + d \pm \sqrt{(a+d)^2 - 4(ad - bc)}}{2}$$

There are two properties we are interested in: stability and existence of oscillations. Recall that stability is determined by the sign of the real part of the eigenvalues. If the square root is imaginary, then the real part is simply the trace $(a + d)$, but if the square root is real, we have to consider the whole expression to determine stability. So let us first state the condition for existence of oscillations (imaginary square root):

1. Complex eigenvalues: oscillatory solutions $4(ad-bc) > (a+d)^2$. If this expression holds, the square root is imaginary, and the stability is determined by the sign of the trace. That is, if $a + d > 0$, the system is unstable, and will grow into unbounded love or hate, but if $a + d < 0$, then the system is stable, and will spiral to indifference. The special case $a + d = 0$, such as we saw above, means that strictly periodic love/hate cycles are the solutions.

2. Real eigenvalues: exponential growth and/or decay $4(ad - bc) < (a + d)^2$. In this case, the square root is real, and no oscillatory solutions exist. In order to determine whether this implies exponential growth, decay, or a combination, we must weigh the relative sizes of $(a + d)$ and $\sqrt{(a+d)^2 - 4(ad - bc)}$. If $|a + d| > \sqrt{(a+d)^2 - 4(ad - bc)}$, then adding or subtracting the square root does not change the sign of $(a+d)$: if it is negative, both eigenvalues are negative, and the origin is a stable node, and if the trace is positive, the origin is an unstable node. However, if the absolute value of the root outweighs the absolute value of the trace $|a + d| < \sqrt{(a+d)^2 - 4(ad - bc)}$, then either adding or subtracting the root will change the sign of the eigenvalues. Therefore, one eigenvalue is positive and the other is negative, and the origin is a saddle point. The emotions will run unchecked in some preferred direction, possibly combining love and hate of the two lovers.

These conditions are not intuitive, and it took some work to express them. The benefit is that now, given any values of the self-involvement parameters a, d and the sensitivity parameters b, c we can predict the long-term dynamics of the model relationship. Whether the results have any bearing on reality, of course, depends on how well the reality is described by these primitive assumptions.

13 Phase portraits in Python

```
#Necessary imports
import numpy as np #package for work with arrays and matrices
import matplotlib.pyplot as plt #package with plotting capabilities
from scipy.integrate import odeint
```

13.0.1 phase plane plots via quiver

We are going to plot phase diagrams for linear ODEs that have the form

$$\begin{aligned} dx/dt = &\quad ax + by \\ dy/dt = &\quad cx + dy \end{aligned}$$

Python's `ax.quiver()` function allows displays vectors with arrows made of the components (u, v), which is exactly what we need. The function takes 4 inputs (x, y, u, v): x and y are the grid points and u and v are the u and v components of the vector, which are given by our ODEs.

In order to make the grid points (x, y), we will use the function `np.meshgrid()`. It's a pretty handy function that takes as input a range of x and y values and returns two matrices x, y that together give us the grid points. Here is the code to produce a grid with an x and y range from (-1.5, 1.5) with a spacing of 0.2, we could do the following:

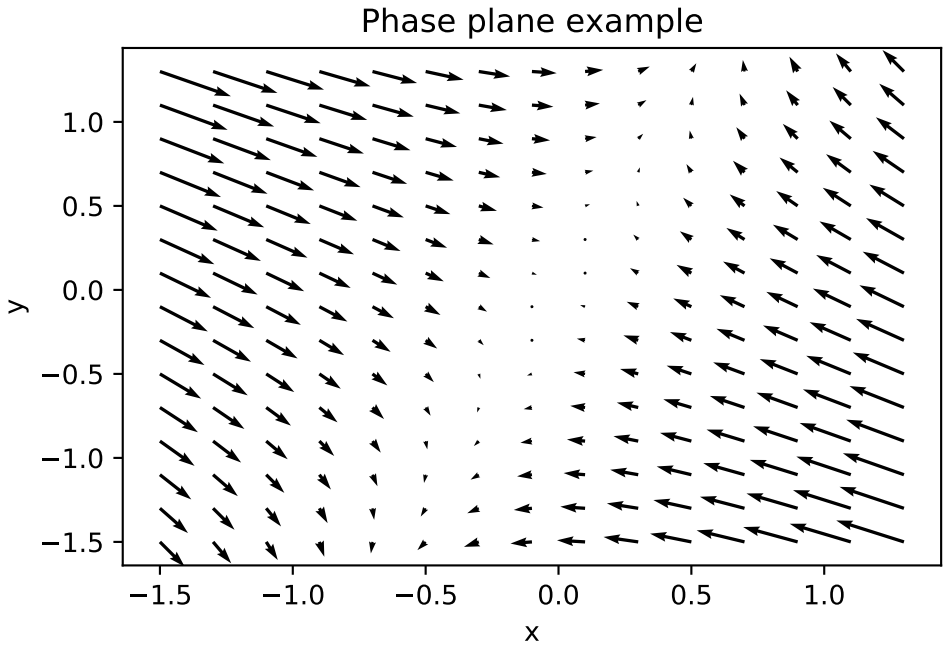
```
xmin = -1.5 #change the parameters here to control the range of the axes
xmax = 1.5
ymin = -1.5
ymax = 1.5
dx = 0.2 #set the size of the x-step on the grid
dy = 0.2 #set the size of the y-step on the grid
X = np.arange(xmin, xmax, dx)
Y = np.arange(ymin, ymax, dy)
x, y = np.meshgrid(X,Y) #create a grid
```

Define the arrays dx and dy based on the ODE in order to compute the flow vectors on that grid. Here is a linear example:

```
a = -2
b = 1
c = 1
d = 0
dx = a*x+b*y #overwrites the other dx
dy = c*x+d*y #overwrites the other dy
```

Then plot the arrows given by arrays dx,dy at points x,y:

```
fig, ax = plt.subplots()
q = ax.quiver(x, y, dx, dy)
plt.xlabel('x')
plt.ylabel('y')
plt.title('Phase plane example')
plt.show()
```



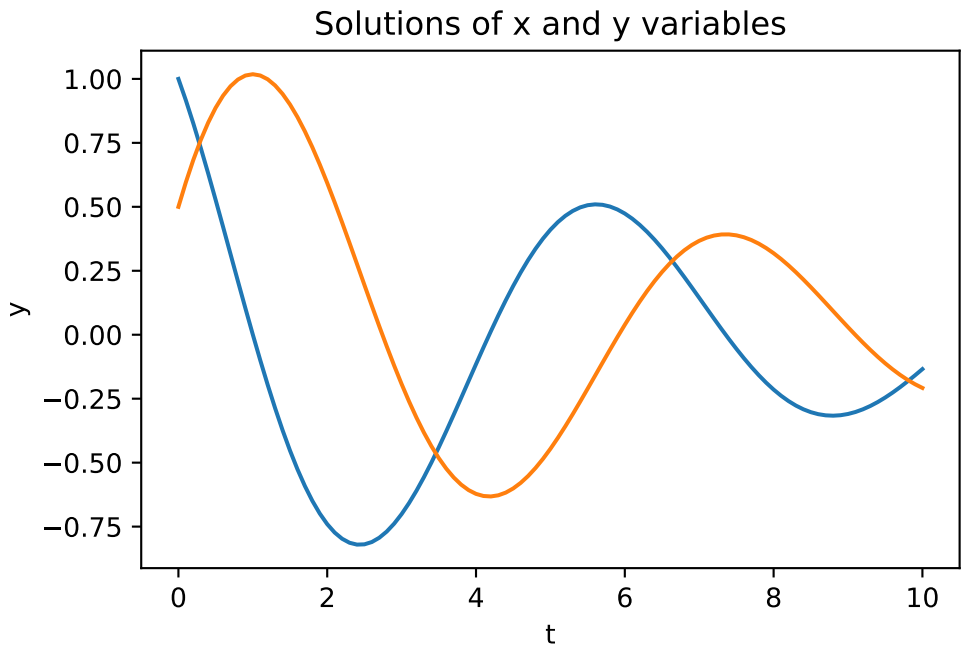
13.0.2 ODE solutions using odeint

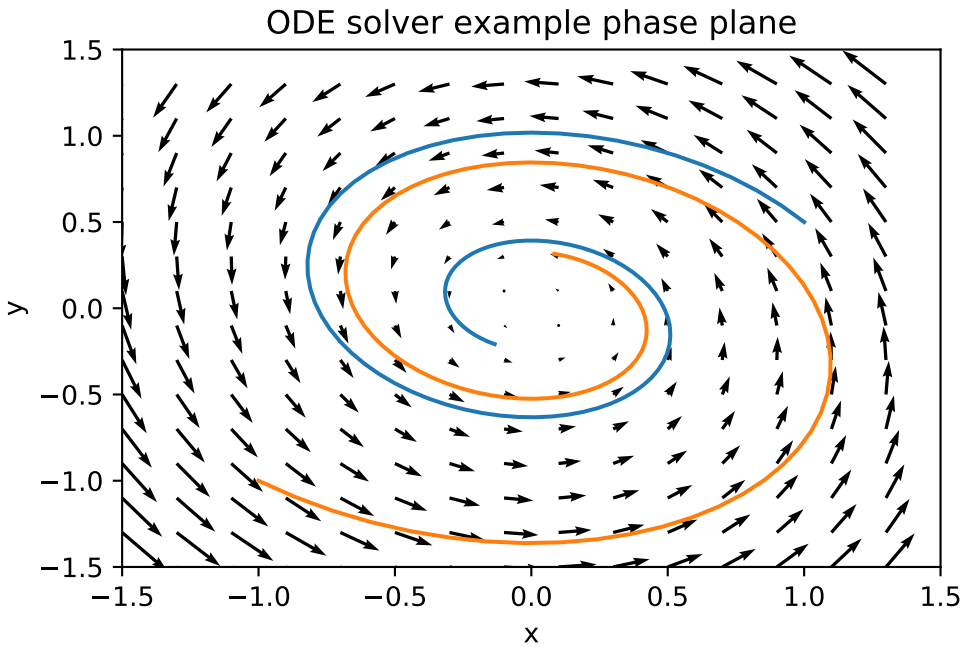
Python has an entire suite of ode solvers. We'll use the function `odeint`, with documentation provided here: https://docs.scipy.org/doc/scipy/reference/generated/scipy.integrate.odeint.html

This requires defining a function `fun` that sets the two functions for the two-variable ODE, to be called by the `odeint`, together with parameter values a, b, c, d. There are many other options that you can read about in the documentation page. Here is the sample code:

```
xmin = -1.5 #change the parameters here to control the range of the axes
xmax = 1.5
ymin = -1.5
ymax = 1.5
dx = 0.2 #set the size of the x-step on the grid
dy = 0.2 #set the size of the y-step on the grid
X = np.arange(xmin, xmax, dx)
Y = np.arange(ymin, ymax, dy)
x, y = np.meshgrid(X, Y);  #create a grid

a = -0.3
b = -1
c = 1
d = 0

dx = a*x+b*y
dy = c*x+d*y

# define the function for the ODES: note the order of inputs
def fun(xy, t, a, b, c, d):  # inputs are: variable array, time, any parameters
    newxy = [a*xy[0]+b*xy[1], c*xy[0]+d*xy[1]]
    return newxy

# Set the initial values, the vector of times, and call the ODE solver
init = [1, 0.5] #[intial x, initial y]
t = np.linspace(0, 10, 101) # create time vector
sol = odeint(fun, init, t, args=(a, b, c, d)) # calculate numeric solution of ODE defined

# Plot the solutions over time
plt.plot(t, sol)
plt.xlabel('t')
plt.ylabel('y')
```

```
plt.title('Solutions of x and y variables')
plt.show()


# plot the arrows given by arrays dx,dy at points x,y:
fig, ax = plt.subplots()
q = ax.quiver(x, y, dx, dy)
plt.xlim(-1.5,1.5)
plt.ylim(-1.5,1.5)
ax.plot(sol[:,0], sol[:,1]) # plot the x and the y variable in the phase plane

# Set different initial values, the vector of times, and call the ODE solver again
init = [-1, -1] #[intial x, initial y]
t = np.linspace(0, 10, 101) # create time vector
sol = odeint(fun, init, t, args=(a, b, c, d)) # calculate numeric solution of ODE defined

ax.plot(sol[:,0], sol[:,1]) # plot the x and the y variable in the phase plane
plt.xlabel('x') #use more informative labels for a real model
plt.ylabel('y')
plt.title('ODE solver example phase plane')
plt.show()
```




ODE solver example phase plane
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
y
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
x

14 Forces and potentials in biological modeling

In the last chapter we learned to analyze the dynamics of linear dynamical systems by finding the eigenvalues of corresponding matrices. In this chapter, we will study a class of mathematical models with a classical physics pedigree. These models are based on a physical potential, which is usually given by a physical law, e.g. gravity. These systems are special, because they have a conserved quantity that is known as energy. We will learn about the consequences of such conserved quantities, and what happens when the conservation is broken and energy dissipates.

The main type of models that we will address are oscillators, with the simplest being simple springs. These models are in fact relevant for biological modeling in a variety of fields. We will apply the analysis to the study of biomolecular flexibility, to predict the preferred directions of internal motion of parts of protein molecules. The study of protein structural dynamics has important implication to understanding the mechanism of function of many biochemical systems. For example, many signaling molecules undergo conformational changes upon binding or phosphorylation, which generates changes in their biochemical actions.

The modeling section explains the basic physics of forces and potential functions. We will use examples of a single mass on a spring, and the scale up to two masses connected by a spring. In the analytic section, we will learn how to turn second order ODEs into first order dynamical systems, then find the explicit solutions for the oscillations of a simple spring. We will show how damping enters the solution, and what effect it has on conservation. We will also learn how to include external forcing in the model. In the computational section we will describe a general system of coupled linear oscillators using normal mode analysis. In the final section normal mode analysis is applied to studying the dynamics of protein structures.

14.1 Forces and simple springs

As we have learned in the last two chapters, linear dynamical systems have general solutions in the form of a weighted sum of exponentials:

$$\vec{x}(t) = \sum_i c_i \vec{v}_i e^{\lambda_i t}$$

The type of dynamics possible in a system is determined by the eigenvalues λ_i of the corresponding matrix. If they are real, the system will grow or decay exponentially, while the

presence of imaginary part produces oscillations, with growing, decaying, or constant amplitude, depending on the real part. Below we describe the modeling situations in which this limited menu of dynamics may be applied.

14.1.1 exponential growth and decay

No biological systems have purely exponential dynamics, as nothing in nature can realistically grow without bound, or decay inexorably to zero. Nevertheless, exponential behavior can be useful for modeling the dynamics in a limited regime, especially near an equilibrium. We have seen an illustration of this idea in the first part of the course, when we analyzed nonlinear one dimensional dynamical systems by approximating them with a linear system near an equilibrium. This linearization process is also applicable to multidimensional systems, where it involves the use of a first derivative matrix (called the *Jacobian*) as the local linear approximation. We will learn how to do this in detail in a future chapter, but the idea remains the same: by analyzing the eigenvalues of the matrix, one can predict whether the solution of the system grows or decays near the equilibrium. In the first case, the equilibrium is unstable, in the second, it is stable.

14.1.2 properties of linear oscillations

Another area of applicability of linear models is for describing oscillatory behavior. Many biological systems exhibit oscillatory dynamics, ranging from heartbeat and circadian rhythms in physiology, to cycles of biochemical reactions and firing of neurons. We know that linear systems with complex eigenvalues have oscillatory dynamics, so it is tempting to describe these phenomena with linear models. As we saw above, however, linear models cannot describe real biological systems over all possible values of its variable. Linear oscillations have specific properties which are generally not found in real systems: if the real part of the eigenvalue is nonzero, they either have exponentially growing or decaying amplitudes, which, as we argued above is not biologically feasible in the long run. This leaves the situation when the real part is zero, and the oscillations have a constant amplitude. This situation is not immediately unrealistic, but it has its own specific limitations: in it, oscillations of all amplitudes are possible, regardless of the frequency (think of the simple harmonic oscillator in the previous chapter.) This is also unrealistic, and nonlinear models are necessary to model the dynamics usually observed in reality, in which oscillations occurs with a preferred frequency and amplitude.

14.1.3 potentials and forces

The dynamics of systems in classical physical are defined by their potential functions. The *potential energy* describes the propensity of the system to do work, that is, to apply *force* to an object over some distance. By definition, work is the difference between the potentials at

the two endpoints of the path, a and b. This can be summarized in the following equation, relating *work* W done on an object by applying force $f(x)$ over the distance from a to b:

$$W = V(b) - V(a) = \int_a^b f(x)dx$$

Intuitively, one can think of a potential function $V(x)$ as a law handed from on high, which dictates what forces are going to act on the system. The forces then make the objects in the systems move, generating dynamics that we care about and will investigate in this chapter.

The above connection between potential and force has a familiar form. In fact, it is equivalent to the Fundamental Theorem of calculus, where we define the force $f(x)$ to be the derivative of the potential $V(x)$. This relationship is at the center of this chapter:

$$f(x) = -V'(x)$$

Graphically speaking, the force is the negative slope of the potential function. This means that if the force is always pushing the object down the slope: if the potential is rising, it is pushing backward, and if the potential is falling, it is pushing forward. Essentially, this definition is consistent with the metaphor of potential as a terrain, with gravitational force bringing objects down to the lowest point of the landscape.

14.1.4 harmonic spring potential

We will define a specific potential for a mass on a spring and investigate the resulting dynamical system in the analytical section. The assumptions are: a) that there is a position x_0 at which the spring is at rest, that is, no force is acting on the object, and b) that the force will push the object back toward the resting position, with strength proportional to the displacement $(x - x_0)$ of the mass from the resting position. Turns out that this model is defined by the following potential function:

$$V(x) = \frac{1}{2}k(x - x_0)^2$$

Here k is known as the *spring constant*, and this parameter describes the strength of the restoring force: for large k, the mass will be pulled toward the resting state with greater force than with a smaller k. Notice that the potential has a minimum at x_0, since this is the position that is most favorable for the system. The variables and parameters of the model are illustrated in figure .

Using the relationship in equation [eq:force_pot] above, we conclude that the *restoring force* in the system obeys the following equation:

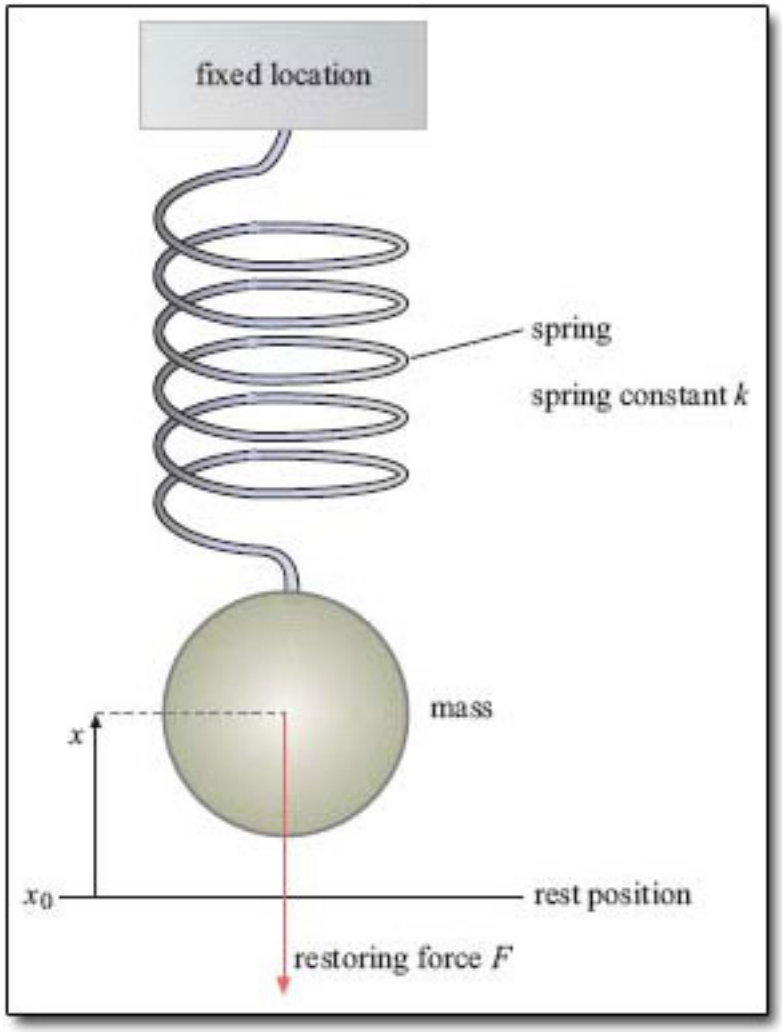


Figure 14.1: Mass on a spring, with a restoring force F and rest position x_0. http://openlearn.open.ac.uk/

$$f(x) = k(x_0 - x)$$

This model of a simple spring with a quadratic potential and linear force is called a *Hookean* spring model, after the physicist Robert Hooke. The idealized linear spring model is also called a *harmonic oscillator*, because, as we will see in the analytical section, its solutions are perfect periodic oscillations.

The expression for force can be translated into the language of ODEs and dynamics using Newton's second law, which states that $f = ma$. Recall that the acceleration in physics is the second derivative of position, and we can write down the dynamical system of a harmonic oscillator as follows:

$$m\ddot{x} = k(x_0 - x)$$

We will learn to solve this equation in the next section.

In reality, there is usually another force that acts to slow down any moving object. This effect is called *kinetic friction*, or *viscosity* if the object is in a fluid, such as air or water. The typical model for kinetic friction assumes the friction force is proportional to the velocity of the object. The relationship has a negative sign, since the force acts in the opposite direction of the velocity, slowing down the motion. The friction force $g(x)$ is defined as follows.

$$g(x) = -\gamma v = -\gamma \dot{x}$$

Thus, a system incorporating a harmonic oscillator with friction has the following equation of motion:

$$m\ddot{x} = k(x_0 - x) - \gamma \dot{x}$$

14.1.5 two masses connected by a spring

We have only looked at a single oscillator, whose dynamics depend solely on its own position. Now, consider two separate objects connected by a linear Hookean spring. Let us define x and y to be the displacements of the two objects from the respective resting positions; this way we don't have to keep around x_0 and y_0. The force depends on the distance between the two of them, and if we restrict the model to one dimension, it depends on the difference between the two displacements. The force on each particle is in the opposite direction of its own displacement and in the same direction as the other particle's displacement, so the equations are:

$$\begin{aligned} m\ddot{x} &= -k(x-y) \\ m\ddot{y} &= -k(y-x) \end{aligned}$$

The dynamical system can be expressed in matrix form, where the matrix A is:

$$A = \frac{k}{m}\begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix}$$

Then the system of ODEs can be written down more concisely, with the vector $\vec{x}$ contains the positions of both x and y:

$$\ddot{\vec{x}} = A\vec{x}$$

14.1.6 converting second order ODEs into first order

In the previous section we found the can transform it into a 2D system and use the geometric methods of phase plane analysis analysis. Let us introduce a second variable $y = \dot{x}$. Then $\dot{y} = \ddot{x}$, and we can write a system of equations in the standard first order form, which will enable us to use our analytical tools.

Example. Let us consider the harmonic oscillator with no damping, as defined in the modeling section. The second order equation can be written as two first order equations:

$$\begin{aligned} \dot{y} &= -\frac{k}{m}x \\ \dot{x} &= y \end{aligned}$$

The eigenvalues of this system are $\lambda = \pm\sqrt{\frac{k}{m}}i$, and $\sqrt{\frac{k}{m}}$ is the frequency of oscillation of this model, which in physics is used to model a simple Hookean spring. The solutions are in the form of sines and cosines, and we can write it:

$$x(t) = A\cos(\sqrt{\frac{k}{m}}t) + B\sin(\sqrt{\frac{k}{m}}t)$$

The constants A and B are determined by the initial conditions. Note that for second order equations, two initial conditions must be specified (e.g. one for x and one for $\dot{x}$) to determine the two integration constants.

Example. Let us now consider the harmonic oscillator with damping added. The second order equation can be written as two first order equations:

$$\begin{aligned} \dot{y} = & -\frac{k}{m}x - \frac{\gamma}{m}y \\ \dot{x} = & \qquad\qquad y \end{aligned}$$

The eigenvalues of this system are: $\lambda = (-\gamma \pm \sqrt{\gamma^2 - 4km})/2m$. This means, depending on the sign of $\gamma^2 - 4km$, the eigenvalues may be either real or complex. If they are complex, the solutions will be damped oscillations because the real part $(-\gamma)$ is negative. The solution can be written with the following exponential and oscillation terms:

$$x(t) = Ae^{-\frac{\gamma}{m}t}\cos[(\sqrt{k/m - \gamma^2/4m^2})t] + Be^{-\frac{\gamma}{m}t}\sin[(\sqrt{k/m - \gamma^2/4m^2})t]$$

14.1.7 dynamic behaviors of the harmonic oscillator

We have seen that when there is a restoring force $F = -kx$, were x is the displacement from a resting value, the differential equation from Newton's second law is $m\ddot{x} = -kx$, or $\ddot{x} = -\omega^2 x$, with $\omega = \sqrt{k/m}$. We can see immediately from this expression that sines and cosines are solutions, because by taking two derivatives one gets back the same function, multiplied by the negative square of the frequency (check this yourself). This is also what we get from eigenvalue analysis, by finding the eigenvalues of the system $\pm\omega i$.

If there is a nonzero first derivative term in the force, $F = -kx + \gamma\dot{x}$; if γ is negative it is called a damping term. Let us analyze this scenario by finding the eigenvalues of the corresponding second-order ODE $\ddot{x} + \gamma/m\dot{x} + \omega^2 x = 0$:

$$\lambda = (-\gamma/m \pm \sqrt{(\gamma/m)^2 - 4\omega^2})/2$$

We see that the eigenvalues are real if $(\gamma/m)^2 > 4\omega^2$, and complex if $(\gamma/m)^2 < 4\omega^2$, and the solutions would be decaying exponentials. Physically speaking, if damping is too strong, the system's propensity for oscillations is overpowered, and the displacement x never gets to the other side of 0 (think of a mass on a spring in molasses). This system is called *overdamped*. On the other hand, if γ is below the threshold value, oscillations persist, although they tend to zero with time, in the form of $e^{-\gamma/m}(A\sin(\omega t) + B\cos(\omega t))$. This system is called *underdamped*. At $(\gamma/m)^2 = 4\omega^2$, the situation is called *critical damping* - the oscillator will return to its resting state and just miss overshooting it.

If γ were positive (I cannot think of a physical or biological example) the situation would be reversed, leading to exponentially growing oscillations or pure exponential growth.

14.1.8 forcing and inhomogeneous ODEs

We will now consider a system that consists of a harmonic oscillator to which which an external force is applied. Mathematically, the force is external if it does not depend on the variables of the system, although it may depend on time.

Now let us consider a harmonic oscillator that is driven by a periodic external force. This could represent a mass on a spring which is being "forced" by an external influence, or an oscillating neuron that receives a periodic signal from another neuron.

We will now consider an inhomogeneous ODE with the *method of undetermined coefficients*. The rule of thumb is: if the form of the inhomogeneity is unchanged by differentiation (e.g. exponentials, polynomials) then use this form multiplied by some constants as a guess for the particular solution. Then substitute it into the ODE, and find which values of the constants will satisfy the ODE.

$$\ddot{x} + \omega_0^2 x = C\cos(\omega t)$$

First, the solution of the homogeneous equation is a sum of sines and cosines with frequency ω_0: $x_h = A\cos(\omega_0 t) + B\sin(\omega_0 t)$. To find the particular solution, let us assume that the solution has the same form as the forcing term, with some undetermined coefficients: $x_h = C_1\cos(\omega t) + C_2\sin(\omega t)$.

The second derivative is then: $\ddot{x}_h = -C_1\omega^2\cos(\omega t) - C_2\omega^2\sin(\omega t)$. Plugging this into the equation we have:

$$-C_1\omega^2\cos(\omega t) - C_2\omega^2\sin(\omega t) + \omega_0^2(C_1\cos(\omega t) + C_2\sin(\omega t)) = C\cos(\omega t)$$

Since there is no sine term on the right hand side, we need to set $C_2 = 0$. The cosine terms yield the following expression: $(-C_1\omega^2 + \omega_0^2 C_1)\cos(\omega t) = C\cos(\omega t) \Rightarrow C_1 = C/(\omega_0^2 - \omega^2)$.

Adding the homogeneous and the particular solutions together, we find the general solution for a harmonic oscillator with a periodic driving force:

$$x(t) = x_h + x_p = A\cos(\omega_0 t) + B\sin(\omega_0 t) + \frac{C}{\omega_0^2 - \omega^2}\cos(\omega t)$$

The solution is a superposition of oscillations at the inherent frequency of the oscillator (ω_0) and the external driving frequency (ω). What happens when the two frequencies match?

14.1.9 forced oscillations and resonance

When $\omega = \omega_0$ he particular solution found above no longer exists because of division by zero. Thus, we need to seek another solution. Let us try the guess of $x_h = C_1 t\cos(\omega t) + C_2 t\sin(\omega t)$. Let us find its derivative, using the product rule: $\dot{x}_h = C_1\cos(\omega t) - C_1\omega t\sin(\omega t) + C_2\sin(\omega t) + C_2\omega t\cos(\omega t)$. The second derivative then becomes: $\ddot{x}_h = -C_1\omega\sin(\omega t) - C_1\omega\sin(\omega t) - C_1\omega^2 t\cos(\omega t) + C_2\omega\cos(\omega t) + C_2\omega\cos(\omega t) - C_2\omega^2 t\sin(\omega t)$. Substituting this into the inhomogeneous ODE, we have:

$$-2C_1\omega\sin(\omega t) - C_1\omega^2 t\cos(\omega t) + 2C_2\omega\cos(\omega t) - C_2\omega^2 t\sin(\omega t) + \omega^2(C_1 t\cos(\omega t) + C_2 t\sin(\omega t)) = C\cos(\omega t)$$

Let us break this up into equations for the different terms:

$$\begin{aligned} -2C_1\omega\sin(\omega t) &= 0 \\ -2C_2\omega\cos(\omega t) &= C\cos(\omega t) \\ -C_1\omega^2 t\cos(\omega t) + C_1\omega^2 t\cos(\omega t) &= 0 \\ -C_2\omega^2 t\sin(\omega t) + C_2\omega^2 t\sin(\omega t) &= 0 \end{aligned}$$

Note that the latter two are true for any values of the constants. The first one requires that $C_1 = 0$, and the second one gives $C_2 = -C/2$. Thus, the particular solution is:

$$x_p(t) = -\frac{C}{2} t\sin(\omega t)$$

Driving an undamped harmonic oscillator at its natural frequency results in linearly growing, unbounded oscillations. This phenomenon is called *resonance*, and although no natural system can exhibit unbounded growth in oscillations, resonance is a profound natural phenomenon, resulting in physical effects such as collapses of bridges if an external force (e.g. wind) happens to match its resonant frequency, or in more useful applications, giving us amplification of radio signals by resonant circuits, as well as sophisticated biological mechanisms that we will discuss later.

14.2 Linearity and vector spaces

In this section we will expand our analysis of linear systems to sketch a broad picture of linear algebra and its fundamental concepts. For a more thorough exposition, see (**strang_linear_2005?**). Whenever we deal with more than one variable, they can be concisely written as a vector of multiple dimensions. We have seen that equations defining linear dynamical systems can be expressed as products of matrices and vectors. In order to

understand how these systems operate and how to express their general solutions, we first need to be specific about the notions of linearity and how it affects vector spaces.

The nomenclature of linearity is derived from the functional description of a line in the plane. Any line passing through the origin can be described as a set of points that can be generated by multiplying them by a single scalar, called the slope, that is, it is generated by a linear transformation $f(x) = ax$. This concept is generalized from dealing with scalars to vectors by the following definition:

Definition

A *linear transformation* or *linear operator* is a mapping L between two sets of vectors with the following properties:

1. *(scalar multiplication)* $L(c\vec{v}) = cL(\vec{v})$; where c is a scalar and $\vec{v}$ is a vector
2. *(additive)* $L(\vec{v}_1 + \vec{v}_2) = L(\vec{v}_1) + L(\vec{v}_2)$; where $\vec{v}_1$ and $\vec{v}_2$ are vectors

We have already seen examples of linear transformations, in the form of matrices multiplying a vector. Matrix multiplication shares the linear property with scalar multiplication, but it transforms vectors to vectors, depending on the size of the matrix, and has more complicated properties. The notion of linearity then leads to the important idea of combining different vectors:

Definition

A *linear combination* of n vectors $\{\vec{v}_i\}$ is a weighted sum of these vectors with any real numbers $\{a_i\}$:

$$a_1\vec{v}_1 + a_2\vec{v}_2 ... + a_n\vec{v}_n$$

Linear combinations arise naturally from the notion of linearity, combining the additive property and the scalar multiplication property. Speaking intuitively, a linear combination of vectors produces a new vector that is related to the original set. Linear combinations give a simple way of generating new vectors, and thus invite the following definition for a collection of vectors closed under linear combinations:

Definition

A *vector space* is a collection of vectors such that a linear combination of any n vectors (with $n \in \mathbb{N}$) is contained in the vector space.

The most common examples are the spaces of all real-valued vectors of dimension n, which are denoted by $\mathbb{R}^n$. For instance, $\mathbb{R}^2$ (pronounced "r two") is the vector space of two dimensional

real-valued vectors such as $(1, 3)$ and $(\pi, -\sqrt{17})$; similarly, $\mathbb{R}^3$ is the vector space consisting of three dimensional real-valued vectors such as $(0.1, 0, -5.6432)$. By taking linear combinations of vectors in the plane, you can generate all the points in the usual Euclidean plane. The real number line can also be thought of as the vector space $\mathbb{R}^1$.

How can we describe a vector space, without trying to list all of its elements? We know that one can generate an element by taking linear combinations of vectors. It turns out that it is possible to generate (or "span") a vector space by taking linear combinations of a subset of its vectors. The challenge is to find a mininal subset of subset that is not redundant. In order to do this, we first introduce a new concept:

i Definition

A set of vectors $\{\vec{v}_i\}$ is called *linearly independent* if the only linear combination involving them that equals the zero vector is if all the coefficients are zero.
($a_1\vec{v}_1 + a_2\vec{v}_2 ... + a_n\vec{v}_n = 0$ only if $a_i = 0$ for all i.)

In the familiar Euclidean spaces, e.g. $\mathbb{R}^2$, linear independence has a geometric meaning: two vectors are linearly independent if the segments from the origin to the endpoint do not lie on the same line. But it can be shown that any set of three vectors in the plane is linearly *dependent*, because there are only two dimensions in the vector space. This brings us to the key definition of this section:

i Definition

A *basis* of a vector space is a linearly independent set of vectors that generate (or span) the vector space.

A vector space generally has many possible bases, as illustrated in figure . In the case of $\mathbb{R}^2$, the canonical basis set is $\{(1, 0); (0, 1)\}$ which obviously generates any point on the plane and is linearly independent. But any two linearly independent vectors can generate any vector in the plane. For instance, the vector $\vec{r} = (2, 1)$ can be represented as a linear combination of the two canonical vectors: $\vec{r} = 2(1, 0) + (0, 1)$. Let us choose another basis set, by taking the vector itself as one of the basis vectors and leaving the second one from the canonical basis: $\{(2, 1); (0, 1)\}$ The same vector can be represented by a linear combination of these two vectors, with coefficients 1 and 0: $\vec{r} = 1(2, 1) + 0(0, 1)$. Since multiple bases are possible, we need a way of evaluating them and changing between them.

14.2.1 inner product and orthogonality

Not all basis sets are created equal. Continuing our geometric analogy, the canonical basis in $\mathbb{R}^2$ is related to the familiar Cartesian coordinates, with two orthogonal axes in the direction of the basis vectors $\{(1, 0); (0, 1)\}$ There are many other, non-orthogonal bases, like $\{(2, 1); (0, 1)\}$

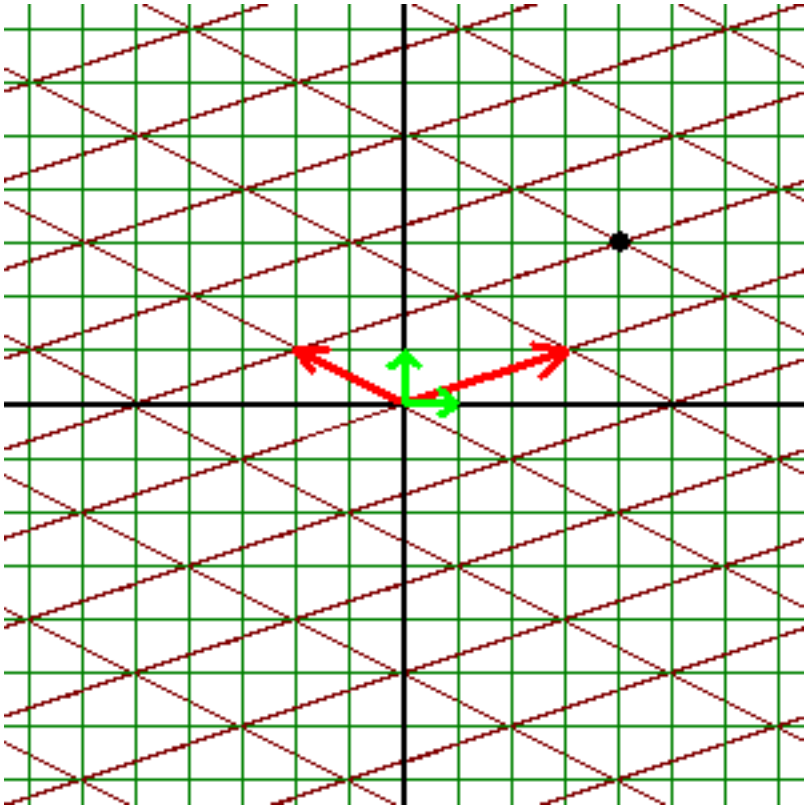


Figure 14.2: Two basis sets in the plane. Green arrows show the canonical Cartesian basis, while the red arrows correspond to the basis set $\{(3,1);(-2,1)\}$. Any point in the plane can be described in terms of both bases. http://www.math.hmc.edu/calculus/tutorials/changebasis/

above, but they are intuitively less economical, since one vector can make a contribution in the direction of the other. This means that a given vector can be represented in non-unique linear combinations with a non-orthogonal basis set. Thus, mathematicians prefer what are called orthogonal bases. In order to define orthogonality, we first introduce this key notion:

> **Definition**
>
> The *inner product* of two vectors u and v is defined as the product of u and the conjugate transpose v^*: $\langle u, v\rangle = uv^*$.

This is the general definition of inner product, rather than the more familiar notion of the dot product, which only applies to real vector spaces. One important difference that the inner product in this definition has a modified commutative property: $\langle u, v\rangle = \overline{\langle v, u\rangle}$ - meaning that switching the order of the inner product results in a complex conjugate of the result. For example: $u = (-i, 5+i)$ and $v = (6, 4-3i)$. $\langle u, v\rangle = -i*6+(5+i)*(4+3i) = -6i+20-3+19i = 17+13i$, and $\langle v, u\rangle = 6i + (4-3i) * (5-i) = 6i + 20 - 3 - 15i - 4i = 17 - 13i$.

The inner product is intimately tied to the geometric notion of direction of vectors in the Euclidean spaces. Let us consider a pair of vectors in $\mathbb{R}^2$ of unit length. If the two vectors have the same direction (are *colinear*), their inner product is equal to 1 or -1, depending on whether they are parallel or anti-parallel - for example, consider the cases of $\langle (1,0), (1,0)\rangle = 1$ and $\langle (0,1), (0,-1)\rangle = -1$. If we rotate one vector relative to the other - for instance, keep the first vector fixed at $(1,0)$ and let the other one be (a,b), with the restriction that $\sqrt{a^2+b^2} = 1$, that is, it is of length 1. Their inner product is clearly equal to a, which you should be able to convince yourself, is the cosine of the angle between the two vectors. With a little bit more work, you can demonstrate that this statement is true for any two unit vectors (those with length 1.)

What about vectors of arbitrary length? First, let us define the notion of length:

> **Definition**
>
> The *length* (also known as the *norm*) of a vector u is defined as the square root of the inner product with itself $||u|| = \sqrt{\langle u, u\rangle} = \sqrt{uu^*}$.

For Euclidean vector spaces, this definition agrees with the familiar Euclidean distance, e.g. in $\mathbb{R}^2$, for $\vec{v} = (x, y)$, $||\vec{v}|| = \sqrt{x^2 + y^2}$.

Using the notion of length, or norm, vectors can be *normalized*, which means divided by the norm, creating a vector of length 1 (a.k.a. unit vector) with the same direction as the original. Since we know that the cosine of the angle θ between the vectors is the inner product of two unit vectors, we have the following relationship between any two vectors:

$$\left\langle \frac{\vec{v}}{||\vec{v}||}, \frac{\vec{u}}{||\vec{u}||} \right\rangle = \cos(\theta) \Rightarrow \langle \vec{v}, \vec{u} \rangle = ||\vec{v}||||\vec{u}|| \cos(\theta)$$

Finally, this leads us to the general statement about orthogonality in vector spaces, which is crucial not only for regular finite-dimensional vector spaces, but also in infinite-dimensional function spaces which we will see later:

> **Definition**
>
> Two vectors are *orthogonal* if their inner product is zero.

Now that we know how to determine whether two vectors are orthogonal, we will call a basis set orthogonal if all pairs of vectors in it are orthogonal. Furthermore, it is typically convenient to require that all the vectors be of unit length, which can be accomplished by normalization. A basis set of mutually orthogonal unit vectors is called *orthonormal*. Typical examples are Cartesian coordinate vectors, such as $\{(1,0,0);(0,1,0);(0,0,1)\}$ in $\mathbb{R}^3$.

14.2.2 projection and decomposition

The basis set of a vector space serves as its defining structure, like a skeleton giving shape to the gelatinous multitude of vectors. Any element in a vector space can be described as a linear combination of the basis set. In the Euclidean plane, the vector $(3,4)$ can be either thought of a collection of two numbers, or as a point with coordinates $x = 3$ and $y = 4$. The latter concept refers to the linear combination of the two standard basis vectors with *coefficients* 3 and 4: $(3,4) = 3(1,0) + 4(0,1)$. The coefficients quantify the overlap of a vector $\vec{r}$ in question with each of the respective basis vectors. Geometrically, if a vector is parallel to a basis vector (of unit length,) then it can be represented as a multiple of the basis vector, and the coefficient will be equal to the length of the vector $\vec{r}$. On the other hand, if a basis vector is orthogonal to the vector $\vec{r}$, the corresponding coefficient will be 0.

The representation of an arbitrary vector of a vector space as a linear combination of a given basis set is called the *decomposition* of the vector in terms of the basis. However, we saw that many possible bases exist for any vector space. Even if we choose only orthonormal bases, there are many possibilities: for instance, in the space of real two dimensional vectors $\mathbb{R}^2$, the standard basis $\{(1,0);(0,1)\}$ can be rotated to produce a different orthonormal basis, e.g. $\{(1/\sqrt{2}, 1/\sqrt{2});\ (-1/\sqrt{2}, 1/\sqrt{2})\}$.

Therefore, the choice of a basis determines how a given vector is represented. The decomposition of a vector in terms of a particular basis is very useful in high-dimensional spaces, where a clever choice of a basis can allow a description of a vector in terms of contributions of only a few basis vectors. The vector may then be represented, given the basis set, with a few coefficients of the relevant basis vectors.

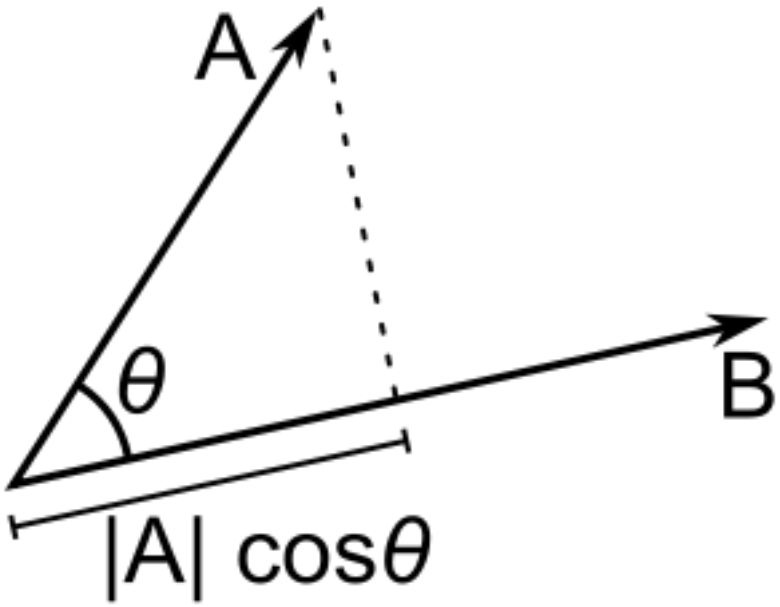


Figure 14.3: Projection of vector A onto vector B, with angle θ between their directions. http://en.wikipedia.org/wiki/Vector_projection

To obtain the coefficients of the basis vectors in a decomposition of a vector $\vec{r}$, we need to perform what is termed a *projection* of the vector onto the basis vectors. Think of shining a light perpendicular to the basis vector, and measuring the length of the shadow cast by the vector $\vec{r}$ onto $\vec{v}_i$. If the vectors are parallel, the shadow is equal to the length of $\vec{r}$; if they are orthogonal, the shadow is nonexistent. To find the length of the shadow, use the inner product of $\vec{r}$ and $\vec{v}$, which as you recall corresponds to the cosine of the angle between the two vectors multiplied by their norms: $\langle \vec{r}, \vec{v} \rangle = ||\vec{r}||||\vec{v}|| \cos(\theta)$ (see figure .) We do not care about the length of the vector $\vec{v}$ we are projecting onto, thus we divide the inner by product by the square norm of $\vec{v}$, and then multiply the vector $\vec{v}$ by this projection coefficient:

$$Proj(\vec{r}; \vec{v}) = \frac{\langle \vec{r}, \vec{v} \rangle}{\langle \vec{v}, \vec{v} \rangle} \vec{v} = \frac{\langle \vec{r}, \vec{v} \rangle}{||\vec{v}||^2} \vec{v} = \frac{||\vec{r}|| \cos(\theta)}{||\vec{v}||} \vec{v}$$

This formula gives the projection of the vector $\vec{r}$ onto $\vec{v}$, the result is a new vector in the direction of $\vec{v}$, with the scalar coefficient $a = \langle \vec{r}, \vec{v} \rangle / ||\vec{v}||^2$.

Example: Let us decompose the vector $(3, 4)$ in a nonstandard basis, e.g. the canonical basis rotated by 45°: $\{(1/\sqrt{2}, 1/\sqrt{2});\ (-1/\sqrt{2}, 1/\sqrt{2})\}$. Use the projection formula above to obtain the coefficients of decomposition. The length of both basis vectors is 1, so the denominator is unity:

$$a_1 = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}) \cdot (3,4) = \frac{7}{\sqrt{2}};\ a_2 = (-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}) \cdot (3,4) = \frac{1}{\sqrt{2}}$$

Therefore, we have the following decomposition:

$$(3,4) = \frac{7}{\sqrt{2}}(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}) + \frac{1}{\sqrt{2}}(-\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$$

Why go through this rigmarole of expressing one vector in terms of N others? A decomposition in terms of a basis is necessary to express a vector in any vector space, as the basis serves as a coordinate system and the coefficients as coordinates of the point designated by the vector. Even in the regular Euclidean space, a vector (e.g. $(3,4)$) requires an implicit basis set in order to make it meaningful. If we choose a different basis set, the formula above allows us to express the vector in a new basis. As mentioned above, in higher-dimensional vector spaces a good choice of basis is important because it can greatly simplify calculations.

14.2.3 general solution of linear ODEs

One very important application of vector decomposition simplifies the solution of linear dynamical systems. Any linear system can be defined in terms of a matrix A, and we saw in Chapter 6 that the solution can be expressed in terms of its eigenvectors. The new concept is that the set of eigenvectors of a matrix forms a basis set for the vector space, provided the matrix is nonsingular. For instance, for a (2x2) matrix A with eigenvectors $\vec{v}_1, \vec{v}_2$ and eigenvalues λ_1, λ_2, the simplest way to compute its effect on a vector $\vec{u}$ is to decompose it in the basis of the two eigenvectors: $\vec{u} = c_1\vec{v}_1 + c_2\vec{v}_2$, and then apply the matrix to the two eigenvectors, using the linear property:

$$A\vec{u} = Ac_1\vec{v}_1 + Ac_2\vec{v}_2 = c_1\lambda_1\vec{v}_1 + c_2\lambda_2\vec{v}_2$$

This gives us a simplification of a matrix multiplication to two scalar multiplications of the two eigenvectors by the eigenvalues λ_i and the coefficients c_i. What is needed is knowledge of the eigenvalues and the eigenvectors, and the coefficients. We have already seen that finding eigenvectors and eigenvalues is a difficult problem, best left to computers. But for a given linear dynamical system, it only needs to be done once. Then, given any initial vector $\vec{u}$, one can decompose it in terms of the normalized eigenvectors, with $c_i = \langle \vec{u}, \vec{v}_i \rangle$. Then we can obtain the general solution for the linear dynamical systems, as a linear combination of the eigenvectors multiplied by exponentials with rates λ_i:

$$\frac{d\vec{x}}{dt} = A\vec{x};\ \vec{x}(0) = \vec{x}_0 \Rightarrow \vec{x}(t) = \sum_i c_i e^{\lambda_i t}\vec{v}_i$$

14.3 Computational: normal mode calculations

14.3.1 harmonic analysis of coupled oscillators

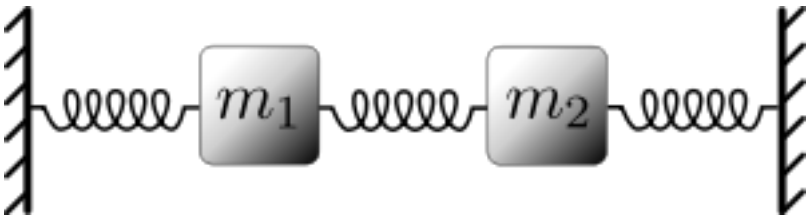


Figure 14.4: Illustration of a model of two coupled springs attached to a wall. http://en.wikipedia.org/wiki/Normal_mode

In the modeling section we saw a model describing the dynamics of two objects connected by a spring (**fig:coupled_springs?**). We used Hookean potentials to describe the interactions between two masses, which correspond to linear forces. This results in the equations of motion that are a linear system of second-order ODEs. Before we have only seen systems of first-order ODEs, but we can make use of linear algebra to find the solutions. We take the matrix of the system,

$$A = \frac{k}{m}\begin{pmatrix} -1 & 1 \\ 1 & -1 \end{pmatrix}$$

This matrix is known as the *Hessian* matrix, which means the matrix of second derivatives of the potential function. The eigenvalues of this particular Hessian matrix are 0 and $-2k/m$. With a little bit of work, we can find the corresponding eigenvectors: $v_1 = (1, 1)$ for the 0 eigenvalue and $v_2 = (1, -1)$ for the -2 eigenvalue. Now consider how the solutions behave in terms of these basis vectors. If the initial condition corresponds to the vector v_2, then the ODE becomes, in matrix and vector form:

$$\ddot{v}_2 = Av_2 = -2\frac{k}{m}v_2$$

We know from previous examples that the solutions of this equation (assuming $k, m > 0$) are purely oscillatory, with frequency $\sqrt{2k/m}$. Note that unlike in the previous first-order ODE examples, negative eigenvalues mean *oscillatory* behavior, rather than exponential decay. The form of the eigenvector $(1, -1)$ indicates that two masses will oscillate with opposite phases: when x is moving right, y is moving left, and vice versa.

The eigenvalue of 0 is a special case. We can say that it indicated an oscillation frequency of 0, since the second derivative equals 0, which implies a constant velocity. This is a called

a *translational* motion, and the form of the eigenvector $(1, 1)$ indicates that x and y move in concert, either left or right.

The eigenvectors of the Hessian of a system of coupled linear oscillators are called *normal modes*. Each one describes a *collective vibrational motion* of a particular frequency, in which the particles participate with relative coefficients given by the normal mode. The corresponding eigenvalues correspond to the squared frequencies of the collective oscillation described by the normal mode.

14.3.2 normal mode calculations

In order to perform normal mode calculations, one needs two things: the Hessian matrix and the ability to diagonalize it (find its eigenvalues and eigenvectors.) Let us consider that we have a system of coupled Hookean potentials, each "spring" connecting nodes i and j with its own force constant k_{ij}. Note that Hookean potentials, so if node i is connected to mode j, then node j is connected to mode i with the same force constant. Then we construct the Hessian matrix as follows:

obtain list of 3D coordinates for N nodes $X_i = (x_i, y_i, z_i)$ set cutoff distance R define a distance function $dist(X_i, X_j)$ (returns distance between two 3-dimensional vectors) pre-allocate Hessian matrix H (N by N) with 0s $H[i, j] \leftarrow -k$ $H[j, i] \leftarrow -k$ $H[j, j] \leftarrow H[j, j] + k$

Consider three nodes connected as a linear chain, with node 1 connected to node 2 with force constant k_1, and node 2 connected to node 3 with force constant k_2, the Hessian matrix is:

$$H = \begin{pmatrix} k_1 & -k_1 & 0 \\ -k_1 & k_1 + k_2 & -k_2 \\ 0 & -k_2 & k_2 \end{pmatrix}$$

Now that we have constructed the Hessian matrix, we need to diagonalize it to find the normal modes. We will not describe the algorithms to find the eigenvalues and eigenvectors of matrices, as those deserve their own chapter, but this topic is well addressed in (**press_numerical_2007?**). Let us stipulate that one can use a function that will produce a set of eigenvectors, sorted by the magnitude of the corresponding eigenvalues.

For the system of three nodes described above, let both springs have the same force constant of $1 (k_1 = k_2 = 1.)$ Then the system has the following eigenvectors and eigenvalues:

$$\vec{v}_1 = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \lambda_1 = 0;\ \vec{v}_2 = \begin{pmatrix} -1 \\ 0 \\ 1 \end{pmatrix} \lambda_2 = 1;\ \vec{v}_3 = \begin{pmatrix} 1 \\ -2 \\ 1 \end{pmatrix} \lambda_3 = 3$$

This demonstrated that a linear chain of three oscillators that are not attached to any other object has three normal modes of different frequencies. The first mode has zero frequency,

which is known as a *rigid-body* mode, in which the entire system moves together, without changes in relative distances. The second mode has frequency 1, and in it the two end nodes move in opposite directions of each other. The third mode has frequency $\sqrt{3}$, and in it the end points move in the same direction, while the middle node moves in the opposite direction with twice the amplitude. This analysis predicts that all motions of the three nodes can be described in terms of the three normal modes.

14.4 Normal mode analysis of biomolecular structures

14.4.1 biomolecular structures as elastic solids

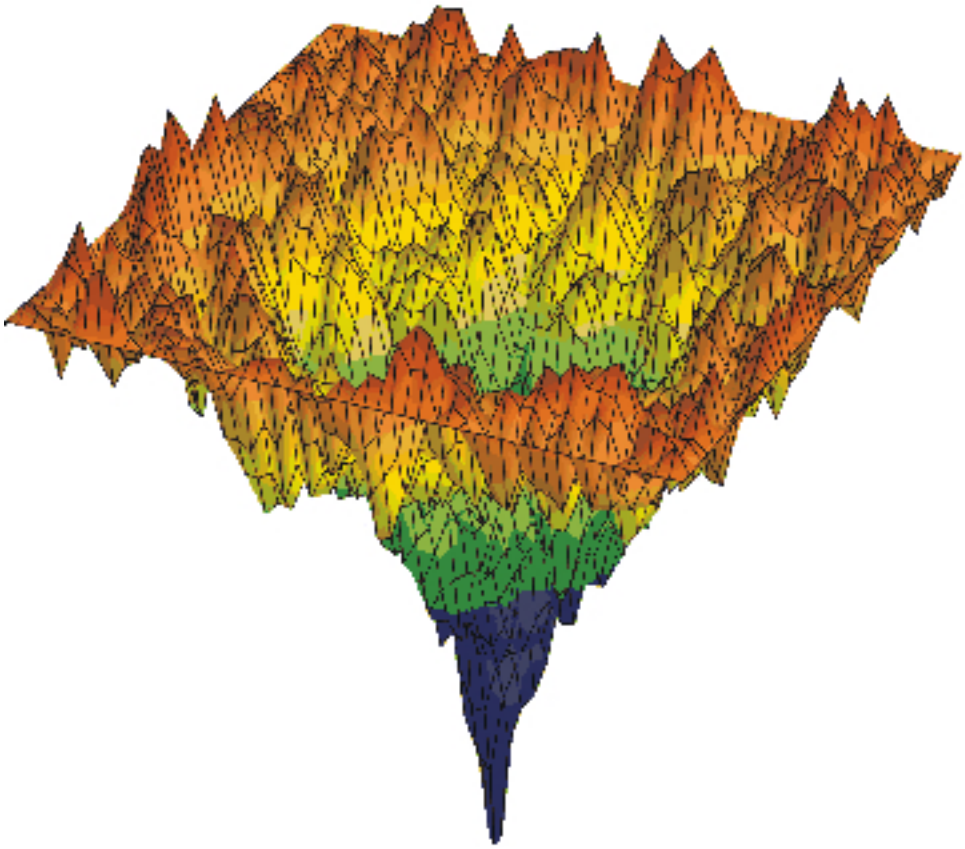


Figure 14.5: Cartoon depiction of the potential function of a protein, reduced to two variables. Vertical dimension indicates relative energy level of different conformations. The sharp well in blue is the native conformation. http://www.btinternet.com/~martin.chaplin/protein2.html

Proteins inside cells fold into exquisitely precise structures, which enable them to perform a

tremendous variety of functions, from catalyzing biochemical reactions to binding signaling molecules. The location of amino acids residues and all the atoms, is called the protein's structure. Determining the structure of a protein is laborious, although thanks to technological advances structural determination is less difficult than in the past. The knowledge of a protein's structure provided a great deal of important information to biochemists, for instance the location of the catalytic active side. However, it does not tell the whole story of how the protein functions.

The structures of proteins and other biomolecules are not static, but instead fluctuate around the most energetically favorable conformation. These fluctuations are cause by the jiggling of the surrounding molecules, such as waters due to the thermal motion at the molecular level. A protein molecule can be thought of as a system with a potential, shaped by the interactions between all the atoms in the molecule, and with the surrounding solvent. The variables of the system are the positions of all the atoms, which means that the system has thousands or tens of thousands of variables. Figure (**fig:prot_land?**) shows a cartoon of the potential energy function of a complex molecule. It shows a sharp well with the preferred folded state at the minimum. Thermal noise adds random kinetic energy to the system, causing the conformations to jiggle in the well, and occasionally causing major conformational changes or even unfolding.

Normal modes are used to study the flexibility of molecular structures. The basic assumption is that the molecules behave as coupled harmonic oscillators, with each atom connected to other atoms by harmonic potentials. This assumption makes physical sense at the bottom of the potential well, near the native conformation, where the potential must have close to quadratic shape, and therefore the restoring forces are nearly linear with displacement.

Various models exist for defining the connections in protein structure between different particles, which could be atoms or amino acid residues, or even blocks of many residues. The connections may be based of physical chemical forces, such as chemical bonds, van der Waals forces, and electrostatic interactions, or may be based on a simple model where parts of the protein in proximity are assumed to interact as if bound by a linear spring. These interactions yield a matrix equivalent to the one we saw for two coupled oscillators, known as Hessian matrix. Figure [fig:calm_gnm] shows the harmonic potentials used to model the structural dynamics of the protein calmodulin. Connections were chosen based on distance proximity between residues. (**cui_normal_2005?**).

14.4.2 sorting normal modes by frequency

The normal modes and the corresponding frequencies are determined by computationally finding the eigenvectors and eigenvalues of the Hessian matrix. For practical purposes, the most interesting modes are those with lowest (but nonzero) frequencies, because they correspond to the slowest and most global collective motions, as opposed to high-frequency vibrations, which

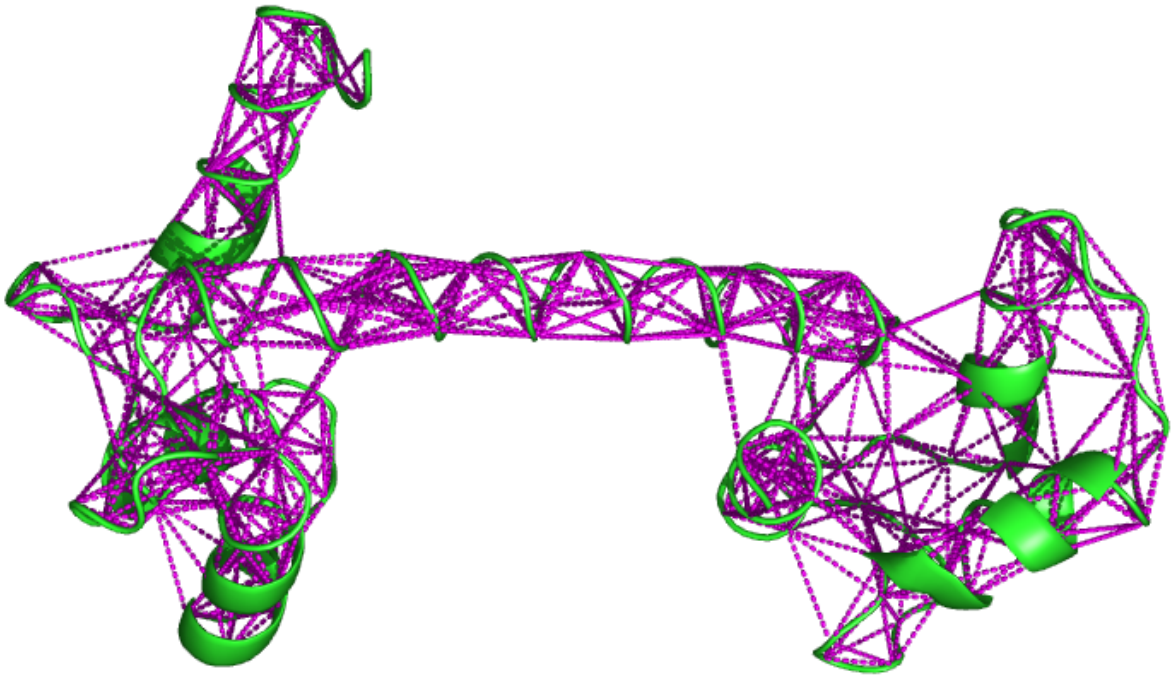


Figure 14.6: Harmonic potential model of the protein calmodulin. Green indicates the backbone of the molecule, maroon lines indicated harmonic interactions between residues.

are restricted both in amplitude and in scope. Intuitively, the lowest frequency modes correspond to the shallowest directions in the potential energy well. Given a reasonable amount of thermal noise, the protein structure is most likely to be deformed along the shallow directions, instead of climbing up the steep directions.

The utility of normal mode analysis of biological molecules lies in obtaining the preferred modes of flexibility from a static structure, which allows biochemists to better understand the mechanism of the molecular function. For instance, in studying the mechanism of opening or closing of an enzyme binding site, normal modes can generate a hypothesis about the intermediate conformations, and help predict which residues play a key role. Figure [fig:calm_nma] shows the directions of the lowest frequency mode of calmodulin, which undergoes a large conformational change in response to binding of calcium ions. The arrows show the extent of involvement of each amino acid residue, as well as the direction of preferred fluctuations. To summarize, changing the coordinate system from individual positions to collective normal modes of motion simplifies the systems and generates predictions relevant for understanding the function of biomolecules.

Figure 14.7: Lowest frequency mode predicted from normal mode analysis of calmodulin. Arrows indicate the direction and magnitude of flexibility associated with each residue.

15 Fourier series: decomposition by frequency

In this chapter we present the method of Fourier analysis, which extracts information about frequency from a data set. In the modeling section we motivate the notion of frequency analysis with the example of electro-encephalogram (EEG) recordings of electrical activity of the central nervous system. The analytic section develops the mathematical notion of function spaces, and the basis of sine and cosine functions. This is used for Fourier decomposition, or description of an arbitrary periodic function as a sum of sines and cosines. In the computational section, we describe the computation of Fourier decomposition using an efficient algorithm called the Fast Fourier Transform, which is a seminal development in the history of computation.

15.1 Periodic signals

15.1.1 amplitude, period, and frequency

Many biological processes are periodic, or repetitive, with a particular pattern that serves biological needs; common examples are waves of activity in the heart muscle, repeated spikes of voltage across neural membranes, and daily Circadian rhythms in physiological regulation. It is highly useful to measure the properties of these periodic activities, and to describe them using idealized mathematical functions, specifically sines and cosines.

As a reminder sines and cosines both have period 2π, but the sine is an odd function: $\sin(x) = -\sin(-x)$, whereas the cosine is an even function: $\cos(x) = \cos(-x)$. Furthermore, by adding a couple of parameters, one can produce a sine or cosine wave of any period and any amplitude. In the following expression, the *period* of both the sine and the cosine is L, and thus the *frequency* is $1/L$, while the amplitude of the sine is A and that of the cosine is B.

$$A\cos(2\pi x/L);\ B\sin(2\pi x/L)$$

15.1.2 brain waves in EEG

Of course, not all periodic functions are sines and cosines, but sines and cosines can be used to describe the types of frequencies present in a periodic signal. Consider the following electroencephalograph (EEG) data collected from electrodes on the scalp of a human:

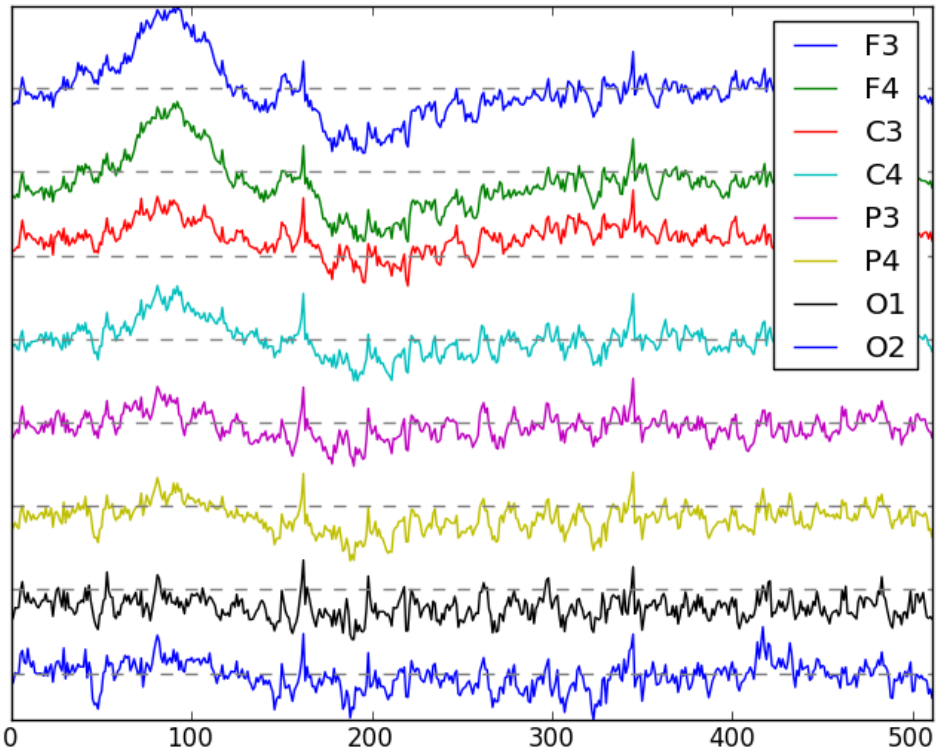


Figure 15.1: EEG signal recordings from multiple electrodes https://www.cs.colostate.edu/eeg/data/json/doc/tutorial/_build/html/getting_started.html#getting-started

By inspection, it appears as if there are some periodic processes producing these data, but these are not neat periodic sine or cosine waves. Instead, we have many different overlapping signals, produced by huge numbers of electrical pulses in the brain, each with different frequency. In order to analyze this signal, we need to decompose it into contributions of different frequencies. Signals of different frequencies,called brain waves in neuroscience, serve distinct purposes: for instance, different stages of sleep can be characterized by the frequencies of the brain waves. In the following section we will learn how to describe complex, periodic data sets, such as the one in {numref}`fig-eeg-plot`, in terms of the contributions, or amplitudes, of sines and cosines of different frequencies. This way we can quantify the presence of different types of brain waves in a given EEG recording.

15.2 Periodic functions as a basis set

We now introduce the idea of a space of functions, instead of vectors, and describe how to decompose any given function in terms of a basis of other functions. Joseph Fourier postulated in 1822 that any function can be described by an infinite sum of sine functions. Some of the details were incorrect, but he introduced an a revolutionary concept that has found fundamental applications in a multitude of fields of science, from acoustics to medical imaging. The fact is that any function on a finite interval of length L (or a periodic function with period L, which is equivalent) can be represented exactly by an infinite sum of sines and cosines, plus a constant term:

$$f(x) = a_0 + \sum_{k=1}^{\infty} a_k \cos(2\pi kx/L) + \sum_{i=1}^{\infty} b_k \sin(2\pi kx/L)$$

Notice that this is the same concept of decomposition in terms of a basis set. Any such function can be written as the sum of sines and cosines, and only the coefficients are different for different functions. The main difference is that vector spaces (such as $\mathbb{R}^3$) have finite basis sets of vectors, while a function space (e.g. the space of all functions with period L) has an infinite basis set of functions (e.g. sines and cosines with different frequencies.) To be specific, let us define these concepts.

> **Definition**
>
> A *function space* is a collection of all functions defined over a given domain, for example the interval $[-L/2, L/2]$, that have a finite norm, to be defined below.

The notion of the norm of a function is similar to the norm, or magnitude of a vector. The reason for restricting function spaces to functions with a finite norm, is to ensure that computations of various quantities of interests are valid and do not blow up. Now let us define the function norm:

Definition

The *norm* of a function $f(x)$, denoted $||f||$, is a mapping from the function space into nonnegative real numbers, which obeys the following rules:

1. $||f|| = 0$ iff $f(x) = 0$ (the function is zero everywhere)
2. $||af|| = a||f||$ for any real number a
3. $||f + g|| \leq ||f|| + ||g||$ for any functions f and g in the function space (triangle inequality)

The norm that we will utilize in the function spaces is called the L^2 norm and it is defined as follows:

Definition

The L^2 norm of a real-valued function $f(x)$ over an interval $[-L/2, L/2]$ is defined as follows:

$$||f|| = \sqrt{\int_{-L/2}^{L/2} f(x)^2 dx}$$

The L^2 norm in function spaces is the square root of the integral of the square of the function values over the interval of its definition (which can be extended to the entire real number line, in the limit of $L \to \infty$). This is the equivalent of the Euclidean distance norm in vector spaces, which if you recall is the square root of the sum of squares of all the components of the vector. There are many possible norms of function spaces, but the L^2 norm is mathematically special, because it is derived from the inner product of the function space:

Definition

The *inner product* between two functions f and g defined on the same interval $[-L/2, L/2]$ is:

$$< f, g >= \int_{-L/2}^{L/2} f(x)g(x)dx$$

The function norm can be defined in terms of the inner product: $||f(x)|| = \sqrt{< f(x), f(x) >}$. This defines the machinery for computing the "size"" of a function, measured by its norm, as well as the "similarity" between two functions, measured by the inner product. If two functions f and g are very similar, the inner product is close to the square of the norm of f (or g). If f and g are very similar, but flipped by a negative sign, the inner product is close to negative of the square of the norm of f (or g). On the other hand, if the two functions are dissimilar -

loosely speaking, if f is positive, g is sometimes positive, sometimes negative, then the product of the values of f and g is sometimes positive and sometimes negative, and thus its integral will add to be close to zero. This is how one can define orthogonality of two functions:

> **Definition**
>
> Two functions f and g in a function space are *orthogonal* if $< f, g >= 0$.

Now that we can describe the coefficients for the sines are cosines, by using the inner product of the function we are decomposing, with the basis functions (sines and cosines) in a manner analogous to the basis decomposition described for vector spaces in Chapter 8. The coefficients for the sines and cosines of the Fourier decomposition of a function $f(x)$ with period L are found as follows:

$$a_k = \frac{< f(x), \cos(2\pi kx/L) >}{< \cos(2\pi kx/L), \cos(2\pi kx/L) >} = \frac{2}{L}\int_{-\frac{L}{2}}^{\frac{L}{2}} f(x)\cos(2\pi kx/L)dx$$

$$b_k = \frac{< f(x), \sin(2\pi kx/L) >}{< \sin(2\pi kx/L), \sin(2\pi kx/L) >} = \frac{2}{L}\int_{-\frac{L}{2}}^{\frac{L}{2}} f(x)\sin(2\pi kx/L)dx$$

$$a_0 = \frac{1}{L}\int_{-\frac{L}{2}}^{\frac{L}{2}} f(x)dx$$

The factor of $L/2$ in front of the integrals serves to divide out the norm of the basis function. Note that the constant term c_0 is the mean value of the function on the interval $(-L/2, L/2)$ - it moves the function up or down, while the sines and cosines all have the mean value of zero.

15.2.1 Fourier decomposition of a square wave

Take a common example of a simple function subject to Fourier decomposition: a square wave, which is a function which equals one constant value (e.g. -1) for half of the interval, and another constant (e.g. 1) for the other half of the interval. Here is how we find the coefficients of the sines and cosines analytically:

$$a_k = \frac{2}{L}\int_{-\frac{L}{2}}^{\frac{L}{2}} f(x)\cos(2\pi kx/L)dx = \frac{2}{L}\int_{-\frac{L}{2}}^{0} -\cos(2\pi kx/L)dx + \frac{2}{L}\int_{0}^{\frac{L}{2}} \cos(2\pi kx/L)dx =$$

$$= \frac{2}{L}\left[\frac{L}{2\pi k}\sin(2\pi kx/L)|_{-\frac{L}{2}}^{0} - \frac{L}{2\pi k}\sin(2\pi kx/L)|_{0}^{\frac{L}{2}}\right] = \frac{1}{\pi k}\left[0 - 0 - 0 + 0\right] = 0$$

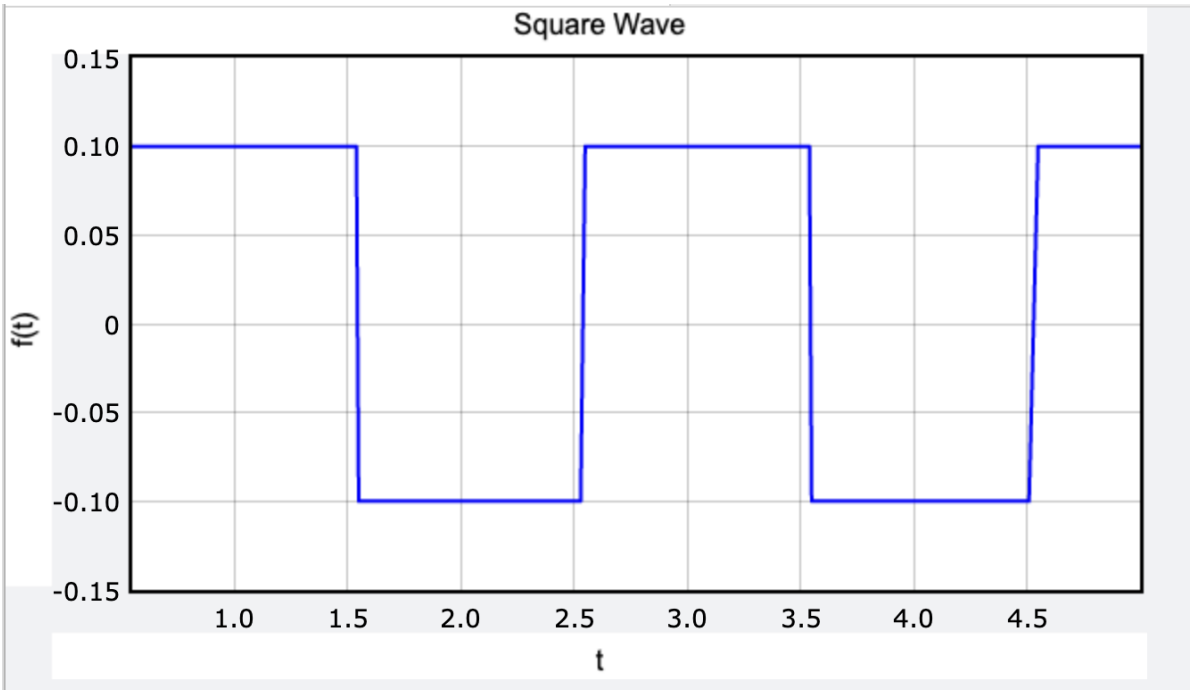


Figure 15.2: A square wave function alternates between two constant values. https://levelup.gitconnected.com/representing-a-square-wave-with-a-fourier-series-and-python-6d43beb19442

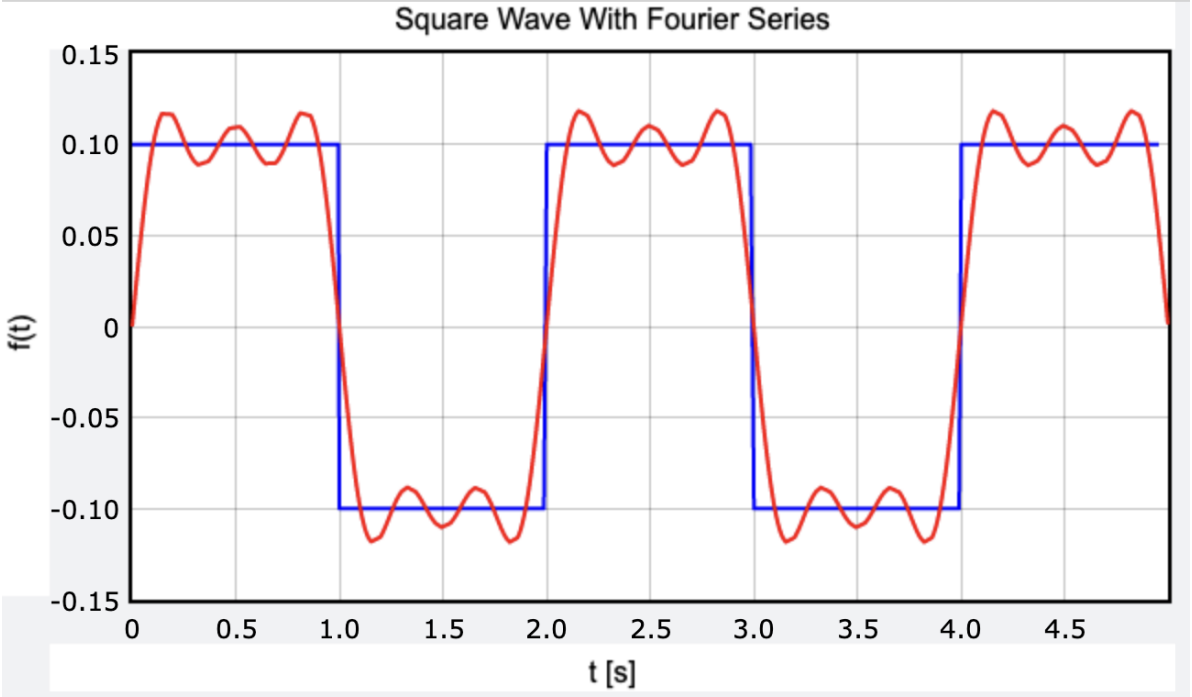


Figure 15.3: Approximations of a square wave function with five terms of the Fourier series. https://levelup.gitconnected.com/representing-a-square-wave-with-a-fourier-series-and-python-6d43beb19442

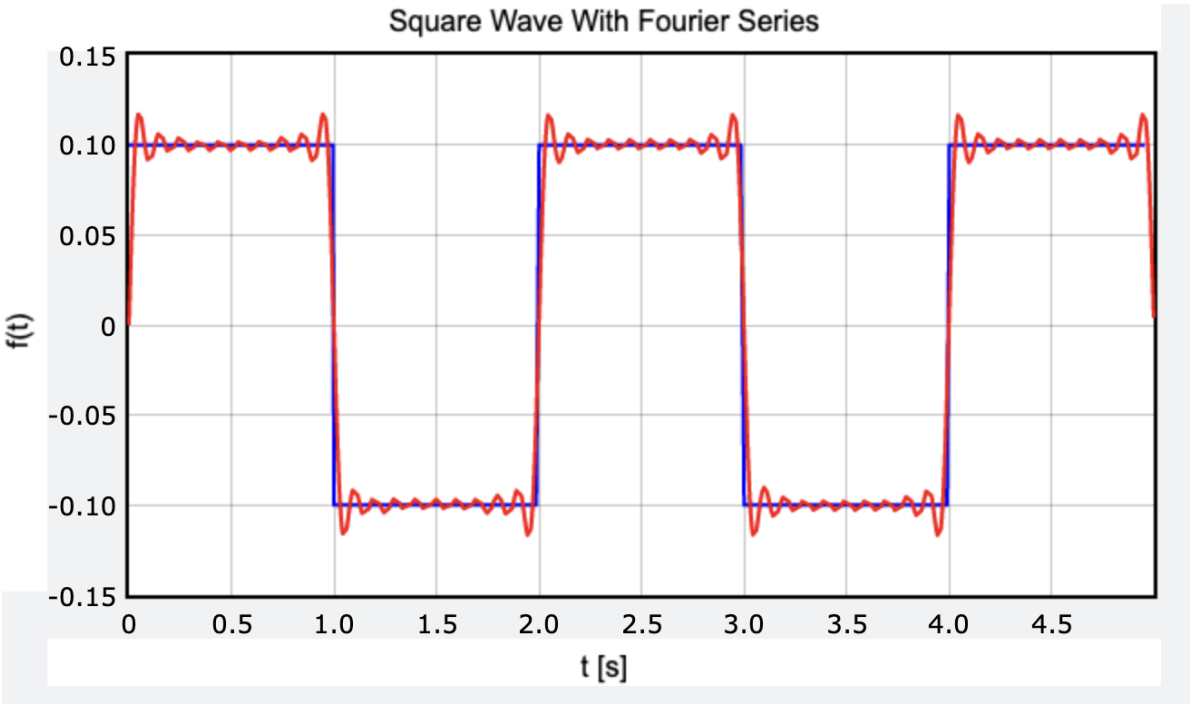


Figure 15.4: Approximations of a square wave function with twenty terms of the Fourier series.https://levelup.gitconnected.com/representing-a-square-wave-with-a-fourier-series-and-python-6d43beb19442

$$b_k = \frac{2}{L}\int_{-\frac{L}{2}}^{\frac{L}{2}} f(x)\sin(2\pi kx/L)dx = \frac{2}{L}\int_{-\frac{L}{2}}^{0} -\sin(2\pi kx/L)dx + \frac{2}{L}\int_{0}^{\frac{L}{2}} \sin(2\pi kx/L)dx =$$

$$= \frac{2}{L}\left[-\frac{L}{2\pi k}\cos(2\pi kx/L)|_{-\frac{L}{2}}^{0} + \frac{L}{2\pi k}\cos(2\pi kx/L)|_{0}^{\frac{L}{2}}\right] = \frac{1}{\pi k}\left[1-\cos(\pi k)+1-\cos(\pi k)\right] = 0 \; or \; \frac{4}{\pi k}$$

The coefficients for the cosines are zeros, but there are nonzero values for the sines, when k is odd, and $\cos(\pi k) = -1$. When k is even, $\cos(\pi k) = 1$, and the expression reduces to 0. Thus, the Fourier series representing the square wave with period L looks like this:

$$f(x) = \frac{4}{\pi}\sin(2\pi x/L) + \frac{4}{3\pi}\sin(2\pi 3x/L) + \frac{4}{5\pi}\sin(2\pi 5x/L) + ...$$

Notice how the coefficients decline for higher frequency terms. This means that one can take a finite, often just a handful of lowest-frequency terms and have a decent approximation of the function. This is typical for most reasonable functions, as we will discuss below.

15.2.2 complex Fourier series

We saw that periodic functions can be approximated by a sum of sines and/or cosines of different frequencies, which is called the Fourier series. Because both sines and cosines are needed to represent a function that is not purely odd or even, a more compact complex representation of the Fourier series is used:

$$f(x) = \sum_{n=-\infty}^{\infty} c_k e^{i2\pi xk/L}$$

Here L denotes the length of the period of the function $f(x)$, and c_n are the complex coefficients of the Fourier series, each corresponding to the frequency n/L. They are found by the same integration as the ones for sine and cosines series:

$$c_k = \frac{1}{L}\int_{-L/2}^{L/2} f(x)e^{-i2\pi kx/L}dx; \;\; -\infty < k < \infty$$

The coefficient c_0, as can be seen from the integral, is the average value of $f(x)$ on the interval $[-L/2, L/2]$.

In the complex Fourier series, positive and negative frequencies are used in order to combine both sines and cosines into the same series, by using the expressions $e^{i2\pi xk/L} + e^{-i2\pi xk/L} =$

$2\cos(\pi xn/L)$ and $e^{i2\pi xn/L} - e^{-i2\pi xn/L} = 2i\sin(2\pi xn/L)$. Thus, the complex coefficients c_n are related to the coefficients a_k and b_k of the cosine and sine series as follows:

$$c_k = \frac{a_k - ib_k}{2};\ c_{-k} = \frac{a_k + ib_k}{2};\ k \geq 1$$

Note that as long as a_n and b_n are real (which is the same as saying the function $f(x)$ is real) the coefficients with opposite signs will be complex conjugates of each other.

15.3 Discrete Fourier Transform

Now let us consider a series of data points, instead of idealized functions, since in reality the data are never described by perfect continuous functions. Let us suppose that they come from measuring a certain function $f(x)$ over a range of length L at regular intervals. This is called *sampling* of the function and the sampling interval (in units of x) between the sample points is called $\Delta = L/N$, where N is the number of sample points. As a result, we get a sequence of N measurements $\{x_i\}$. In order to decompose the sampled inputs into their frequency components, we need to find the coefficients of the Fourier series. Let us use the notation $\{X_k\}$ for the Fourier coefficients, and define the following the *Discrete Fourier Transform*:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N}$$

It is called the Discrete Fourier Transform because it is based on the finite data set, and thus computation of coefficients requires summation instead of integration. Let us consider what frequency each coefficient X_k corresponds to. When $k = 0$, $e^0 = 1$ and we just have the sum of all the sample points. This is called the zero frequency term, or sometimes the DC (direct current) term by electrical engineers. The other terms have frequencies given by k/N, all the way up to $(N-1)/N$. Note here that we assume for convenience that the interval Δ is 1, so the frequency corresponds to the fraction of points in the cycle (e.g. k/N). The frequency ranges from the lowest of $1/N$ to the highest of $(N-1)/N$. However, the highest frequency is actually equivalent to $-1/N$, because going around the $(N-1)/N$ fraction of a circle in one direction is the same as going $1/N$ fraction in the opposite direction.

Thus, the first half of the coefficients correspond to positive frequencies in increasing order, until the frequency $1/2$ is reached, and then the coefficients correspond to negative frequencies in descending order of the absolute value. In fact, if the input points are real, the coefficients of positive and negative frequencies are symmetric, and so for the Pyhon indexing above, we have: $C(k+1) = C(N-k+1)$ (for $k > 0$). This is because the complex terms have to add up to real numbers, so this ensures that terms with opposite frequencies are complex conjugates (convince yourself of this fact).

15.4 sampling theorem and aliasing

By representing a periodic function $f(x)$ in terms of the Fourier series, we reduce its description to the values of the coefficients c_k. We say that this set of coefficients is a representation in the *frequency domain* as opposed to the time or space domain of the original variable x. This is very useful for analyzing the types of frequencies that a function contains.

To elaborate, c_k gives the weight of the sine or the cosine function with frequency k, that is, one which has k repetitions within the period L. Higher frequency terms are more wiggly, and are needed to represent functions with high slopes. Lower frequency terms represent the larger, slower varying shape of the function. For any reasonable function, the higher frequency terms will generally have smaller coefficients than lower-frequency terms, and for really high frequencies will be very small. This enables one simple use of the Fourier series: a periodic function can be approximated by a few lower-frequency terms, so it can be represented by a few numbers. This has great applications in image and sound compression.

The highest frequency possibly in a sample of N data points is called the *Nyquist critical frequency* f_c. It depends on the sampling interval Δ like this: $f_c = 1/2\Delta$. The intuition behind this is that in order to detect a frequency f, one needs to make at least two measurements during one period $1/f_c$. (Convince yourself of this by drawing a sine wave and sampling it two or fewer times per period.) Since each measurement takes Δ units, the highest frequency we can sample is $1/2\Delta$. This leads to a remarkable result, called the Sampling Theorem:

> **Theorem**
>
> If a function $h(x)$ contains no higher frequencies than some f_c, more precisely its Fourier transform $\hat{h}(f) = 0$ for any $f > f_c$, then this function can be completely represented by a sample with the interval Δ such that $\Delta < 1/2f_c$.

Practically, this means that any function that does not change too abruptly (which requires higher frequency terms) can be represented by a discrete set of low-frequency terms. However, in practice there is always noise or abrupt changes, so one cannot have what is known a *bandwidth limit* (meaning the band of frequencies contributing to the function is limited). Then, when we try to represent a function with a discrete set of points that we Fourier-transform, when we perform the inverse FT, we get error due to lack of the high-frequency terms. This is called *aliasing error*.

15.5 Fast Fourier Transform

Now let us get down to the business of computing the Fourier decomposition of an input of N data points. In equation **??** in the discrete Fourier transform section, we found an analytic formula for finding the coefficients of a complex Fourier series by summation of N components.

In order to obtain all N Fourier coefficients, we would need to perform approximately N^2 operations (N multiplications plus $N-1$ additions for each of the N coefficients). This means that as the number of inputs grows, the computational cost of performing the Discrete Fourier Transform grows quadratically. This is a major problem because Discrete Fourier Transforms are so ubiquitous - they are at the heart of graphics engines, audio and image analysis, and many other computationally intensive applications. In this section we will describe a truly transformational algorithm which dramatically reduces the computational cost of a DFT, descriptively called the Fast Fourier Transform (FFT). Specifically, we will describe the classic Cooley-Tukey algorithm [**?**], which was the first type of FFT; subsequently other variations were developed, which have some advantages, but the original FFT is so fundamental to modern computing that I will present it in this section.

15.5.1 splitting the data into even and odd inputs

Let the set of inputs for the Discrete Fourier Transform consist of N numbers, $\{x_n\}$. This number N could be large, and practicing computational scientists have thought about a way of simplifying the calculation. It turns out that there is a beautiful symmetry in the Fourier calculation that enables the calculation of the Fourier coefficients of N data points in terms of the Fourier coefficients of two halves of the data set: the even and the odd numbered inputs. First, let us write down the expression in equation **??** in terms of sums of the $N/2$ even and the $N/2$ odd inputs, as follows:

$$X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N} = \sum_{m=0}^{N/2-1} x_{2m} e^{-i2\pi(2m)k/N} + \sum_{m=0}^{N/2-1} x_{2m+1} e^{-i2\pi(2m+1)k/N}$$

The two sums look very similar to the sum that produces the Fourier coefficients for the N inputs. In fact, the first sum, is identical to the DFT of the even-numbered inputs, which we will denote as $X_k^{(e)}$. The second sum can be transformed by taking the factor $e^{-i2\pi k/N}$ out of the sum into the sum for the DFT of the odd-numbered inputs , which we denote $X_k^{(o)}$. Conventionally in Fourier literature, the factor $e^{-i2\pi k/N}$, which is the Nth root of unity raised to the kth power, is called the *twiddle factor*, and is notated w^k (for N inputs). Therefore, we have the following expression:

$$X_k = X_k^{(e)} + w^k X_k^{(o)}$$

Note that this formula works for $0 \leq k \leq N/2 - 1$, since the DFTs of halves of the data set have only half of the outputs ($N/2$). However, due to its periodicity, the DFT repeats itself for coefficients that go beyond the size of the inputs; for a DFT of size N, $X_k = X_{k-N}$. Therefore, we can compute the other half of the Fourier coefficients of the original data set ($0 \leq k \leq N/2 - 1$) to obtain the same formula:

$$X_k = X_k^{(e)} + w^k X_k^{(o)}$$

This result, known as the *Danielson-Lanczos lemma*, allows the calculation of a DFT with N inputs, in terms of the coefficients of two DFTs with $N/2$ inputs. It is clear that even applying this splitting once leads to computational advantage, since as we noted above, DFT requires on the order of N^2 arithmetic operations. Thus, performing DFTs on half of the number of inputs will reduce the number of calculations by a factor of 4, and since it is performed for each half of the data, this results in approximately two-fold reduction in operations, as it requires only on the order of N additional operations to reassemble the full DFT.

15.5.2 recursive splitting and reassembly

If splitting the problem in half once reduces the computational cost, why not do it again? and again? This was the idea that Cooley and Tukey came up with in 1965. For example, if the number of inputs is divisible by 4, one can split the data sets into even- and odd-numbered halves, and then split each of those into even and odd-numbered halves, and perform DFT on the quarter-data sets separately. The resulting four sets of Fourier coefficients will be labeled $\{X^{(ee)}\}$, $\{X^{(eo)}\}$, $\{X^{(oe)}\}$, and $\{X^{(oo)}\}$ (e.g. the second one represents the quarter of data set that had even indices in the original set, and odd indices in the even half, corresponding to indices 2,6,10, etc.), and they can be recombined in order to compute the Fourier coefficients of the entire set. Using the above formula, we can find the the expression for reassembling the four quarter-size DFTs to compute the DFT. The twiddle factor for quarter-size data sets is $e^{-i2\pi k/(N/2)} = e^{-i2\pi 2k/N} = w^{2k}$. Therefore, the formula for the DFT, for indices $0 \leq k \leq N-1$ is:

$$X_k = X^{(ee)} + w^{2k} X^{(eo)} + w^k X^{(oe)} + w^{3k} X^{(oo)}$$

We can continue further dividing the data into halves and reassembling the resulting DFT coefficients, as long as the number of data points in the subsets is divisible by two. In order to achieve maximal decomposition, let us assume that the number of inputs is a power of 2 ($N = 2^M$). Then after M such divisions into even and odd subsets, the data are subdivided into N subsets of single values. The DFT of a single data point, by the formula in equation above is just the data point. Therefore, for any data point with a given pattern of even and odd divisions, e.g. *ooeeo*..., there is a corresponding singlet DFT with index n:

$$x_n = X^{(ooeeo\ldots)}$$

The question is, how does the index n of the data point correspond to the string of even and odd divisions in the DFT? The answer turns out to be simple and elegant in binary representation of indices. Consider, for example, a data set of four input values, indexed $\{x_0, x_1, x_2, x_3\}$. The first division splits them into $\{x_0, x_2\}$ and $\{x_1, x_3\}$, and the second subdivides them into singleton sets: $\{x_0\}, \{x_2\}, \{x_1\}, \{x_3\}$. The rearrangement of indices due

to divisions into evens and odds is captured by *bit reversal* of the binary indices. In binary, we can write $0 = 00; 1 = 01; 2 = 10; 3 = 11$. Reversing the bits, that is re-writing the binary numbers from right to left, yields: $00 = 0, 10 = 2, 01 = 1; 11 = 3$, which is exactly the order we produced by two splittings. Therefore, we can find the DFTs of each of the resulting singleton sets by reordering the input values by bit-reversal and then recombining them using the Danielson-Lanczos formula above.

Example. Let us calculate the DFT for the data set $\{x_0, x_1, x_2, x_3\} = \{2, -1, 2, -1\}$. As we saw above, we split the four inputs into halves twice until we are left with singleton sets, which are then arranges as follows: $ {x_0, x_1, x_2, x_3 } = {2, 2 -1, -1 }$. Then we recombine the value with appropriate twiddle factors to calculate the DFT. First, calculate the twiddle factors for DFT with $N = 2$:

$$w_0 = 1;\ w_1 = e^{-i\pi} = -1$$

$$X_0^{(e)} = x_0 + x_2 = 4$$

$$X_1^{(e)} = x_0 + w^1 x_2 = 0$$

$$X_0^{(o)} = x_1 + x_3 = -2$$

$$X_1^{(o)} = x_1 + w^1 x_3 = 0$$

Calculate the twiddle factors for $N = 4$:

$$w_0 = 1;\ w_1 = e^{-i\pi/2} = -i;\ w_2 = e^{-i\pi} = -1;\ w_3 = e^{-i3\pi/2} = i$$

$$X_0 = X_0^{(e)} + X_0^{(o)} = 2$$

$$X_1 = X_1^{(e)} + w^1 X_1^{(o)} = 0$$

$$X_2 = X_0^{(e)} + w^2 X_1^{(o)} = 6$$

$$X_3 = X_1^{(e)} + w^3 X_1^{(o)} = 0$$

Each DFT coefficient contains information about the periodicity of the data set: the zeroth one is the sum (average signal); the first one measures the strength of the period one component (in this case, none), the second one is the strength of the period two component (in this case, the only frequency present), and the third one mirrors the period one (since there cannot be a period three signal measured in four points.)

This calculation for a small data set illustrates how the FFT algorithm reduces the calculation of the DFT coefficients to reassembling the values of the inputs x_k with the bit-reversed indices with appropriate twiddle factors. This is illustrated in {numref}`fig-fft-butterfly`, known as the "FFT butterfly" for its visual appearance. The figure demonstrates how the original data points on the left, if arranged in the bit-reversed order are scrambled up by the even/odd divisions, and how they end up in the normal order on the right hand side.

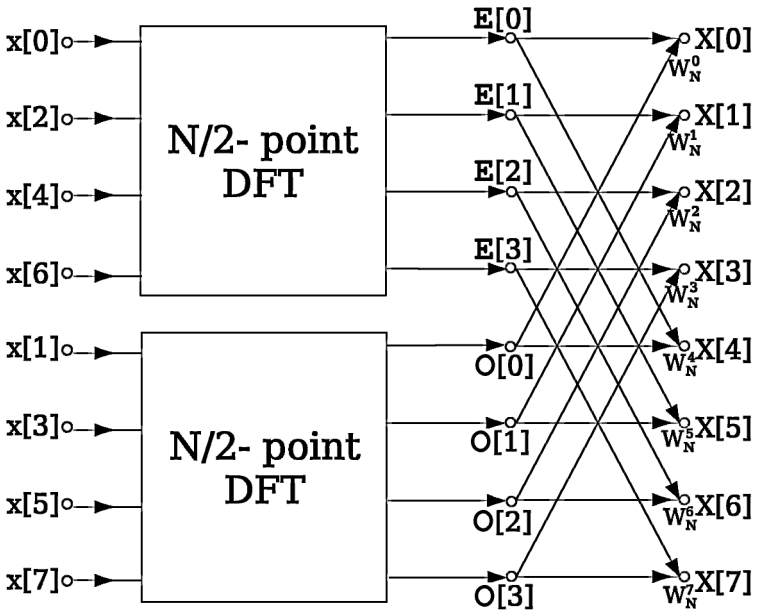


Figure 15.5: Schematic of the FFT butterfly for 8 input points https://commons.wikimedia.org/wiki/File:DIT-FFT-butterfly.png

The result of the FFT calculation is exactly the same as the direct DFT, but the FFT takes fewer arithmetic operations to perform. As mentioned above, computing the DFT directly requires $O(N^2)$ operations (the notation means a scalar multiple of N^2). The FFT starts by rearranging the data using bit-reversal (which takes only a small number of calculations). The key to efficiency is recursive reassembly of the DFT, which happens $\log_2(N)$ times, one for each split (assuming that N is a power of two, which allows for a clean division into singletons.) This calculation, as shown in equation, requires only two operations (an addition and a multiplication by the pre-computed twiddle factor) for each of the N DFT coefficients. Therefore, the total number of operations for FFT is $O(N \log_2(N))$ instead of $O(N^2)$ for DFT.

This results is a huge gain in efficiency for large data sets, for example, for a million data points $\log_2(10^6) \approx 20$, the number of operations is reduced by a factor of 50,000.

16 Linearization of ODEs

16.1 Introduction

Previously we learned to analyze and solve linear ODEs with two variables. The main idea was that the type of dynamics in a multivariable ODE is completely determined by the eigenvalues of the defining matrix, with the eigenvectors and initial conditions providing the relative weights of the variables. We will use these ideas to analyze nonlinear models, for which, typically, no analytical solution exists. Instead, we will focus on phase plane analysis to qualitatively describe the types of solutions possible for a dynamical system.

The main idea of this chapter is linearization, or approximating a nonlinear dynamical system with a matrix near a particular fixed point. This enables the use of the tools of linear algebra to characterize the local dynamics of the system. Linearization is an essential tool of applied mathematics and is quite powerful, even though it is not exact, and only applies in a local, possibly small neighborhood of a fixed point. In the next chapter, the stability of fixed points and the direction of flow of solutions will be used to obtain a more comprehensive picture of dynamics in the phase plane.

In this chapter, we will see examples of nonlinear ODEs in biology in the modeling section, and discuss the origin of product terms. In the analytical section, we will learn to linearize differential equations with more than one variable, and to analyze the dynamics near fixed points for two-variable ODEs. In the computational section, we will use numerical methods to find the eigenvalues of the linearized systems, and use computational tools for drawing the phase plane portraits near the fixed points. We will then use these techniques to analyze the dynamics of infections in a SIR epidemiology model.

16.2 Modeling: product terms in nonlinear differential equations

Nonlinear ODE is not a particularly descriptive term: it includes all equations that contain terms other than a constant or a constant times the dependent variable, e.g. x^2, y^3, ae^{x-y}. One common type of nonlinearity is a product term axy (x and y are dependent variables, and a is a constant). This term arises in diverse biological models, as illustrated below. The significance of the product terms is that they quantify the effect of **independent encounters** between the two entities modeled by the dependent variables, influence their rates of change. Thus, a

product of the two variables captures this dependence, which grows proportional to the size of both variables, and is zero if either variable vanishes.

16.2.1 ecological competition

Suppose two species are competing for the same resources, e.g. rabbits and sheep want to graze in the same fields. Let their individual growth rates be governed by logistic equations, with certain carrying capacities K_r and K_s and rates of growth a_r and a_s. In addition, the two species influence each others' rates of growth when their respective populations grow, by using up resources that both populations need to survive. Thus, the effect of competition can be represented by a term proportional to the **product of the two populations**. The effect of competition on the rabbits is represented by c_r and that on the sheep c_s, and the two parameters need not be the same, e.g. because the sheep are less sensitive to competition from rabbits than vice versa. The two variable ODE model can then be written as follows, with S representing sheep and R rabbits:

$$\begin{aligned} \dot{S} = &\quad r_1 S(K_1 - S) - c_s SR \\ \dot{R} = &\quad r_2 R(K_2 - R) - c_r SR \end{aligned}$$

These equations are nonlinear due to the squared terms and the SR terms. We cannot apply the powerful methods of linear algebra and find the eigenvalues of the system, since the equations cannot be written in matrix-vector form. In the analytical section, we will learn to use the method of linearization around a fixed point to determine behavior locally near equilibrium points.

16.2.2 chemical reactions with two molecules

Many biochemical reactions depend on **encounters between two molecules**, e.g. an enzyme and a substrate. As we would expect, this process is represented by a product terms in models of chemical kinetics. Take, for instance, a reversible bimolecular reaction:

$$A + B \underset{k_{-1}}{\overset{k_1}{\rightleftarrows}} C$$

The chemical reaction rates k_1 and k_{-1} are related to the speed of the forward (association) and backward (dissociation) reactions, but the two processes are fundamentally different, as in the SIS model above. The forward reaction depends on molecule A encountering molecule B, and thus its rate is proportional to the product of their concentrations. The dissociation reaction rate depends linearly on the concentration of C. Thus, the differential equations describing the rates of change of the concentrations of the three molecules are:

$$\begin{aligned}\dot{A} &= -k_1 AB + k_{-1} C \\ \dot{B} &= -k_1 AB + k_{-1} C \\ \dot{C} &= k_1 AB - k_{-1} C\end{aligned}$$

The equation are redundant, as the rates of change of A and B are equal. Therefore, if A_0 and B_0 are the initial concentrations of the two molecules, the concentration of $B(t)$ at some further time is given by $B(t) = A(t) - A_0 + B_0$ (adjusting for the difference in initial concentrations).

Further, there is a conserved quantity found by adding either $\dot{A} + \dot{C} = 0$ or $\dot{B} + \dot{C} = 0$. Thus, the sum of either the number of molecules A and C or of B and C is constant in the course of the reaction. Let C_0 be the initial concentration of C. Since the sum of two concentrations is constant, $C(t) + A(t) = C_0 + A_0 \Rightarrow C(t) = C_0 + A_0 - A(t)$. Note that the total number of molecules $A + B + C$ is not conserved, since the reactions change the number of molecules. The two redundancies make it possible to reduce the three equations to one:

$$A' = -k_1 A(A - A_0 + B_0) + k_{-1} A(C_0 + A_0 - A)$$

This equation can be examined using standard stability analysis for one-dimensional ODEs.

16.3 Analytical: linearization in multiple dimensions

In this section we learn to analyze the stability of fixed points in multiple dimensions. The concepts of fixed points and stability are the same as for one-dimensional dynamical systems. A fixed point is a point in the phase space (line, plane, etc.) at which all time derivatives are zero, i.e. there is no flow. The flow near a fixed point determines its stability: if all solutions in a neighborhood of a fixed point approach it for all time, the the fixed point is stable, otherwise it is not. We will now learn the mathematical to tools to do this analysis.

16.3.1 finding fixed points of nonlinear ODEs

A fixed point for a dynamical system of more than one variable is a set of values of all the variables, at which the dynamical system has no changes in any direction. More precisely,

Definition

For a dynamical system of N variables $\{x_i(t)\}$ defined by N first-order ordinary differential equations of the form $\dot{x}_i = f_i(x_1, x_2, ..., x_N)$, a *fixed point* is a vector of values of all N variables, $\vec{x}^* = (x_1^*, x_2^*, ..., x_N^*)$ for which every time derivative is zero, that is, $ 0 = f_i(x_1^, x_2^, ..., x_N^*)$ for all $i = 1, ..., N$.

It follows from the definition that to find a fixed point in multiple dimensions, we must solve several nonlinear algebraic equations simultaneously. This is not always easy, nor is it necessarily possible to do by hand. One standard approach is to consider the equations separately, and then the points where they all agree are the fixed points.

In two dimensions, this process is easy to visualize in terms of flow in the phase plane. Let us call the two dependent variables x and y, with differential equations $\dot{x} = f_1(x, y)$ and $\dot{y} = f_2(x, y)$. The two equations for the fixed points are:

$$\begin{aligned} 0 = \quad & f_1(x, y) \\ 0 = \quad & f_2(x, y) \end{aligned}$$

The first equation specificies the condition that $dx/dt = 0$, while the second corresponds to $dy/dt = 0$. In terms of the geometry of the phase plane with x on the horizontal axis and y on the vertical, it means that at any point (x, y) that satisfies the first equation, the flow is strictly vertical, with no horizontal component, while for a point that satisfies the second equation, the flow is strictly horizontal, with no vertical component. This leads us to a useful new concept in the phase plane:

Definition

For a two-dimensional autonomous ODE defined by $\dot{x} = f_1(x, y)$ and $\dot{y} = f_2(x, y)$, the set of points which satisfy $0 = f_1(x, y)$ is called the *x-nullcline*, and the set of points which satisfy $0 = f_2(x, y)$ is called the *y-nullcline*. The direction of flow at any point on the x-nullcline is strictly vertical, while on the y-nullcline it is strictly horizontal.

This idea gives us a graphical tool for visualizing the flow in the phase plane. The sets of points that satisfy the equations $0 = f_1(x, y)$ and $0 = f_2(x, y)$ may be found by solving the equation for y, and then plotting the results as curves in the phase plane. In some cases, the equation may not be solvable analytically (e.g. a transcendental equation) but one can still find the nullcline numerically and to plot its graph. Once all the nullclines are graphed, one may find the fixed points at the intersections of the x- and y-nullclines.

Example: rabbits and sheep

The fixed points of the model introduced in the modeling section, equation **??** must satisfy the conditions $\dot{s} = \dot{r} = 0$. Let us set the following constants for our example $K_1 = 2$, $K_r 2 = 3$, $r_1 = r_2 = 1$, $c_s = 1$, $c_r = 2$. Then the two equations that define the fixed points are:

$$\begin{aligned} 0 = \quad & s(2 - s) - sr \\ 0 = \quad & r(3 - r) - 2sr \end{aligned}$$

Let us find the nullclines of this dynamical system. The first equation is satisfied under two conditions: either $s = 0$ or $r = 2 - s$; these are the two s-nullclines. The second equation is satisfied under two conditions as well: either $r = 0$ or $2s = 3 - r$; these are the two r-nullclines.
We then look for the intersections of the nullclines to determine the fixed points. First, $s = 0$ and $r = 0$ intersect at the origin, so $s = r = 0$ is a fixed point. Second, $s = 0$ intersects the line $2s = 3 - r$ at $s = 0; r = 3$, so this is a second fixed point. Third, $r = 0$ intersects the line $r = 2 - s$ at $s = 2; r = 0$, and we have the third fixed point, Finally, the two lines $r = 2 - s$ and $2s = 3 - r$ intersect at $s = 1; r = 1$, which may be found either by sibstitution of $r = 2 - s$ into $2s = 3 - r$ or by graphing the two lines. We have found four fixed points, and because all the nullclines have straight line graphs, they can only intersect once, therefore this is the complete list of fixed points.
Let us interpret these equilibria. The one at the origin corresponds to the situation when there are no sheep or rabbits, so it is the dual extinction equilibrium. The two fixed points on the axes ($s = 0; r = 3$ and $s = 2, r = 0$) correspond to one species surviving and reaching its carrying capacity, with the other one dying off. The final fixed point $s = 1, r = 1$ is called a *coexistence equilibrium*, in which both species survive.

16.3.2 linear stability analysis of fixed points

We now turn to the question of stability of fixed points. As mentioned above, the definition of stability in multiple dimensions is identical to that is one dimension.

i Definition

A fixed point $\vec{x}^*$ of a dynamical system is *stable* if there exists a neighborhood of the fixed point (open set containing $\vec{x}^*$) such that any trajectory with initial value in that neighborhood approaches the fixed point for all time.

To analyze the behavior of the ODE in the vicinity of the fixed point, we use the power of partial deriviatives, which are defined as follows:

i Definition

For a multivariable function, e.g. $f(x, y)$ a *partial derivative* $\frac{\partial f}{\partial x}$ is the derivative of the function calculated while treating the other variables (e.g. y) constant. The partial derivative can then be evaluated at any point (x_0, y_0) to obtain the rate of change of the function in the x direction at that point.

Example: For $f(x, y) = 4x^3y - 5y^2$ the partial derivatives are:

- $\frac{\partial f}{\partial x} = 12x^2y$

- $\frac{\partial f}{\partial y} = 4x^3 - 10y$
- Evaluated at the point $(x, y) = (1, 2)$ the partials are $\frac{\partial f}{\partial x}(1, 2) = 24$ and $\frac{\partial f}{\partial y}(1, 2) = -16$

The partial derivatives of a nonlinear function evaluated at a point allow us to approximate the behavior of the function using only linear terms. This is analogous to the linear Taylor approximation of a function of one variable, which approximates a nonlinear function with its tangent line. Specifically, for a function $f(x, y)$ which is zero at a particular point (x^*, y^*) - that is it, it's a fixed point - the function can be approximated like this:

$$f(x^* + u, y^* + v) = u\frac{\partial f}{\partial x} + v\frac{\partial f}{\partial y} + O((u, v)^2)$$

The new variables u and v represent the deviations from the fixed point in the x and y direction, respectively and the $O((u, v)^2)$ term represents the higher-order terms that are neglected by using only the linear terms. Since in the vicinity of the fixed point u and v are very small, raising them to higher powers produces very small numbers that can be neglected (although sometimes they must be considered, when linear stability analysis is inconclusive).

Note that these partial derivatives are the directions of flow **at a particular fixed point**, thus they have to calculated separately for each fixed point (x^*, y^*). This produces a linear approximation in terms of the small increments u and v, and the partial derivatives play the role of constants, the local rates of change in x and y directions.

Let us focus on two-dimensional systems, for specificity. We can now approximate the functions $f_1(x, y)$ and $f_2(x, y)$ in the vicinity of a fixed point (x^*, y^*) using their partial derivatives as above. We can express the two variables near the fixed point as $x = x^* + u$ and $y = y^* + v$ and therefore obtain a local linear approximation for the system of two ODEs:

$$\begin{aligned} \dot{u} = &\quad u\frac{\partial f_1}{\partial x}(x^*, y^*) + v\frac{\partial f_1}{\partial y}(x^*, y^*) \\ \dot{v} = &\quad u\frac{\partial f_2}{\partial x}(x^*, y^*) + v\frac{\partial f_2}{\partial y}(x^*, y^*) \end{aligned}$$

What this achieves is an approximation of nonlinear differential equations by a set of linear differential equations, with the dependent variables expressed in terms of deviations from the fixed point. We now have a linear dynamical system, which can be written in matrix form:

$$\begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = J \begin{pmatrix} u \\ v \end{pmatrix}$$

The matrix J is made up of partial derivatives of the two functions f_1 and f_2 evaluated at some particular point (x, y), and it has a special name:

Definition

Let $(f_1(x,y), f_2(x,y)) : \Re^2 \to \Re^2$ (a function that takes in a two dimensional vector, and returns a two dimensional vector) be differentiable with to both variables x and y. Then its *Jacobian* matrix is defined as the matrix of first partial derivatives as follows:

$$J(x,y) = \begin{pmatrix} \frac{\partial f_1}{\partial x}\big|_{(x,y)} & \frac{\partial f_1}{\partial y}\big|_{(x,y)} \\ \frac{\partial f_2}{\partial x}\big|_{(x,y)} & \frac{\partial f_2}{\partial y}\big|_{(x,y)} \end{pmatrix}$$

The Jacobian matrix, evaluated at a fixed point, defines a linear dynamical system which approximates the original nonlinear dynamical system, in the vicinity of the fixed point. Thus, we can now use the tools from linear dynamical systems to determine the stability of a given fixed point. Much as the stability of a fixed point of a one-dimensional ODE was determined by the value of the derivative of the defining function at the fixed point, in higher dimensions the stability is determined by the eigenvalues of the Jacobian matrix at the fixed point.

To summarize, here is the outline for the steps for finding and analyzing the stability of fixed points of two-dimensional dynamical systems:

Outline for analyzing fixed points of 2-variable ODEs

- find the fixed points, as two dimensional vectors (x^*, y^*)
- find the expression for the Jacobian matrix of the ODE $J(x,y)$
- plug in the values of the fixed points to find the local linearizations around each fixed point $J(x^*, y^*)$
- find the eigenvalues of each Jacobian matrix and classify the local dynamics

Example: rabbits and sheep, continued

Let us write down the linear approximation (first terms of the Taylor series) for the two functions defining the differential equation in the ecological competition model from equation **??**, with specific parameters that we chose above.

$$\begin{aligned} \dot{u} = &\quad (2-2s-r)u + -sv \\ \dot{v} = &\quad (3-r-2s)v - 2ru \end{aligned}$$

You see that we have turned the nonlinear ODE into a linear system with variables u and v, defined by the Jacobian matrix evaluated at a point (r,s):

$$\begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} = \begin{pmatrix} 2-2s-r & -s \\ -2r & 3-2r-2s \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix}$$

In practice, one can typically proceed directly to calculating the Jacobian by finding the partial derivatives and putting them in the right places. Let me emphasize that the linearization only describes the behavior of the system near a particular fixed point. For each fixed point, the linear equation is different. In order to analyze stability of each of the fixed points, we need to plug in the four values of the fixed points individually:

- $r = 0, s = 0$. The matrix is

$$\begin{pmatrix} 2 & 0 \\ 0 & 3 \end{pmatrix}$$

and clearly has eigenvalues $\lambda = 2, 3$ with the corresponding eigenvectors on the two axes (1,0) and (0,1). This is an unstable node, with exponential growth away from it.

- $r = 0, s = 2$. The matrix is

$$\begin{pmatrix} -2 & -2 \\ 0 & -1 \end{pmatrix}$$

with eigenvalues $\lambda = -1, -2$ with the corresponding eigenvectors (-2,1) and (1,0). This is a stable node (exponential decay toward the fixed point).

- $r = 3, s = 0$. The matrix is

$$\begin{pmatrix} -1 & 0 \\ -6 & -3 \end{pmatrix}$$

with eigenvalues $\lambda = -1, -6$ with the corresponding eigenvectors (-1,3) and (0,1). This is a stable node (exponential decay toward the fixed point).

- $r = 1, s = 1$. The matrix is

$$\begin{pmatrix} -1 & -1 \\ -2 & -1 \end{pmatrix}$$

with eigenvalues $\lambda = -1 \pm \sqrt{2}$ with the eigenvectors $(1, -\sqrt{2})$ for the positive λ and $(1, \sqrt{2})$ for the negative one. This is a saddle point, with one stable direction and one unstable.

The linear stability analysis we employed yielded only local information around each individual fixed point. We now know that starting with a very small population of rabbits and sheep, neither population will approach extinction, and that starting close to the carrying capacity

for one population, and close to zero for the other, the solution will approach those two fixed points. The saddle point at the coexistence equilibrium appears more complex, but for practical purposes it is an unstable fixed point, and almost any solution near the coexistence equilibrium (but not exactly at $s = 1, r = 1$) will deviate away from it. However, this analysis gives no information for what happens to the solutions once they are away from the fixed points. In the next section we will demonstrate how to describe the global flow in the phase plane.

16.4 Computational analysis of Jacobian matrices

In practice scientists usually employ computational tools to find the eigenvalues of Jacobian matrices of dynamical systems. The procedure is identical for a two-dimensional or an N-dimensional differential equation, with the important difference that it is impractical to diagonalize matrices larger than 2 by 2 by hand. One can use any number of computational algorithms for finding eigenvalues, e.g. QR method [**?**]. We will leave out the discussion of these algorithms, instead rely on the reader having access to an implementation of such algorithms, e.g in Python.

Outline for classification of fixed points

Suppose that we have a two variable ODE and we have calculated its Jacobian matrix in general form.

- define the Jacobian function $Jac(x, y)$ which takes in two scalars x and y and returns the Jacobian matrix at that point
- obtain a list of N fixed points in two arrays x and y
- for loop through all N fixed points
  - obtain the Jacobian by calling $Jac(x[i], y[i])$
  - calculated its eigenvalues $v[0]$ and $v[1]$ (by calling an eigenvalue function)
  - if $Re(v[0]) > 0$ and $Re(v[1]) > 0$
    - if $Im(v[0]) == 0$
      - print 'unstable node'
    - else
      - print 'unstable spiral'
  - if $Re(v[0]) < 0$ and $Re(v[1]) < 0$
    - if $Im(v[0])) == 0$
      - print 'stable node'
    - else
      - print 'stable spiral'

- if $Re(v[0]) * Re(v[1]) < 0$
  - print 'saddle point'
- if $Re(v[0]) * Re(v[1])) == 0$
  - if $Im(v[0]) \neq 0$
    - print 'center point'
  - else
    - print 'line of fixed points'

16.5 Application: SIR model

Mathematical epidemiology models the dynamics of infectious disease in human (or other) populations. In Chapter 3, we saw the simplest models divide the population into two groups: those not infected (a.k.a. susceptible S) and infected (I). Here we will add a third category: recovered individuals (a group which may include those who died from the disease). There are two dynamic processes: the susceptible become infected at some rate upon encountering an infected individual, and the infected can recover. These two processes are different: the rate of infection fundamentally depends on **encounters between susceptible and infected individuals**, and thus is modeled as a product of S and I, while the rate of recovery depends only on the infected individuals, and is thus is represented by a linear term proportional to I.

$$S + I \xrightarrow{\beta} 2I$$
$$I \xrightarrow{\gamma} R$$

This is known as the Kermack-McKendrick model [**?**], or SIR model of epidemics, and it can be described by the following system of equations:

$$\begin{aligned} \dot{S} &= -\beta IS \\ \dot{I} &= -\gamma I + \beta IS \\ \dot{R} &= \gamma I \end{aligned}$$

Notice that, just like the simpler SIS model in Chapter 3, this system has a conserved quantity, the total number of individuals N. This can be shown by adding all three equations together, to see that $\dot{S} + \dot{I} + \dot{R} = 0$, and therefore the sum of the three categories of individuals N is constant. This makes intuitive sense because all the dynamics in the model consists of transferring individuals from one category to another, but we are assuming no births or deaths, outside of those we rolled into the recovered category.

Since there is a conserved quantity, the three variables can be reduced to two, e.g. by expressing $R = N - I - S$. Since R has no effect on the dynamics of S or I, let us leave it out, and analyze the two-variable ODE:

$$\begin{aligned} \dot{S} &= -\beta IS \\ \dot{I} &= -\gamma I + \beta IS \end{aligned}$$

First, let us find its fixed points. There are two nullclines for S: $I = 0$ and $S = 0$. Similarly, there are two nullclines for I: $I = 0$ and $S = \beta/\gamma$. Notice that the two directions have one nullcline in common $I = 0$. This means that the line $I = 0$ (the S-axis) is a line of fixed points! As strange as it may seem, it should make sense: when there are no infected individuals, there can be no epidemic. There are no other fixed points, since if $I \neq 0$, the fixed points would have to lie on the intersection of $S = 0$ and $S = \beta/\gamma$, which does not exist as long as the parameters are positive numbers.

Now let's find the Jacobian of the system:

$$J(S, I) = \begin{pmatrix} -\beta I & -\beta S \\ \beta I & -\gamma + \beta S \end{pmatrix}$$

On the line of fixed points, $I = 0$, the Jacobian becomes:

$$J = \begin{pmatrix} 0 & -\beta S \\ 0 & -\gamma + \beta S \end{pmatrix}$$

This matrix is obviously singular, and has a zero eigenvalue with eigenvector $(1, 0)$. This means that the flow on the S-axis (provided it is horizontal) is strictly vertical. The other eigenvalue is $-\gamma + \beta S$, with eigenvector $(0, 1)$. This eigenvalue changes with S, or position on the S-axis. For $S > \gamma/\beta$, the eigenvalue is positive, while for $S < \gamma/\beta$, the eigenvalue is negative. Thus, we can predict stability of the infection-free equilibrium based on the size of S relative to the threshold γ/β.

We can now predict the local dynamics of an epidemic, starting with a small number of infected individuals I_0. Further, let us assume that initially there are no recovered individuals, therefore $N = I_0 + S_0 \approx S_0$. If $N\beta/\gamma > 1$, the number of infected individuals with increase, because the eigenvalue is positive and the flow takes the solution away from $I = 0$, and the infection will spread. If $N\beta/\gamma < 1$, the number of infected individuals will decrease, and the epidemic will die out. $N\beta/\gamma$, as in the SIS model, is called the disease reproductive ratio (and often denoted R_0, in what may be considered notational abuse, since it has nothing to do with the number of recovered individuals). Notice that in both situations, the infection will spread beyond the initially infected individuals. The difference between the two scenarios is that in the first one, the number of infected individuals at a given time increases, while in the second

one the number of infected at a given time decreases. However, in both cases, the number of people infected over the course of the epidemic is greater than the initial number I_0.

As noted before, the local stability analysis does not describe the behavior away from the line of fixed points, but we have drawn some conclusions about the global dynamics of epidemics, based on the features of the flow in the phase plane. The total number of individuals is fixed, and all the infected individuals eventually recover, with no-reinfection possible. Therefore, no epidemic can persist, unlike in the SIS model, in which reinfection was possible. In fact, all solution trajectories eventually approach the line of fixed points at $I = 0$. If the reproductive ratio $R_0 < 1$, the solution heads for $I = 0$ right away, but if $R_0 > 1$, the solution first travels away from $I = 0$, reaches its peak where it crosses the vertical nullcilne $S = \beta/\gamma$ (the direction of flow on that nullcline is strictly horizontal) and then turns toward $I = 0$, and eventually approaches an equilibrium of no infections, having burned itself out. The dynamics of the model for the two cases are shown in figure **??**. There are other interesting mathematical properties of the Kermack-McKendrick model, for a good summary of which see .

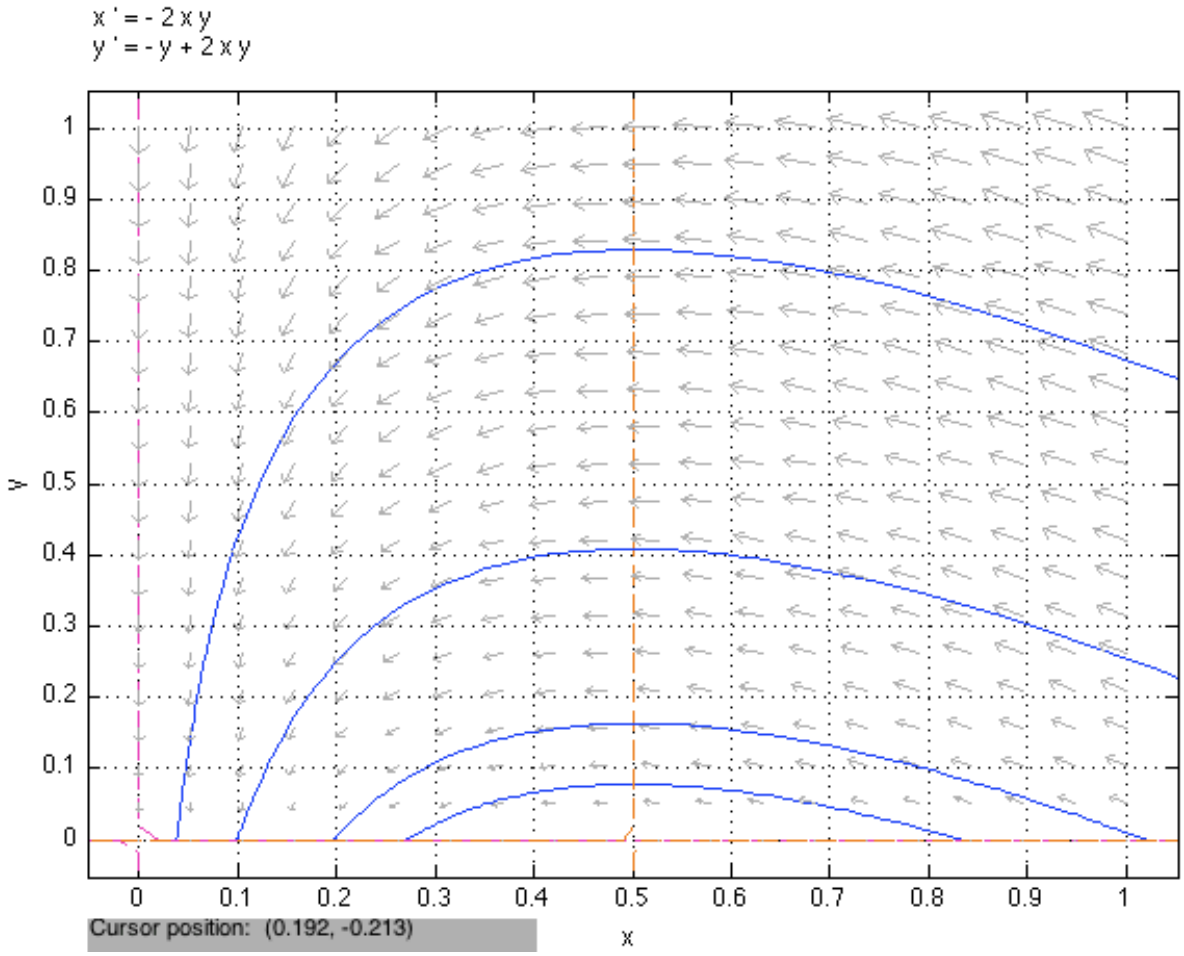


Figure 16.1: Dynamics in the phase plane of SIR models depends on the rep both have a line of fixed points on the x axis. Shown here is the phase plane with $R_0 = 2$, with infections that start to the right of the vertical nullcline spread before running out and approaching $y = 0$.

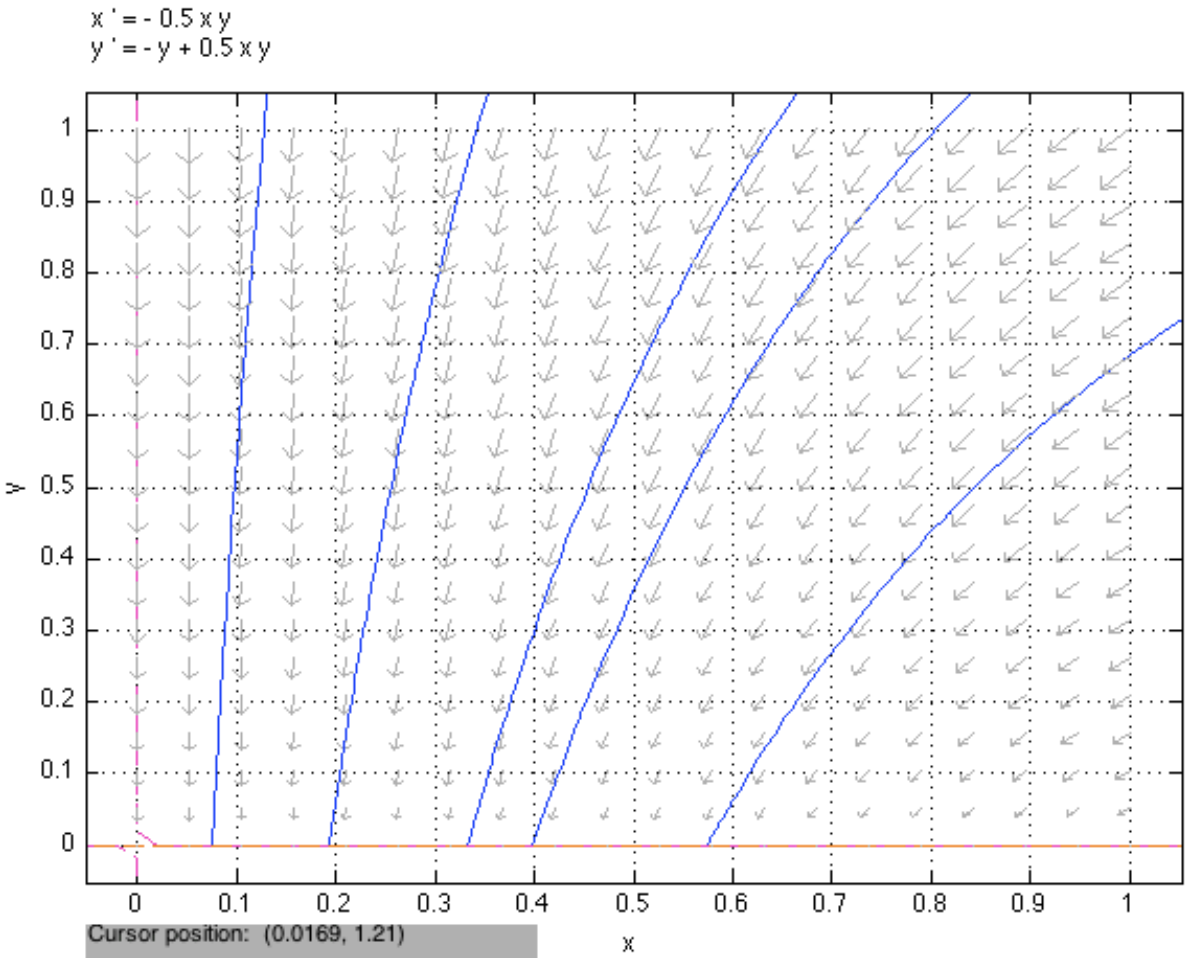


Figure 16.2: Dynamics in the phase plane of SIR models depends on the rep both have a line of fixed points on the x axis. Shown here is the phase plane with $R_0 = 0.5$ and infections with any initial number of infecteds immediately decrease to zero.

17 Nonlinear oscillations in biology

17.1 Introduction

In previous chapters we examined periodic solutions that arise in linear or conservative nonlinear systems. These solutions show up in the phase plane as closed trajectories, looping around a fixed point of the center classification; they are known as linear oscillations, since their time plots are oscillatory, and because the dynamics around the fixed point are correctly described by the linear approximation. In this chapter we will see a new kind of closed trajectories, known as nonlinear oscillations. These solution describe very different dynamics, which is found in many biological systems, characterized by cycles with excitatory and refractory phases, for example action potentials in neurons.

In the modeling section, we will see a simple prototype model which produces nonlinear oscillations, and then describe the qualitative properties of excitatory systems in biology. In the analytic section, we will describe some types bifurcations in the plane. In particular, we will describe the bifurcation which results in the birth of nonlinear oscillations. Finally, in the synthesis section we will analyze the classic neuroscience model of an action potential, called the Fitzhugh-Nagumo model.

17.2 Modeling: nonlinear oscillations

17.2.1 Van der Pol oscillator

Let us consider a harmonic oscillator model with a nonlinear damping term, known as the van der Pol oscillator (here k and μ are positive parameters):

$$m\ddot{x} = -kx - \mu(x^2 - 1)\dot{x}$$

This equation is similar to the damped harmonic oscillator equation we saw in the chapter 10, but in this model the damping coefficient, or the multiplier of velocity $\dot{x}$, is $\mu(x^2 - 1)$ and thus depends on the variable x. If $x > 1$, the coefficient of damping is positive (with a negative sign in front) and there are damped harmonic oscillations. However, if $x < 1$, the coefficient is negative, so instead of damping, there is amplification of oscillations. Let us write this as two first-order ODEs, and analyze its behavior is the phase plane.

$$x' = y$$
$$y' = -\frac{k}{m}x - \frac{\mu}{m}(x^2 - 1)y$$

There is only one fixed point, at $x = 0, y = 0$. It is unstable because of the amplification (reverse damping) effect when x is small. This can be properly shown by linearizing around the fixed point and finding the eigenvalues - we'll leave it as an exercise. However, since damping kicks in for $x > 1$ and is stronger for higher values of x, trajectories that start far away from the origin will flow toward it. The effect of the outward flow from the origin combined with flow toward the origin (from trajectories that start sufficiently far away) creates a new type of periodic trajectory, called a *limit cycle*.

17.2.2 oscillatory behavior in biology

The properties of linear oscillations do not correspond well to periodic behavior in living systems. Closed trajectories around center points can have any amplitude as determined by the initial conditions, and a frequency which is determined by the eigenvalues of the Jacobian, therefore independent of the amplitude of oscillations. In biological systems (and many other natural systems) periodic behavior typically has an optimal period as well as an optimal amplitude. For instance, the action potential of a neuron, as we will discuss in the Synthesis section, has a well-defined maximum and minimum membrane voltage, as well as a set period between peak voltages (neuron firing). Other periodic biological processes, such as oscillations in metabolism or circadian rhythms, share the same property of having well-defined frequency and amplitude. For a mathematical modeler this suggests constructing dynamical systems that have a *unique and attracting* periodic trajectory.

There are several common features of biological oscillatory models. One is the competitive interplay between an inhibiting and activating species, which are commonly seen in biochemical reactions or gene regulatory models. We will see such an example in the model of oscillations in glycolysis. The other is the so-called excitable systems, in which there is a threshold value, for stimulus to produce an oscillatory response. Such systems have oscillations which are divided into fast stages of excitation and slow stages of relaxation. We will see a prototypical model of this kind in the Synthesis section.

17.3 Limit cycles and flow-trapping regions

17.3.1 limit cycles

There is an important result in dynamical systems theory that states that the only possible attractors (sets of points which solutions approach in the long run) in two dimensions are

either fixed points (steady states) or closed orbits (oscillations). We saw linear oscillations in the harmonic oscillator model, and the oscillatory solutions in the Lotka-Volterra model similarly correspond to an infinite set of closed trajectories. Below we consider circumstances under which unique closed trajectories arise.

A limit cycle is an isolated closed trajectory, which makes it different from oscillations around a center. Limit cycles can be stable (with nearby trajectories spiraling toward it), unstable (with nearby trajectories spiraling farther away) and half-stable (inner trajectories and outer trajectories behaving differently). This phenomenon requires some nonlinearity in the equations, like the nonlinear damping above, and thus limit cycles are known as *nonlinear oscillations*. Limit cycles cannot be found by linearization (since it neglects the essential nonlinearity), so we need another method.

> **i** Theorem (Poincare - Bendixson)
>
> A region of the phase plane which contains no stable fixed points, and for which the direction of the flow everywhere on the boundary is inward, must contain a limit cycle.

Another important difference between linear and nonlinear oscillations is in *robustness* of the solutions to changes in parameters. Linear centers are fragile with respect to parameter changes, because in order for the eigenvalues to be purely imaginary, the trace of the matrix (sum of the diagonal elements) has to be exactly 0, and the smallest deviation will destroy the closed orbits, transforming them into a spiral. Limit cycles, in contrast, are generally robust to parameter changes, because a small change will not dramatically affect the direction of the flow into the region of Poincare-Bendixson Theorem.

This result allows us to find limit cycles around fixed points by checking if there is a region with flow oriented in the opposite direction from that near the fixed point. In practice, finding such a region may be difficult, so we demonstrate one common type of ODE which satisfied the conditions of the theorem.

17.3.2 Example: cubic nullclines

One common type of models in physics and biology which has limit cycle solutions are those with cubic terms in the governing ODEs. For instance, there is a class of models that look as follows:

$$x' = y - f(x)$$
$$y' = -x$$

This system has nullclines on the curves: $y = f(x)$ and $x = 0$ (y-axis). If the function $f(x)$ is a cubic function with a maximum and a minimum, e.g. $f(x) = x^3 - x$, then the phase plane will look like figure {numref}`fig-cub-null`.

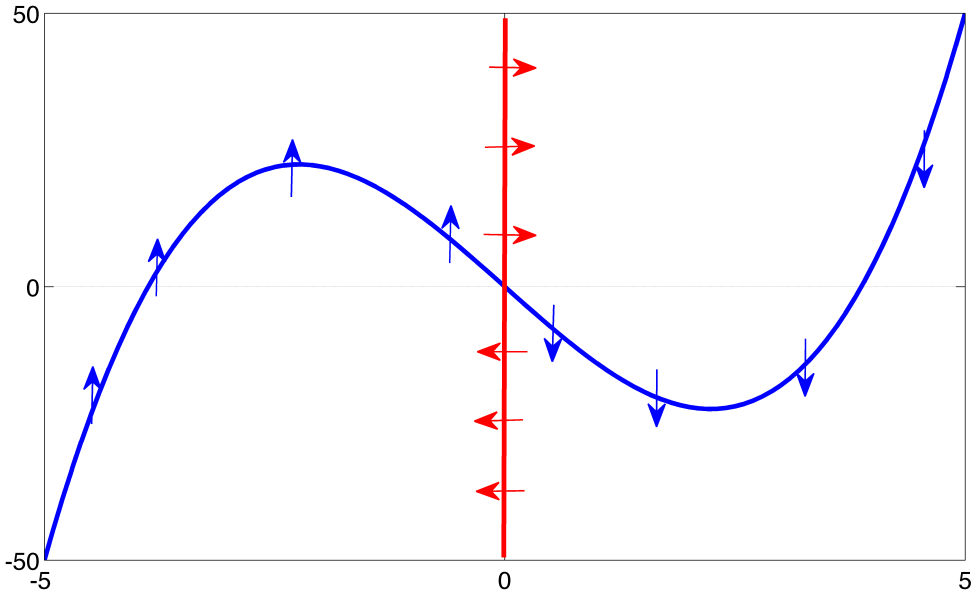


Figure 17.1: Cubic nullcline with a maximum and minimum generates a limit cycle

Let us analyze the geometry of the flow. First, there is only one fixed point, at the origin, and the flow may circulate around it. To show that a limit cycle exists, we must:

- Show that the fixed point is unstable
- Show that there is a region surrounding the fixed point, with inward flow on the boundary

First, we will show geometrically that whether the flow is attracted to or repelled by the fixed point depends on the slope of the function $f(x)$. To see this analytically, let's do the linearization procedure of this ODE. The Jacobian matrix of the ODEs is:

$$J(x,y) = \begin{pmatrix} -f'(x) & 1 \\ -1 & 0 \end{pmatrix}$$

To assess stability, we find the eigenvalues:

$$\lambda = \frac{-f'(x) \pm \sqrt{f'(x)^2 - 4}}{2}$$

The eigenvalues are real if $f'(x)^2 > 4$ and complex if $f'(x)^2 < 4$. If they are complex, the real part is $-f'(x)/2$, and so the stability is determined by the sign of $f'(x)$. If the eigenvalues are real, stability still depends on the sign of $f'(x)$, because $f'(x) > \sqrt{f'(x)^2 - 4}$ and thus

adding or subtracting the square root will not change the sign of the expression. Check for yourself the direction of the flow on the nullclines (e.g. for the y-nullcline, on which $\dot{y} = 0$, we need to consider when $\dot{x} > 0$ and when $\dot{x} < 0$). In figure {numref}`fig-pos-slope`, the horizontal flow coming off the y-axis misses the nullcline that would turn it around, while in figure {numref}`fig-neg-slope`, the horizontal flow runs right into the positive-sloped nullcline. Thus, when $f'(x) < 0$, the fixed point is stable (either a node or a spiral), and when $f'(x) > 0$, the fixed point is unstable (either a node or a spiral). As a corollary, when the two nullclines are perpendicular, the fixed point is a center.

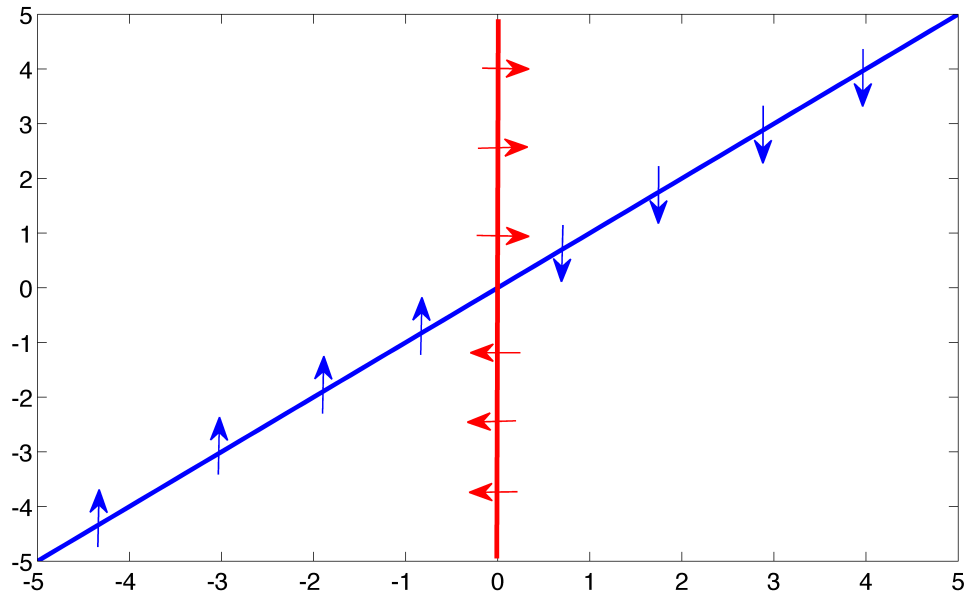


Figure 17.2: Slope of x-nullcline (in blue) determines the stability of the fixed point; the positive slope of the function $f(x)$ creates an unstable fixed point.

Now let us go back to the cubic nullcline in figure {numref}`fig-cub-null`. It should be clear that there is a trapping region of phase space, because at any point sufficiently far from the origin, the direction of flow is toward the origin. Intuitively, the directions of flow on the nullclines ensure that the flow heads down on the right hand side of the plane, and upward on the left-hand side of the plane; the flow is trapped by the cubic "arms" of the graph of $f(x)$. One can construct a flow trapping-region more rigorously, but I will leave this as a potential exercise for the reader. Therefore, as long as there is an unstable fixed point inside a flow-trapping region, which is determined by the slope of $f(x)$ at the intersection of the nullclines, the Poincare-Bendixson theorem ensures the existence of a limit cycle in that region. This result is used in many biological models.

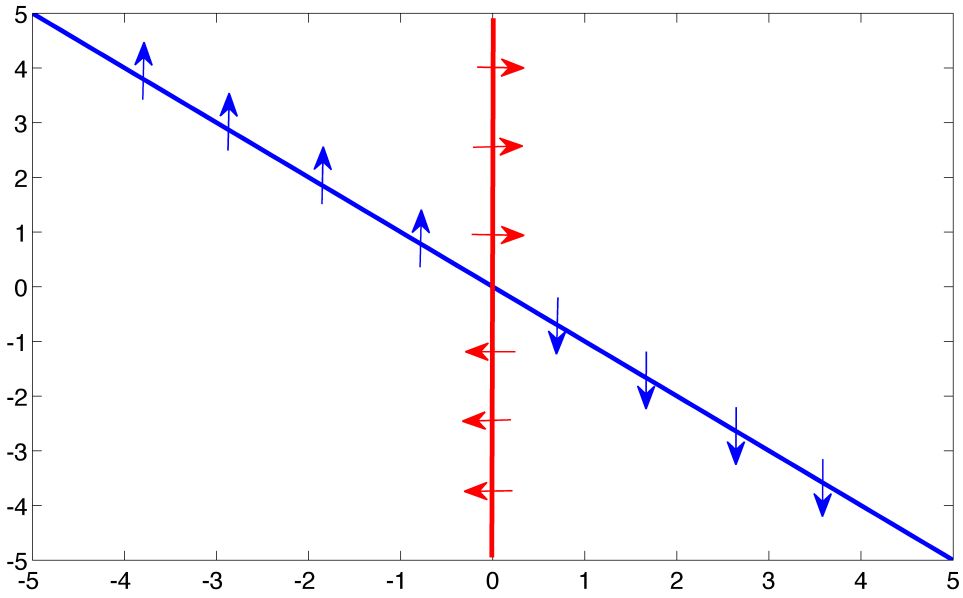


Figure 17.3: Slope of x-nullcline (in blue) determines the stability of the fixed point; the negative slope of the function $f(x)$ creates an stable fixed point.

17.3.3 Example: glycolytic oscillator

The metabolism of glucose to produce energy in the form of ATP (adenosine triphosphate) is a crucial process in most lifeforms. The process involves many stages, most of which involve phosphorylation, or attachment of a phosphate group, and each one is catalyzed by a specialized enzyme. One intermediate step involves the phosphorylation of Fructose 6-phosphate (Fru-6P) to produce Fructose 1,6-biphosphate (Fru-1,6bP), by an enzyme called phosphofructokinase (PFK). In many organisms this step is known to lead to periodic oscillations in the concentrations of the molecules involved.

The reaction involves several types of molecules: the two sugars, the enzyme, and a molecule of ATP and ADP:

$$Fru-6P+ATP+PFK \rightarrow Fru-1,6bP+ADP+PFK$$

One other key feature is that one product of this reaction, ADP, allosterically activates the enzyme PFK, therefore leading to more ADP and Fru-1,6bP production. We will model these processes by keeping track of two dynamic variables: concentration of Fru-6P (y) and concentration of ADP (x). The other species are closely related to these two, and the concentration of PFK is assumed to stay constant.

ADP is produced with a rate proportional to the concentration of Fru-6P, but this rate depends on the concentration of ADP. Because it is known that two molecules of ADP need to bind to activate PFK, this dependence is quadratic, by the principles of mass action. In addition, ADP is depleted at a rate proportional to its concentration by other cellular processes. The situation for Fru-6P is opposite, since it is depleted by this reaction. It is depleted at the rate which is a constant plus the square of ADP concentration. It is supplied with a constant rate from the upstream reaction, and thus independent of its own concentration. These assumptions were expressed in a simple, dimensionless model proposed by Sel'kov:

$$x' = -x + (a + x^2)y$$
$$y' = b - (a + x^2)y$$

Let us examine this model by our usual methods. First, let us find the nullclines: for x the nullcline is $y = x/(a + x^2)$, and for y the nullcline is $y = b/(a + x^2)$. These nullclines intersect at a single point $x = b$ and $y = b/(a + b^2)$. The Jacobian of the ODE is:

$$J(x, y) = \begin{pmatrix} -1 + 2xy & a + x^2 \\ -2xy & -(a + x^2) \end{pmatrix}$$

and at the fixed point, the matrix is

$$J(b, b/(a + b^2)) = \begin{pmatrix} -1 + 2b^2/(a + b^2) & a + b^2 \\ -2b^2/(a + b^2) & -(a + b^2) \end{pmatrix}$$

The Jacobian has determinant $\Delta = a + b^2$ and trace

$$\tau = -\frac{b^4 + (2a - 1)b^2 + (a + a^2)}{a + b^2}$$

Since the eigenvalue expression is this: $\lambda = (\tau \pm \sqrt{\tau^2 - 4\Delta})/2$, and the determinant is strictly positive, the stability of the fixed point depends solely on the sign of the trace. Doing a little algebra we can find the curve which divides the stable and the unstable region in the a-b parameter plane:

$$b^2 = \frac{1}{2}\left(1 - 2a \pm \sqrt{1 - 8a}\right)$$

The second step in finding nonlinear oscillations is searching for a flow-trapping region. I will leave out the details here, but one can construct the following region, in which the direction of flow on the boundary is everywhere inward. One boundary is the y axis from 0 to b/a. The second boundary is the line from $(0, b/a)$ to $(b, b/a)$. the third segment is the line with slope

-1, from the point $(b, b/a)$ to the x axis, and the final boundary is the x axis between (0,0) and the intersection with that line.

Now that we have a flow-trapping region, which is independent of parameters, the existence of limit cycle depends on the stability of the fixed point inside the region. We have shown that for some values of a and b, the fixed point is unstable, and thus Poincare-Bendixson predicts the existence of a limit cycle. Figure {numref}`fig-glyco-stable` shows the flow and a solution trajectory for the case where the fixed point is stable, and figure {numref}`fig-glyco-cycle` shows the phase portrait when the fixed point is unstable and a solution converging to a limit cycle.

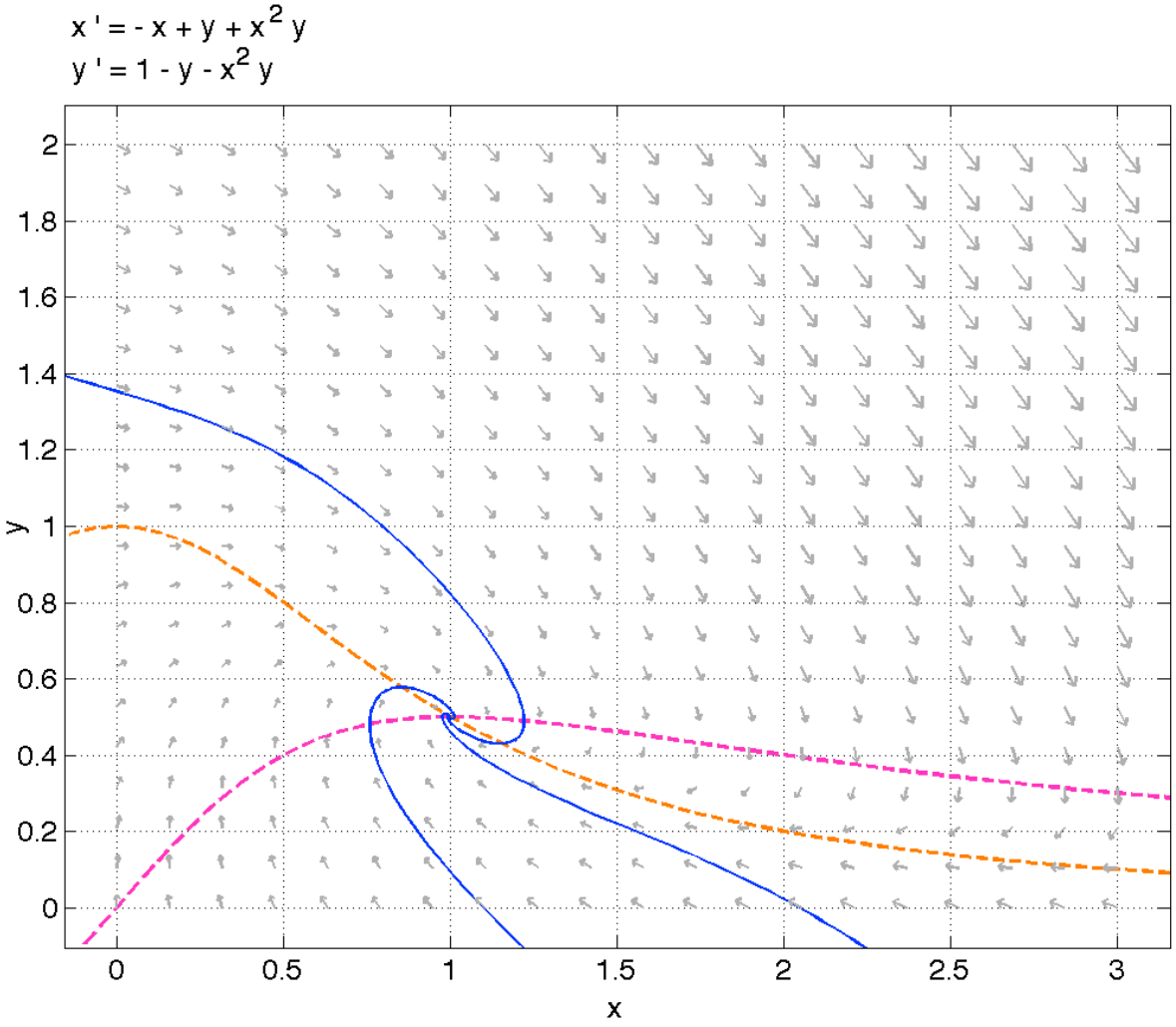


Figure 17.4: Phase portrait of the glycolysis model (nullclines for x and y shown in orange and pink): with a stable fixed point.

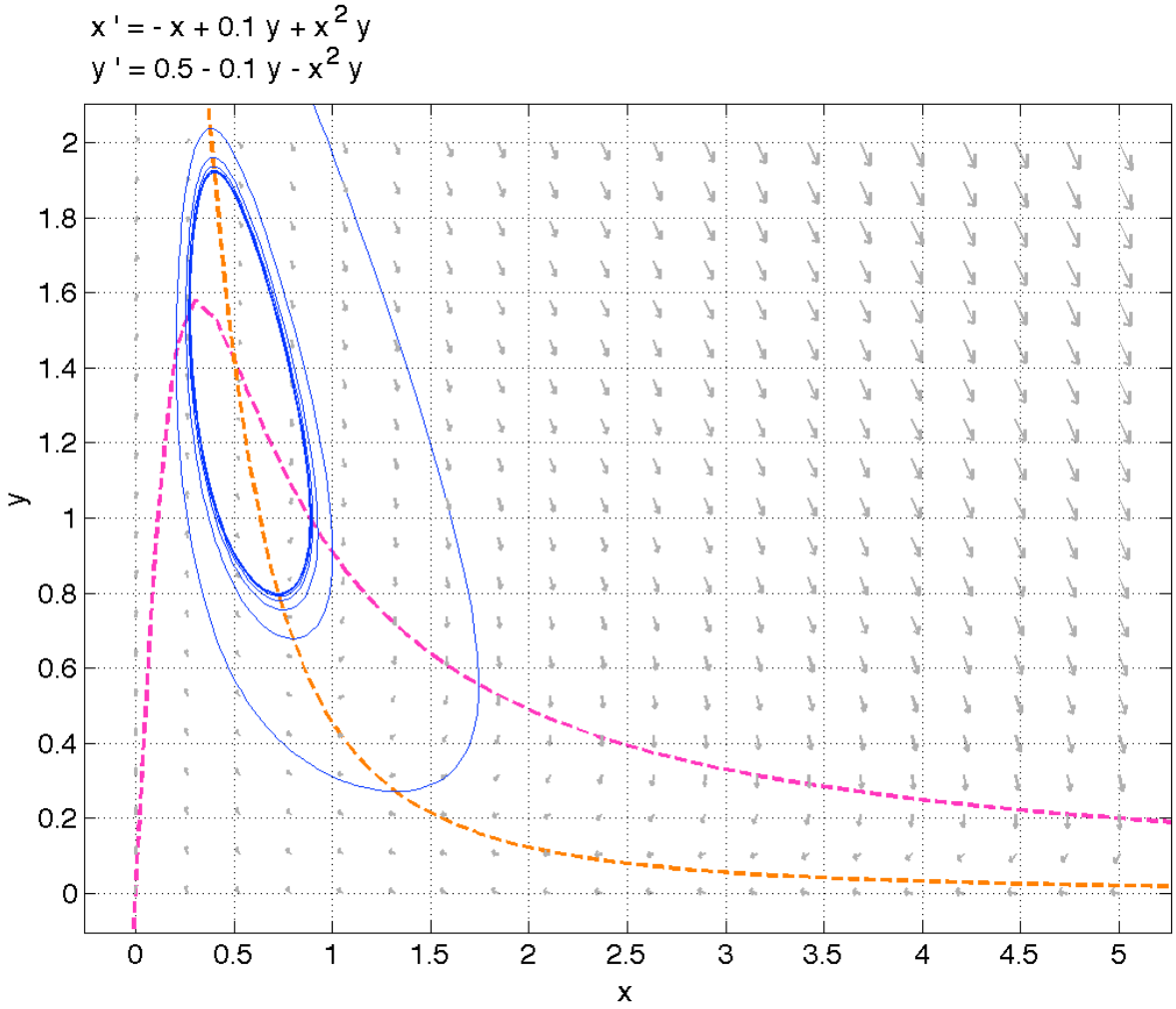


Figure 17.5: Phase portrait of the glycolysis model (nullclines for x and y shown in orange and pink): with an unstable fixed point and a solution trajectory converging to a limit cycle.

17.4 Fitzhugh-Nagumo model of neural excitation

Limit cycles are often used to model *excitable* systems, in which spikes of activity are followed by a period of relaxation. One such example is the action potential of a neuron, consisting of a quick spike and the depolarization of the potential built up across the membrane, followed by a slow return to the resting potential. A very simple, albeit largely unrealistic, model is one proposed independently by Fitzhugh and Nagumo. In this model, the meaning of the variables is approximate, with x standing in for the membrane potential, and y signifying the extent of the recovery via active pumping of ions, and $z(t)$ representing the external stimulus, such as an action potential from a connected neuron:

$$x' = c\left(y + x - \frac{x^3}{3} + z(t)\right)$$

$$y' = -\frac{x - a + by}{c}$$

To properly analyze the stability of the fixed point, find the Jacobian of the Fitzhugh-Nagumo model, which does not depend on the value of the inhomogenous term $z(t)$:

$$J(x, y) = \begin{pmatrix} c - x^2 & 1 \\ -1 & b/c \end{pmatrix}$$

The eigenvalues of the Jacobian are:

$$\lambda = \left(c - x^2 + b/c \pm \sqrt{(c - x^2 + b/c)^2 - 4(b - bx^2/c + 1)}\right)/2$$

As you can see, the analysis of stability by algebra is going to be painful, since we need to find the expressions for the fixed point and express it in terms of the parameters and the stimulus $z(t)$. Instead, we are going to use the qualitative analysis we developed in the analytic section to analyze the existence of limit cycles in the phase plane.

The nullclines of this system are similar to those in the generic cubic model analyzed above. The nullcline for $\dot{x}$ is given by $y = x^3/3 - x - z(t)$, which is a cubic with a max and a min, with an extra term added. The nullcline for $\dot{y}$ is given by $y = (a - x)/b$, a negative-slope line, as opposed to the vertical line in the model above, but the directions of the flow relative to the blue nullcline remain the same (check by analyzing the equation for $\dot{x}$).

Depending on the values of the parameters, and on the value of the external input z, the nullclines may intersect in different ways. If the cubic at the intersection has a positive slope, that, as we saw above, the fixed points will be stable, as can be seen seen in figure {numref}`fig-fn-stable`. If the line and cubic intersect in the negative-slope part of the cubic, then the fixed point is unstable, as illustrated in figure {numref}`fig-fn-unstable`.

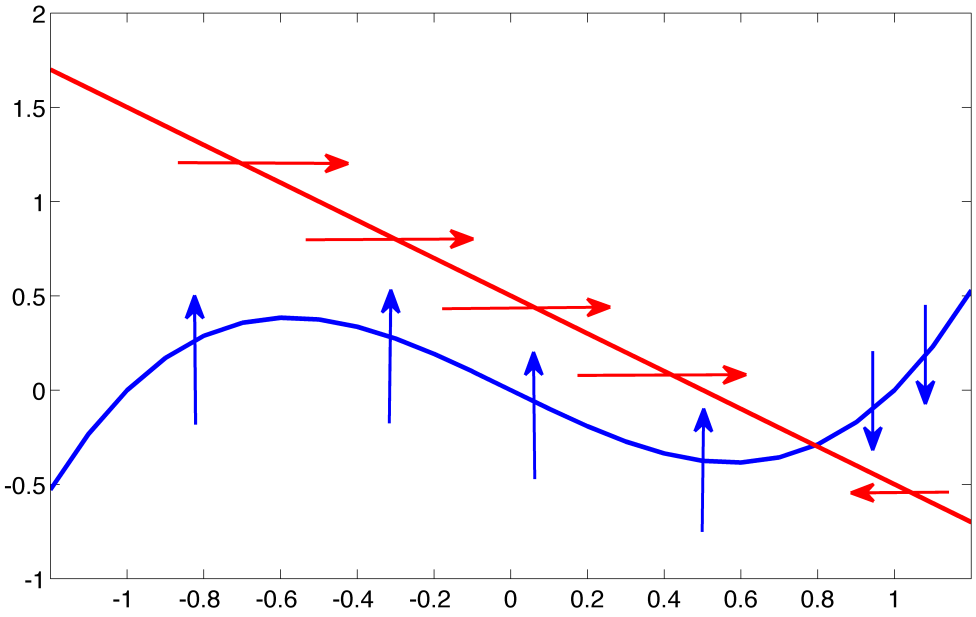


Figure 17.6: Stable fixed point in the Fitzhugh-Nagumo model.

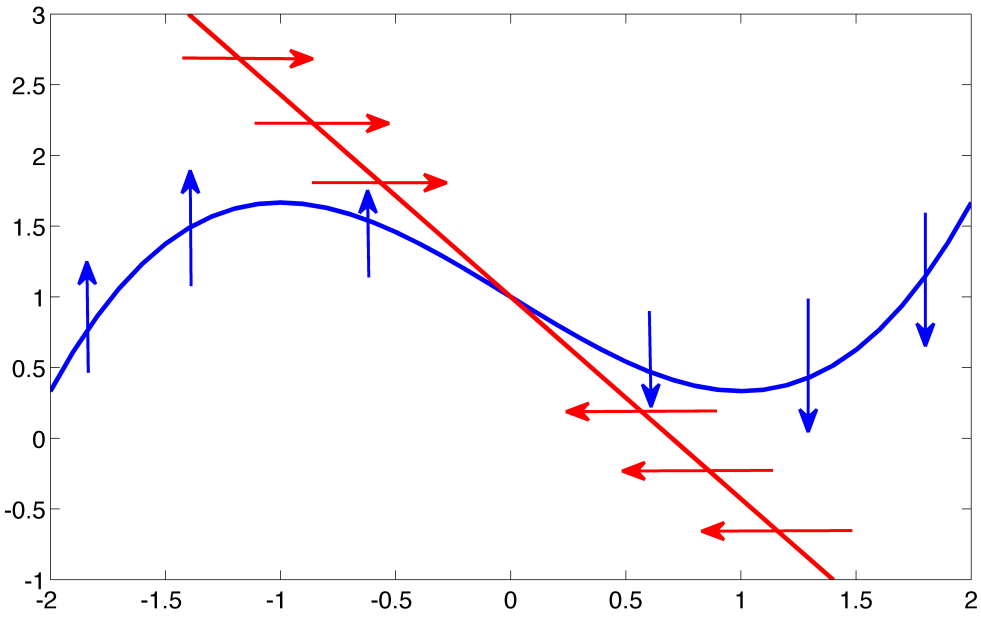


Figure 17.7: Unstable fixed point in the Fitzhugh-Nagumo model.

As we saw above, if the fixed point is unstable and the shape of cubic prevents the flow from leaving a larger region, then there must exist a limit cycle. On the other hand, if the fixed point is stable, then there is no limit cycle. So the Fitzhugh-Nagumo provides a simple model for the transition between a resting state with constant voltage (stable fixed point) to the excited state where periodic action potentials are generated (limit cycle). The readers can convince themselves that the addition of the external signal $z(t)$ can lead to this transition.

References