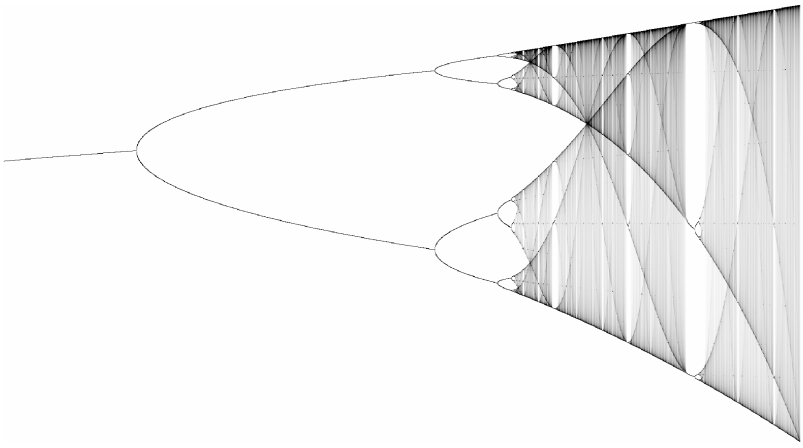
Scientific Computing

Jeffrey R. Chasnov



HARVEY MUDD COLLEGE

HKUST

The Hong Kong University of Science and Technology
Department of Mathematics
Clear Water Bay, Kowloon
Hong Kong

Preface

What follows are my lecture notes for Math 164: *Scientific Computing*, taught at Harvey Mudd College during Spring 2013 while I was on a one semester sabbatical leave from the Hong Kong University of Science & Technology. These lecture notes are based on two courses previously taught by me at HKUST: Introduction to Scientific Computation and Introduction to Numerical Methods.

Math 164 at Harvey-Mudd is primarily for Math majors and supposes no previous knowledge of numerical analysis or methods. This course consists of both numerical methods and computational physics. The numerical methods content includes standard topics such as IEEE arithmetic, root finding, linear algebra, interpolation and least-squares, integration, differentiation, and differential equations. The physics content includes nonlinear dynamical systems with the pendulum as a model, and computational fluid dynamics with a focus on the steady two-dimensional flow past either a rectangle or a circle.

Course work is divided into three parts. In the first part, numerical methods are learned and MATLAB is used to solve various computational math problems. The second and third parts requires students to work on project assignments in dynamical systems and in computational fluid dynamics. In the first project, students are given the freedom to choose a dynamical system to study numerically and to write a paper detailing their work. In the second project, students compute the steady flow past either a rectangle or a circle. Here, students are given the option to focus on either the efficiency of the numerical methods employed or on the physics of the flow field solutions. Written papers should be less than eight pages in length and composed using the LaTeX typesetting software.

All web surfers are welcome to download these notes at

http://www.math.ust.hk/~machas/scientific-computing.pdf

and to use these notes freely for teaching and learning. I welcome any comments, suggestions or corrections sent by email to jeffrey.chasnov@ust.hk.

Contents

Part I

Numerical methods

The first part of this course consists of a concise introduction to numerical methods. We begin by learning how numbers are represented in the computer using the IEEE standard, and how this can result in round-off errors in numerical computations. We will then learn some fundamental numerical methods and their associated MATLAB functions. The numerical methods included are those used for root finding, integration, solving differential equations, solving systems of equations, finite difference methods, and interpolation.

Chapter 1

IEEE arithmetic

1.1 Definitions

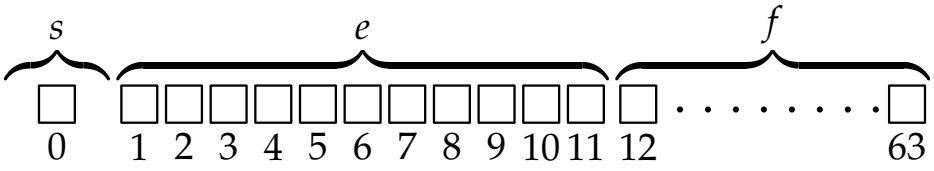
The real line is continuous, while computer numbers are discrete. We learn here how numbers are represented in the computer, and how this can lead to round-off errors. A number can be represented in the computer as a signed or unsigned integer, or as a real (floating point) number. We introduce the following definitions:

Bit	=	0 or 1	
Byte	=	8 bits	
Word	=	Reals:	4 bytes (single precision)
			8 bytes (double precision)
	=	Integers:	1, 2, 4, or 8 byte signed
			1, 2, 4, or 8 byte unsigned

Typically, scientific computing in MATLAB is in double precision using 8-byte real numbers. Single precision may be used infrequently in large problems to conserve memory. Integers may also be used infrequently in special situations. Since double precision is the default—and what will be used in this class—we will focus here on its representation.

1.2 IEEE double precision format

Double precision makes use of 8 byte words (64 bits), and usually results in sufficiently accurate computations. The format for a double precision number is



$$\# = (-1)^{s} \times 2^{e-1023} \times 1.f,$$

where s is the sign bit, e is the biased exponent, and 1.f (using a binary point) is the significand. We see that the 64 bits are distributed so that the sign uses 1-bit, the exponent uses 11-bits, and the significand uses 52-bits. The distribution of bits between the exponent and the significand is to reconcile two conflicting desires: that the numbers should range from very large to very small values, and that the relative spacing between numbers should be small.

1.3 Machine numbers

We show for illustration how the numbers 1, 2, and 1/2 are represented in double precision. For the number 1, we write

$$1.0 = (-1)^0 \times 2^{(1023-1023)} \times 1.0\,. \tag{1.1}$$

From (1.1), we find s = 0, e = 1023, and f = 0. Now

$$1023 = 011\,1111\,1111(\text{base } 2),$$

so that the machine representation of the number 1 is given by

$$0011\,1111\,1111\,0000\,\ldots\,0000\,. \tag{1.2}$$

One can view the machine representation of numbers in MATLAB using the `format hex` command. MATLAB then displays the hex number corresponding to the binary machine number. Hex maps four bit binary numbers to a single character, where the binary numbers corresponding to the decimal 0–9 are mapped to the same decimal numbers, and the binary numbers corresponding to 10–15 are mapped to a–f. The hex representation of the number 1 given by (1.2) is therefore given by

$$3\text{ff}0\,0000\,0000\,0000.$$

For the number 2, we have

$$2.0 = (-1)^0 \times 2^{(1024-1023)} \times 1.0,$$

with binary representation

$$0100\,0000\,0000\,0000\,\ldots\,0000,$$

and hex representation

$$4000\,0000\,0000\,0000.$$

For the number 1/2, we have

$$2.0 = (-1)^0 \times 2^{(1022-1023)} \times 1.0,$$

with binary representation

$$0011\,1111\,1110\,0000\,\ldots\,0000,$$

and hex representation

$$4\text{fe}0\,0000\,0000\,0000.$$

The numbers 1, 2 and 1/2 can be represented exactly in double precision. But very large integers, and most real numbers can not. For example, the number 1/3 is inexact, and so is 1/5, which we consider here. We write

$$\begin{aligned}\frac{1}{5} &= (-1)^0 \times \frac{1}{8} \times (1+\frac{3}{5}),\\ &= (-1)^0 \times 2^{1020-1023} \times (1+\frac{3}{5}),\end{aligned}$$

so that s = 0, e = 1020 = 011 1111 1100 (base 2), and f = 3/5. The reason 1/5 is inexact is that 3/5 does not have a finite representation in binary. To convert 3/5 to binary, we multiply successively by 2 as follows:

0.6 . . .	0.
1.2 . . .	0.1
0.4 . . .	0.10
0.8 . . .	0.100
1.6 . . .	0.1001
etc.	

so that 3/5 exactly in binary is $0.\overline{1001}$, where the bar denotes an endless repetition. With only 52 bits to represent f, the fraction 3/5 is inexact and we have

$$\mathrm{f} = 1001\,1001\,\ldots\,1001\,1010,$$

where we have rounded the repetitive end of the number $\overline{1001}$ to the nearest binary number 1010. Here the rounding is up because the 53-bit is a 1 followed by not all zeros. But rounding can also be down if the 53 bit is a 0. Exactly on the boundary (the 53-bit is a 1 followed by all zeros), rounding is to the nearest even number. In this special situation that occurs only rarely, if the 52-bit is a 0, then rounding is down, and if the 52 bit is a 1, then rounding is up.

The machine number 1/5 is therefore represented as

$$0011\,1111\,1100\,1001\,1001\,\ldots\,1001\,1010,$$

or in hex,

$$3\mathrm{fc}999999999999\mathrm{a}\,.$$

The small error in representing a number such as 1/5 in double precision usually makes little difference in a computation. One is usually satisfied to obtain results accurate to a few significant digits. Nevertheless, computational scientists need to be aware that most numbers are not represented exactly.

For example, consider subtracting what should be two identical real numbers that are not identical in the computer. In MATLAB, if one enters on the command line

```
5*1/5- 5*(1/5)
```

the resulting answer is `0`, as one would expect. But if one enters

```
5^2*1/5^2 - 5^2*(1/5)^2
```

the resulting answer is `-2.2204e-16`, a number slightly different than zero. The reason for this error is that although the number $5^2 * 1/5^2 = 25/25 = 1$ can be represented exactly, the number $5^2 * (1/5)^2$ is inexact and slightly greater than one. Testing for exactly zero in a calculation that depends on the cancelation of real numbers, then, may not work. More problematic, though, are calculations that subtract two large numbers with the hope of obtaining a sensible small number. A total loss of precision of the small number may occur.

1.4 Special numbers

Both the largest and smallest exponent are reserved in IEEE arithmetic. When f $= 0$, the largest exponent, e $= 111\,1111\,1111$, is used to represent $\pm\infty$ (written in MATLAB as `Inf` and `-Inf`). When f $\neq 0$, the largest exponent is used to represent 'not a number' (written in MATLAB as `NaN`). IEEE arithmetic also implements what is called denormal numbers, also called graceful underflow. It reserves the smallest exponent, e $= 000\,0000\,0000$, to represent numbers for which the representation changes from 1.f to 0.f.

With the largest exponent reserved, the largest positive double precision number has s $= 0$, e $= 111\,1111\,1110 = 2046$, and f $= 1111\,1111\ldots1111 = 1 - 2^{-52}$. Called `realmax` in MATLAB, we have

```
realmax = 1.7977e+308 .
```

With the smallest exponent reserved, the smallest positive double precision number has s $= 0$, e $= 000\,0000\,0001 = 1$, and f $= 0000\,0000\ldots0000 = 0$. Called `realmin` in MATLAB, we have

```
realmin = 2.2251e-308 .
```

Above `realmax`, one obtains `Inf`, and below `realmin`, one obtains first denormal numbers and then 0.

Another important number is called machine epsilon (called `eps` in MATLAB). Machine epsilon is defined as the distance between 1 and the next largest number. If $0 \leq \delta <$ `eps`$/2$, then $1 + \delta = 1$ in computer math. Also since

$$x + y = x(1 + y/x),$$

if $0 \leq y/x <$ `eps`$/2$, then $x + y = x$ in computer math.

Now, the number 1 in double precision IEEE format is written as

$$1 = 2^0 \times 1.000\ldots0,$$

with 52 0's following the binary point. The number just larger than 1 has a 1 in the 52nd position after the decimal point. Therefore,

$$\texttt{eps} = 2^{-52} \approx 2.2204e-016.$$

What is the distance between 1 and the number just smaller than 1? Here, the number just smaller than one can be written as

$$2^{-1} \times 1.111\ldots1 = 2^{-1}(1 + (1 - 2^{-52})) = 1 - 2^{-53}$$

Therefore, this distance is $2^{-53} =$ `eps`$/2$.

Here, we observe that the spacing between numbers is uniform between powers of 2, but changes by a factor of two with each additional power of two. For example, the spacing of numbers between 1 and 2 is 2^{-52}, between 2 and 4 is 2^{-51}, between 4 and 8 is 2^{-50}, etc. An exception occurs for denormal numbers, where the spacing becomes uniform all the way down to zero. Denormal numbers implement a graceful underflow, and are not to be used for normal computation.

1.5 Roundoff error

Consider solving the quadratic equation

$$x^2 + 2bx - 1 = 0,$$

where b is a parameter. The quadratic formula yields the two solutions

$$x_\pm = -b \pm \sqrt{b^2+1}.$$

Now, consider the solution with $b > 0$ and $x > 0$ (the x_+ solution) given by

$$x = -b + \sqrt{b^2+1}. \tag{1.3}$$

As $b \to \infty$,

$$\begin{aligned} x &= -b + \sqrt{b^2+1} \\ &= -b + b\sqrt{1+1/b^2} \\ &= b(\sqrt{1+1/b^2} - 1) \\ &\approx b\left(1 + \frac{1}{2b^2} - 1\right) \\ &= \frac{1}{2b}. \end{aligned}$$

Now in double precision, `realmin` $\approx 2.2 \times 10^{-308}$ and in professional software one would like x to be accurate to this value before it goes to 0 via denormal numbers. Therefore, x should be computed accurately to $b \approx 1/(2 \times \texttt{realmin}) \approx 2 \times 10^{307}$. What happens if we compute (1.3) directly? Then $x = 0$ when $b^2 + 1 = b^2$, or $1 + 1/b^2 = 1$. Therefore, $x = 0$ when $1/b^2 < \texttt{eps}/2$, or $b > \sqrt{2/\texttt{eps}} \approx 10^8$.

The way that a professional mathematical software designer solves this problem is to compute the solution for x when $b > 0$ as

$$x = \frac{1}{b + \sqrt{b^2+1}}.$$

In this form, when $b^2 + 1 = b^2$, then $x = 1/2b$, which is the correct asymptotic form.

Chapter 2

Root finding

The problem is to solve $f(x) = 0$ for x when an explicit analytical solution is impossible. All methods compute a sequence $x_0, x_1, x_2, \ldots$ that converges to the root $x = r$ satisfying $f(r) = 0$.

2.1 Bisection Method

The bisection method is conceptually the simplest method and almost always works. However, it is also the slowest method, and most of the time should be avoided.

We first choose x_0 and x_1 such that $x_0 < r < x_1$. We say that x_0 and x_1 bracket the root. With $f(r) = 0$, we want $f(x_0)$ and $f(x_1)$ to be of opposite sign, so that $f(x_0)f(x_1) < 0$. We then assign x_2 to be the midpoint of x_0 and x_1, that is $x_2 = (x_1 + x_0)/2$, or

$$x_2 = x_1 - \frac{x_1 - x_0}{2}.$$

The sign of $f(x_2)$ is then determined, and the value of x_3 is chosen as either the midpoint of x_2 and x_0 or as the midpoint of x_2 and x_1, depending on whether x_2 and x_0 bracket the root, or x_2 and x_1 bracket the root. The root, therefore, stays bracketed at all times. The algorithm proceeds in this fashion and is typically stopped when the absolute value of the increment to the last best approximation to the root (above, given by $|x_1 - x_0|/2$) is smaller than some required precision.

2.1.1 Estimate $\sqrt{2} = 1.41421\ldots$ using $x_0 = 1$ and $x_1 = 2$.

Now $\sqrt{2}$ is the zero of the function $f(x) = x^2 - 2$. We iterate with $x_0 = 1$ and $x_1 = 2$. We have

$$x_2 = 2 - \frac{2-1}{2} = \frac{3}{2} = 1.5.$$

Now, $f(x_2) = 9/4 - 2 = 1/4 > 0$ so that x_2 and x_0 bracket the root. Therefore,

$$x_3 = \frac{3}{2} - \frac{\frac{3}{2} - 1}{2} = \frac{5}{4} = 1.25.$$

Now, $f(x_3) = 25/16 - 2 = -7/16 < 0$ so that x_3 and x_2 bracket the root. Therefore,

$$x_4 = \frac{5}{4} - \frac{\frac{5}{4} - \frac{3}{2}}{2} = \frac{11}{8} = 1.375,$$

and so on.

2.2 Newton's Method

This is the fastest method, but requires analytical computation of the derivative of $f(x)$. If the derivative is known, then this method should be used, although convergence to the desired root is not guaranteed.

We can derive Newton's Method from a Taylor series expansion. We again want to construct a sequence $x_0, x_1, x_2, \ldots$ that converges to the root $x = r$. Consider the x_{n+1} member of this sequence, and Taylor series expand $f(x_{n+1})$ about the point x_n. We have

$$f(x_{n+1}) = f(x_n) + (x_{n+1} - x_n)f'(x_n) + \ldots.$$

To determine x_{n+1}, we drop the higher-order terms in the Taylor series, and assume $f(x_{n+1}) = 0$. Solving for x_{n+1}, we have

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}.$$

Starting Newton's Method requires a guess for x_0, to be chosen as close as possible to the root $x = r$.

2.2.1 Estimate $\sqrt{2}$ using $x_0 = 1$.

Again, we solve $f(x) = 0$, where $f(x) = x^2 - 2$. To implement Newton's Method, we use $f'(x) = 2x$. Therefore, Newton's Method is the iteration

$$x_{n+1} = x_n - \frac{x_n^2 - 2}{2x_n}.$$

With our initial guess $x_0 = 1$, we have

$$\begin{aligned}
x_1 &= 1 - \frac{-1}{2} = \frac{3}{2} = 1.5, \\
x_2 &= \frac{3}{2} - \frac{\frac{9}{4} - 2}{3} = \frac{17}{12} = 1.416667, \\
x_3 &= \frac{17}{12} - \frac{\frac{17^2}{12^2} - 2}{\frac{17}{6}} = \frac{577}{408} = 1.41426.
\end{aligned}$$

2.3 Secant Method

The Secant Method is second fastest to Newton's Method, and is most often used when it is not possible to take an analytical derivative of the function $f(x)$. We write in place of $f'(x_n)$,

$$f'(x_n) \approx \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}.$$

Starting the Secant Method requires a guess for both x_0 and x_1. These values need not bracket the root, but convergence is not guaranteed.

2.3.1 Estimate $\sqrt{2}$ using $x_0 = 1$ and $x_1 = 2$.

Again, we solve $f(x) = 0$, where $f(x) = x^2 - 2$. The Secant Method iterates

$$\begin{aligned} x_{n+1} &= x_n - \frac{(x_n^2 - 2)(x_n - x_{n-1})}{x_n^2 - x_{n-1}^2} \\ &= x_n - \frac{x_n^2 - 2}{x_n + x_{n-1}}. \end{aligned}$$

With $x_0 = 1$ and $x_1 = 2$, we have

$$\begin{aligned} x_2 &= 2 - \frac{4-2}{3} = \frac{4}{3} = 1.33333, \\ x_3 &= \frac{4}{3} - \frac{\frac{16}{9} - 2}{\frac{4}{3} + 2} = \frac{21}{15} = 1.4, \\ x_4 &= \frac{21}{15} - \frac{\left(\frac{21}{15}\right)^2 - 2}{\frac{21}{15} + \frac{4}{3}} = \frac{174}{123} = 1.41463. \end{aligned}$$

2.4 A fractal from Newton's Method

Consider the complex roots of the equation $f(z) = 0$, where

$$f(z) = z^3 - 1.$$

These roots are the three cubic roots of unity. With

$$e^{i2\pi n} = 1, \quad n = 0, 1, 2, \ldots$$

the three unique cubic roots of unity are given by

$$1, \quad e^{i2\pi/3}, \quad e^{i4\pi/3}.$$

With

$$e^{i\theta} = \cos\theta + i\sin\theta,$$

and $\cos(2\pi/3) = -1/2$, $\sin(2\pi/3) = \sqrt{3}/2$, the three cubic roots of unity are

$$r_1 = 1, \quad r_2 = -\frac{1}{2} + \frac{\sqrt{3}}{2}i, \quad r_3 = -\frac{1}{2} - \frac{\sqrt{3}}{2}i.$$

The interesting idea here is to determine in the complex plane which initial values of z_0 converge using Newton's method to which of the three cube roots of unity.

Newton's method generalized to the complex plane is

$$z_{n+1} = z_n - \frac{z_n^3 - 1}{3z_n^2}.$$

If the iteration converges to r_1, we color z_0 red; r_2, blue; r_3, green. The result will be shown in lecture.

2.5 Order of convergence

Let r be the root and x_n be the nth approximation to the root. Define the error as

$$\epsilon_n = r - x_n.$$

If for large n we have the approximate relationship

$$|\epsilon_{n+1}| = k|\epsilon_n|^p,$$

with k a positive constant and $p \geq 1$, then we say the root-finding numerical method is of order p. Larger values of p correspond to faster convergence to the root. The order of convergence of bisection is one: the error is reduced by approximately a factor of 2 with each iteration so that

$$|\epsilon_{n+1}| = \frac{1}{2}|\epsilon_n|.$$

We now find the order of convergence for Newton's Method and for the Secant Method.

2.5.1 Newton's Method

We start with Newton's Method

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)},$$

where the root r satisfies $f(r) = 0$. Subtracting both sides from r, we have

$$r - x_{n+1} = r - x_n + \frac{f(x_n)}{f'(x_n)},$$

or

$$\epsilon_{n+1} = \epsilon_n + \frac{f(x_n)}{f'(x_n)}. \tag{2.1}$$

We use Taylor series to expand the functions $f(x_n)$ and $f'(x_n)$ about the root r, using $f(r) = 0$. We have

$$\begin{aligned} f(x_n) &= f(r) + (x_n - r)f'(r) + \frac{1}{2}(x_n - r)^2 f''(r) + \ldots, \\ &= -\epsilon_n f'(r) + \frac{1}{2}\epsilon_n^2 f''(r) + \ldots; \\ f'(x_n) &= f'(r) + (x_n - r)f''(r) + \frac{1}{2}(x_n - r)^2 f'''(r) + \ldots, \\ &= f'(r) - \epsilon_n f''(r) + \frac{1}{2}\epsilon_n^2 f'''(r) + \ldots. \end{aligned} \tag{2.2}$$

To make further progress, we will make use of the following standard Taylor series:

$$\frac{1}{1-\epsilon} = 1 + \epsilon + \epsilon^2 + \ldots, \tag{2.3}$$

which converges for $|\epsilon| < 1$. Substituting (2.2) into (2.1), and using (2.3) yields

$$\begin{aligned}
\epsilon_{n+1} &= \epsilon_n + \frac{f(x_n)}{f'(x_n)} \\
&= \epsilon_n + \frac{-\epsilon_n f'(r) + \frac{1}{2}\epsilon_n^2 f''(r) + \dots}{f'(r) - \epsilon_n f''(r) + \frac{1}{2}\epsilon_n^2 f'''(r) + \dots} \\
&= \epsilon_n + \frac{-\epsilon_n + \frac{1}{2}\epsilon_n^2 \frac{f''(r)}{f'(r)} + \dots}{1 - \epsilon_n \frac{f''(r)}{f'(r)} + \dots} \\
&= \epsilon_n + \left(-\epsilon_n + \frac{1}{2}\epsilon_n^2 \frac{f''(r)}{f'(r)} + \dots\right)\left(1 + \epsilon_n \frac{f''(r)}{f'(r)} + \dots\right) \\
&= \epsilon_n + \left(-\epsilon_n + \epsilon_n^2\left(\frac{1}{2}\frac{f''(r)}{f'(r)} - \frac{f''(r)}{f'(r)}\right) + \dots\right) \\
&= -\frac{1}{2}\frac{f''(r)}{f'(r)}\epsilon_n^2 + \dots
\end{aligned}$$

Therefore, we have shown that

$$|\epsilon_{n+1}| = k|\epsilon_n|^2$$

as $n \to \infty$, with

$$k = \frac{1}{2}\left|\frac{f''(r)}{f'(r)}\right|,$$

provided $f'(r) \neq 0$. Newton's method is thus of order 2 at simple roots.

2.5.2 Secant Method

Determining the order of the Secant Method proceeds in a similar fashion. We start with

$$x_{n+1} = x_n - \frac{(x_n - x_{n-1})f(x_n)}{f(x_n) - f(x_{n-1})}.$$

We subtract both sides from r and make use of

$$\begin{aligned}
x_n - x_{n-1} &= (r - x_{n-1}) - (r - x_n) \\
&= \epsilon_{n-1} - \epsilon_n,
\end{aligned}$$

and the Taylor series

$$\begin{aligned}
f(x_n) &= -\epsilon_n f'(r) + \frac{1}{2}\epsilon_n^2 f''(r) + \dots, \\
f(x_{n-1}) &= -\epsilon_{n-1} f'(r) + \frac{1}{2}\epsilon_{n-1}^2 f''(r) + \dots,
\end{aligned}$$

so that

$$\begin{aligned}
f(x_n) - f(x_{n-1}) &= (\epsilon_{n-1} - \epsilon_n)f'(r) + \frac{1}{2}(\epsilon_n^2 - \epsilon_{n-1}^2)f''(r) + \dots \\
&= (\epsilon_{n-1} - \epsilon_n)\left(f'(r) - \frac{1}{2}(\epsilon_{n-1} + \epsilon_n)f''(r) + \dots\right).
\end{aligned}$$

We therefore have

$$\begin{aligned}
\epsilon_{n+1} &= \epsilon_n + \frac{-\epsilon_n f'(r) + \frac{1}{2}\epsilon_n^2 f''(r) + \dots}{f'(r) - \frac{1}{2}(\epsilon_{n-1} + \epsilon_n) f''(r) + \dots} \\
&= \epsilon_n - \epsilon_n \frac{1 - \frac{1}{2}\epsilon_n \frac{f''(r)}{f'(r)} + \dots}{1 - \frac{1}{2}(\epsilon_{n-1} + \epsilon_n)\frac{f''(r)}{f'(r)} + \dots} \\
&= \epsilon_n - \epsilon_n \left(1 - \frac{1}{2}\epsilon_n \frac{f''(r)}{f'(r)} + \dots\right)\left(1 + \frac{1}{2}(\epsilon_{n-1} + \epsilon_n)\frac{f''(r)}{f'(r)} + \dots\right) \\
&= -\frac{1}{2}\frac{f''(r)}{f'(r)}\epsilon_{n-1}\epsilon_n + \dots,
\end{aligned}$$

or to leading order

$$|\epsilon_{n+1}| = \frac{1}{2}\left|\frac{f''(r)}{f'(r)}\right| |\epsilon_{n-1}||\epsilon_n|. \tag{2.4}$$

The order of convergence is not yet obvious from this equation, but should be less than quadratic because $|\epsilon_{n-1}| > |\epsilon_n|$. To determine the scaling law we look for a solution of the form

$$|\epsilon_{n+1}| = k|\epsilon_n|^p.$$

From this ansatz, we also have

$$|\epsilon_n| = k|\epsilon_{n-1}|^p,$$

and therefore

$$|\epsilon_{n+1}| = k^{p+1}|\epsilon_{n-1}|^{p^2}.$$

Substitution into (2.4) results in

$$k^{p+1}|\epsilon_{n-1}|^{p^2} = \frac{k}{2}\left|\frac{f''(r)}{f'(r)}\right| |\epsilon_{n-1}|^{p+1}.$$

Equating the coefficient and the power of ϵ_{n-1} results in

$$k^p = \frac{1}{2}\left|\frac{f''(r)}{f'(r)}\right|,$$

and

$$p^2 = p + 1.$$

The order of convergence of the Secant Method, given by p, is therefore the positive root of the quadratic equation $p^2 - p - 1 = 0$, or

$$p = \frac{1 + \sqrt{5}}{2} \approx 1.618,$$

which coincidentally is a famous irrational number that is called The Golden Ratio, and goes by the symbol Φ. We see that the Secant Method has an order of convergence lying between the Bisection Method and Newton's Method.

Chapter 3

Integration

We want to construct numerical algorithms that can perform definite integrals of the form

$$I = \int_a^b f(x)dx. \tag{3.1}$$

Calculating these definite integrals numerically is called *numerical integration*, *numerical quadrature*, or more simply *quadrature*.

3.1 Elementary formulas

We first consider integration from 0 to h, with h small, to serve as the building blocks for integration over larger domains. We here define I_h as the following integral:

$$I_h = \int_0^h f(x)dx. \tag{3.2}$$

To perform this integral, we consider a Taylor series expansion of $f(x)$ about the value $x = h/2$:

$$f(x) = f(h/2) + (x - h/2)f'(h/2) + \frac{(x-h/2)^2}{2}f''(h/2) + \frac{(x-h/2)^3}{6}f'''(h/2) + \frac{(x-h/2)^4}{24}f''''(h/2) + \dots$$

3.1.1 Midpoint rule

The midpoint rule makes use of only the first term in the Taylor series expansion. Here, we will determine the error in this approximation. Integrating,

$$I_h = hf(h/2) + \int_0^h \left((x - h/2)f'(h/2) + \frac{(x-h/2)^2}{2}f''(h/2) + \frac{(x-h/2)^3}{6}f'''(h/2) + \frac{(x-h/2)^4}{24}f''''(h/2) + \dots \right) dx.$$

Changing variables by letting $y = x - h/2$ and $dy = dx$, and simplifying the integral depending on whether the integrand is even or odd, we have

$$\begin{aligned} I_h &= hf(h/2) \\ &\quad + \int_{-h/2}^{h/2} \left(yf'(h/2) + \frac{y^2}{2}f''(h/2) + \frac{y^3}{6}f'''(h/2) + \frac{y^4}{24}f''''(h/2) + \dots \right) dy \\ &= hf(h/2) + \int_0^{h/2} \left(y^2 f''(h/2) + \frac{y^4}{12}f''''(h/2) + \dots \right) dy. \end{aligned}$$

The integrals that we need here are

$$\int_0^{\frac{h}{2}} y^2 dy = \frac{h^3}{24}, \quad \int_0^{\frac{h}{2}} y^4 dy = \frac{h^5}{160}.$$

Therefore,

$$I_h = hf(h/2) + \frac{h^3}{24} f''(h/2) + \frac{h^5}{1920} f''''(h/2) + \dots. \tag{3.3}$$

3.1.2 Trapezoidal rule

From the Taylor series expansion of $f(x)$ about $x = h/2$, we have

$$f(0) = f(h/2) - \frac{h}{2} f'(h/2) + \frac{h^2}{8} f''(h/2) - \frac{h^3}{48} f'''(h/2) + \frac{h^4}{384} f''''(h/2) + \dots,$$

and

$$f(h) = f(h/2) + \frac{h}{2} f'(h/2) + \frac{h^2}{8} f''(h/2) + \frac{h^3}{48} f'''(h/2) + \frac{h^4}{384} f''''(h/2) + \dots.$$

Adding and multiplying by $h/2$ we obtain

$$\frac{h}{2}\big(f(0) + f(h)\big) = hf(h/2) + \frac{h^3}{8} f''(h/2) + \frac{h^5}{384} f''''(h/2) + \dots.$$

We now substitute for the first term on the right-hand-side using the midpoint rule formula:

$$\frac{h}{2}\big(f(0) + f(h)\big) = \left(I_h - \frac{h^3}{24} f''(h/2) - \frac{h^5}{1920} f''''(h/2)\right) + \frac{h^3}{8} f''(h/2) + \frac{h^5}{384} f''''(h/2) + \dots,$$

and solving for I_h, we find

$$I_h = \frac{h}{2}\big(f(0) + f(h)\big) - \frac{h^3}{12} f''(h/2) - \frac{h^5}{480} f''''(h/2) + \dots. \tag{3.4}$$

3.1.3 Simpson's rule

To obtain Simpson's rule, we combine the midpoint and trapezoidal rule to eliminate the error term proportional to h^3. Multiplying (3.3) by two and adding to (3.4), we obtain

$$3I_h = h\left(2f(h/2) + \frac{1}{2}\big(f(0) + f(h)\big)\right) + h^5\left(\frac{2}{1920} - \frac{1}{480}\right) f''''(h/2) + \dots,$$

or

$$I_h = \frac{h}{6}\big(f(0) + 4f(h/2) + f(h)\big) - \frac{h^5}{2880} f''''(h/2) + \dots.$$

Usually, Simpson's rule is written by considering the three consecutive points 0, h and $2h$. Substituting $h \to 2h$, we obtain the standard result

$$I_{2h} = \frac{h}{3}\big(f(0) + 4f(h) + f(2h)\big) - \frac{h^5}{90} f''''(h) + \dots. \tag{3.5}$$

3.2 Composite rules

We now use our elementary formulas obtained for (3.2) to perform the integral given by (3.1).

3.2.1 Trapezoidal rule

We suppose that the function $f(x)$ is known at the $n+1$ points labeled as $x_0, x_1, \ldots, x_n$, with the endpoints given by $x_0 = a$ and $x_n = b$. Define

$$f_i = f(x_i), \quad h_i = x_{i+1} - x_i.$$

Then the integral of (3.1) may be decomposed as

$$\begin{aligned}\int_a^b f(x)dx &= \sum_{i=0}^{n-1} \int_{x_i}^{x_{i+1}} f(x)dx \\ &= \sum_{i=0}^{n-1} \int_0^{h_i} f(x_i + s)ds,\end{aligned}$$

where the last equality arises from the change-of-variables $s = x - x_i$. Applying the trapezoidal rule to the integral, we have

$$\int_a^b f(x)dx = \frac{1}{2} \sum_{i=0}^{n-1} h_i \left(f_i + f_{i+1}\right). \tag{3.6}$$

If the points are not evenly spaced, say because the data are experimental values, then the h_i may differ for each value of i and (3.6) is to be used directly.

However, if the points are evenly spaced, say because $f(x)$ can be computed, we have $h_i = h$, independent of i. We can then define

$$x_i = a + ih, \quad i = 0, 1, \ldots, n;$$

and since the end point b satisfies $b = a + nh$, we have

$$h = \frac{b-a}{n}.$$

The composite trapezoidal rule for evenly space points then becomes

$$\begin{aligned}\int_a^b f(x)dx &= \frac{h}{2} \sum_{i=0}^{n-1} \left(f_i + f_{i+1}\right) \\ &= \frac{h}{2} \left(f_0 + 2f_1 + \cdots + 2f_{n-1} + f_n\right).\end{aligned} \tag{3.7}$$

The first and last terms have a multiple of one; all other terms have a multiple of two; and the entire sum is multiplied by $h/2$.

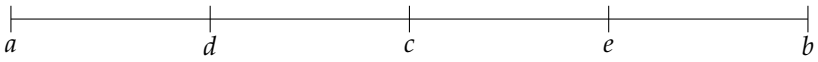


Figure 3.1: Adaptive Simpson quadrature: Level 1.

3.2.2 Simpson's rule

We here consider the composite Simpson's rule for evenly space points. We apply Simpson's rule over intervals of $2h$, starting from a and ending at b:

$$\int_a^b f(x)dx = \frac{h}{3}\left(f_0 + 4f_1 + f_2\right) + \frac{h}{3}\left(f_2 + 4f_3 + f_4\right) + \ldots + \frac{h}{3}\left(f_{n-2} + 4f_{n-1} + f_n\right).$$

Note that n must be even for this scheme to work. Combining terms, we have

$$\int_a^b f(x)dx = \frac{h}{3}\left(f_0 + 4f_1 + 2f_2 + 4f_3 + 2f_4 + \cdots + 4f_{n-1} + f_n\right).$$

The first and last terms have a multiple of one; the even indexed terms have a multiple of 2; the odd indexed terms have a multiple of 4; and the entire sum is multiplied by $h/3$.

3.3 Adaptive integration

The useful MATLAB function quad.m performs numerical integration using adaptive Simpson quadrature. The idea is to let the computation itself decide on the grid size required to achieve a certain level of accuracy. Moreover, the grid size need not be the same over the entire region of integration.

We begin the adaptive integration at what is called Level 1. The uniformly spaced points at which the function $f(x)$ is to be evaluated are shown in Fig. 3.1. The distance between the points a and b is taken to be $2h$, so that

$$h = \frac{b-a}{2}.$$

Integration using Simpson's rule (3.5) with grid size h yields for the integral I,

$$I = \frac{h}{3}\big(f(a) + 4f(c) + f(b)\big) - \frac{h^5}{90}f''''(\xi),$$

where ξ is some value satisfying $a \leq \xi \leq b$.

Integration using Simpson's rule twice with grid size $h/2$ yields

$$I = \frac{h}{6}\big(f(a) + 4f(d) + 2f(c) + 4f(e) + f(b)\big) - \frac{(h/2)^5}{90}f''''(\xi_l) - \frac{(h/2)^5}{90}f''''(\xi_r),$$

with ξ_l and ξ_r some values satisfying $a \le \xi_l \le c$ and $c \le \xi_r \le b$.

We now define the two approximations to the integral by

$$S_1 = \frac{h}{3}\big(f(a) + 4f(c) + f(b)\big),$$
$$S_2 = \frac{h}{6}\big(f(a) + 4f(d) + 2f(c) + 4f(e) + f(b)\big),$$

and the two associated errors by

$$E_1 = -\frac{h^5}{90}f''''(\xi),$$
$$E_2 = -\frac{h^5}{2^5 \cdot 90}\big(f''''(\xi_l) + f''''(\xi_r)\big).$$

We now ask whether the value of S_2 for the integral is accurate enough, or must we further refine the calculation and go to Level 2? To answer this question, we make the simplifying approximation that all of the fourth-order derivatives of $f(x)$ in the error terms are equal; that is,

$$f''''(\xi) = f''''(\xi_l) = f''''(\xi_r) = C.$$

Then

$$E_1 = -\frac{h^5}{90}C,$$
$$E_2 = -\frac{h^5}{2^4 \cdot 90}C = \frac{1}{16}E_1.$$

Now since the integral is equal to the approximation plus its associated error,

$$S_1 + E_1 = S_2 + E_2,$$

and since

$$E_1 = 16E_2,$$

we can derive an estimate for the error term E_2:

$$E_2 = \frac{1}{15}(S_2 - S_1).$$

Therefore, given some specific value of the tolerance tol, if

$$\left|\frac{1}{15}(S_2 - S_1)\right| < \text{tol},$$

then we can accept S_2 as I. If the error estimate is larger in magnitude than tol, then we proceed to Level 2.

The computation at Level 2 further divides the integration interval from a to b into the two integration intervals a to c and c to b, and proceeds with the above procedure independently on both halves. Integration can be stopped on either half provided the tolerance is less than tol/2 (since the sum of both errors must be less than tol). Otherwise, either half can proceed to Level 3, and so on.

As a side note, the two values of I given above (for integration with step size h and $h/2$) can be combined to give a more accurate value for I given by

$$I = \frac{16S_2 - S_1}{15} + \mathrm{O}(h^7),$$

where the error terms of $\mathrm{O}(h^5)$ approximately cancel. This free lunch, so to speak, is called Richardson's extrapolation.

Chapter 4

Differential equations

We now discuss the numerical solution of ordinary differential equations. We will include the initial value problem and the boundary value problem.

4.1 Initial value problem

We begin with the simple Euler method, then discuss the more sophisticated Runge-Kutta methods, and conclude with the Runge-Kutta-Fehlberg method, as implemented in the MATLAB function `ode45.m`. Our differential equations are for $x = x(t)$, where the time t is the independent variable.

4.1.1 Euler method

The Euler method is the most straightforward method to integrate a differential equation. Consider the first-order differential equation

$$\dot{x} = f(t, x), \tag{4.1}$$

with the initial condition $x(0) = x_0$. Define $t_n = n\Delta t$ and $x_n = x(t_n)$. A Taylor series expansion of x_{n+1} results in

$$\begin{aligned} x_{n+1} &= x(t_n + \Delta t) \\ &= x(t_n) + \Delta t \dot{x}(t_n) + \mathrm{O}(\Delta t^2) \\ &= x(t_n) + \Delta t f(t_n, x_n) + \mathrm{O}(\Delta t^2). \end{aligned}$$

The Euler Method is therefore written as

$$x_{n+1} = x(t_n) + \Delta t f(t_n, x_n).$$

We say that the Euler method steps forward in time using a time-step Δt, starting from the initial value $x_0 = x(0)$. The local error of the Euler Method is $\mathrm{O}(\Delta t^2)$. The global error, however, incurred when integrating to a time T, is a factor of $1/\Delta t$ larger and is given by $\mathrm{O}(\Delta t)$. It is therefore customary to call the Euler Method a *first-order method*.

4.1.2 Modified Euler method

This method is a so-called predictor-corrector method. It is also the first of what we will see are Runge-Kutta methods. As before, we want to solve (4.1). The idea is to average the value of $\dot{x}$ at the beginning and end of the time step. That is, we would like to modify the Euler method and write

$$x_{n+1} = x_n + \frac{1}{2}\Delta t\big(f(t_n, x_n) + f(t_n + \Delta t, x_{n+1})\big).$$

The obvious problem with this formula is that the unknown value x_{n+1} appears on the right-hand-side. We can, however, estimate this value, in what is called the predictor step. For the predictor step, we use the Euler method to find

$$x_{n+1}^p = x_n + \Delta t f(t_n, x_n).$$

The corrector step then becomes

$$x_{n+1} = x_n + \frac{1}{2}\Delta t\big(f(t_n, x_n) + f(t_n + \Delta t, x_{n+1}^p)\big).$$

The Modified Euler Method can be rewritten in the following form that we will later identify as a Runge-Kutta method:

$$\begin{aligned} k_1 &= \Delta t f(t_n, x_n), \\ k_2 &= \Delta t f(t_n + \Delta t, x_n + k_1), \\ x_{n+1} &= x_n + \frac{1}{2}(k_1 + k_2). \end{aligned} \tag{4.2}$$

4.1.3 Second-order Runge-Kutta methods

We now derive the complete family of second-order Runge-Kutta methods. Higher-order methods can be similarly derived, but require substantially more algebra.

We again consider the differential equation given by (4.1). A general second-order Runge-Kutta method may be written in the form

$$\begin{aligned} k_1 &= \Delta t f(t_n, x_n), \\ k_2 &= \Delta t f(t_n + \alpha\Delta t, x_n + \beta k_1), \\ x_{n+1} &= x_n + a k_1 + b k_2, \end{aligned} \tag{4.3}$$

with α, β, a and b constants that define the particular second-order Runge-Kutta method. These constants are to be constrained by setting the local error of the second-order Runge-Kutta method to be $\mathrm{O}(\Delta t^3)$. Intuitively, we might guess that two of the constraints will be $a + b = 1$ and $\alpha = \beta$.

We compute the Taylor series of x_{n+1} directly, and from the Runge-Kutta method, and require them to be the same to order Δt^2. First, we compute the Taylor series of x_{n+1}. We have

$$\begin{aligned} x_{n+1} &= x(t_n + \Delta t) \\ &= x(t_n) + \Delta t\dot{x}(t_n) + \frac{1}{2}(\Delta t)^2\ddot{x}(t_n) + \mathrm{O}(\Delta t^3). \end{aligned}$$

Now,

$$\dot{x}(t_n) = f(t_n, x_n).$$

The second derivative is more complicated and requires partial derivatives. We have

$$\begin{aligned} \ddot{x}(t_n) &= \frac{d}{dt}f(t, x(t))\Big]_{t=t_n} \\ &= f_t(t_n, x_n) + \dot{x}(t_n)f_x(t_n, x_n) \\ &= f_t(t_n, x_n) + f(t_n, x_n)f_x(t_n, x_n). \end{aligned}$$

Therefore,

$$x_{n+1} = x_n + \Delta t f(t_n, x_n) + \frac{1}{2}(\Delta t)^2 \left(f_t(t_n, x_n) + f(t_n, x_n) f_x(t_n, x_n)\right) + \mathrm{O}(\Delta t^3). \quad (4.4)$$

Second, we compute x_{n+1} from the Runge-Kutta method given by (4.3). Combining (4.3) into a single expression, we have

$$x_{n+1} = x_n + a\Delta t f(t_n, x_n) + b\Delta t f\big(t_n + \alpha\Delta t, x_n + \beta\Delta t f(t_n, x_n)\big) + \mathrm{O}(\Delta t^3).$$

We Taylor series expand using

$$f\big(t_n + \alpha\Delta t, x_n + \beta\Delta t f(t_n, x_n)\big) = f(t_n, x_n) + \alpha\Delta t f_t(t_n, x_n) + \beta\Delta t f(t_n, x_n) f_x(t_n, x_n) + \mathrm{O}(\Delta t^2).$$

The Runge-Kutta formula is therefore

$$x_{n+1} = x_n + (a+b)\Delta t f(t_n, x_n) + (\Delta t)^2\big(\alpha b f_t(t_n, x_n) + \beta b f(t_n, x_n) f_x(t_n, x_n)\big) + \mathrm{O}(\Delta t^3). \quad (4.5)$$

Comparing (4.4) and (4.5), we find

$$a + b = 1, \quad \alpha b = 1/2, \quad \beta b = 1/2.$$

Since there are only three equations for four parameters, there exists a family of second-order Runge-Kutta methods.

The modified Euler method given by (4.2) corresponds to $\alpha = \beta = 1$ and $a = b = 1/2$. Another second-order Runge-Kutta method, called the midpoint method, corresponds to $\alpha = \beta = 1/2$, $a = 0$ and $b = 1$. This method is written as

$$\begin{aligned} k_1 &= \Delta t f(t_n, x_n), \\ k_2 &= \Delta t f\left(t_n + \frac{1}{2}\Delta t, x_n + \frac{1}{2}k_1\right), \\ x_{n+1} &= x_n + k_2. \end{aligned}$$

4.1.4 Higher-order Runge-Kutta methods

The general second-order Runge-Kutta method was given by (4.3). The general form of the third-order method is given by

$$\begin{aligned} k_1 &= \Delta t f(t_n, x_n), \\ k_2 &= \Delta t f(t_n + \alpha\Delta t, x_n + \beta k_1), \\ k_3 &= \Delta t f(t_n + \gamma\Delta t, x_n + \delta k_1 + \epsilon k_2), \\ x_{n+1} &= x_n + a k_1 + b k_2 + c k_3. \end{aligned}$$

The following constraints on the constants can be guessed: $\alpha = \beta$, $\gamma = \delta + \epsilon$, and $a + b + c = 1$. Remaining constraints need to be derived.

The fourth-order method has a k_1, k_2, k_3 and k_4. The fifth-order method requires at least to k_6. The table below gives the minimum order of the method and the number of stages required.

order	2	3	4	5	6	7	8
minimum # stages	2	3	4	6	7	9	11

Because of the jump in the number of stages required between the fourth- and fifth-order methods, the fourth-order Runge-Kutta method has some appeal. The general fourth-order method with four stages has 13 constants and 11 constraints. A particularly simple fourth-order method that has been widely used in the past by physicists is given by

$$
\begin{aligned}
k_1 &= \Delta t f(t_n, x_n), \\
k_2 &= \Delta t f\left(t_n + \frac{1}{2}\Delta t, x_n + \frac{1}{2}k_1\right), \\
k_3 &= \Delta t f\left(t_n + \frac{1}{2}\Delta t, x_n + \frac{1}{2}k_2\right), \\
k_4 &= \Delta t f\left(t_n + \Delta t, x_n + k_3\right), \\
x_{n+1} &= x_n + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right).
\end{aligned}
$$

4.1.5 Adaptive Runge-Kutta Methods

As in adaptive integration, it is useful to devise an ode integrator that automatically finds the appropriate Δt. The Dormand-Prince Method, which is implemented in MATLAB's `ode45.m`, finds the appropriate step size by comparing the results of a fifth-order and fourth-order method. It requires six function evaluations per time step, and constructs both a fifth-order and a fourth-order method from the same function evaluations.

Suppose the fifth-order method finds x_{n+1} with local error $O(\Delta t^6)$, and the fourth-order method finds x'_{n+1} with local error $O(\Delta t^5)$. Let ε be the desired error tolerance of the method, and let e be the actual error. We can estimate e from the difference between the fifth- and fourth-order methods; that is,

$$e = |x_{n+1} - x'_{n+1}|.$$

Now e is of $O(\Delta t^5)$, where Δt is the step size taken. Let $\Delta\tau$ be the estimated step size required to get the desired error ε. Then we have

$$e/\varepsilon = (\Delta t)^5/(\Delta\tau)^5,$$

or solving for $\Delta\tau$,

$$\Delta\tau = \Delta t \left(\frac{\varepsilon}{e}\right)^{1/5}.$$

On the one hand, if the actual error is less that the desired error, $e < \varepsilon$, then we accept x_{n+1} and do the next time step using the larger value of $\Delta\tau$. On the other hand, if the actual error is greater than the desired error, $e > \varepsilon$, then we reject the integration step and redo the time step using the smaller value of $\Delta\tau$. In practice, one usually increases the time step slightly less and decreases the time step slightly more to prevent the wastefulness of too many failed time steps.

4.1.6 System of differential equations

Our numerical methods can be easily adapted to solve higher-order differential equations, or equivalently, a system of differential equations. First, we show how a second-order differential equation can be reduced to two first-order equations. Consider

$$\ddot{x} = f(t, x, \dot{x}).$$

This second-order equation can be rewritten as two first-order equations by defining $u = \dot{x}$. We then have the system

$$\dot{x} = u, \quad \dot{u} = f(t, x, u).$$

This trick also works for higher-order equations. For example, the third-order equation

$$\dddot{x} = f(t, x, \dot{x}, \ddot{x}),$$

can be written as

$$\dot{x} = u, \quad \dot{u} = v, \quad \dot{v} = f(t, x, u, v).$$

We can generalize the Runge-Kutta method to solve a system of differential equations. As an example, consider the following system of two odes,

$$\dot{x} = f(t, x, y), \quad \dot{y} = g(t, x, y),$$

with the initial conditions $x(0) = x_0$ and $y(0) = y_0$. The generalization of the commonly

used fourth-order Runge-Kutta method would be

$$
\begin{aligned}
k_1 &= \Delta t f(t_n, x_n, y_n), \\
l_1 &= \Delta t g(t_n, x_n, y_n), \\
\\
k_2 &= \Delta t f\left(t_n + \frac{1}{2}\Delta t, x_n + \frac{1}{2}k_1, y_n + \frac{1}{2}l_1\right), \\
l_2 &= \Delta t g\left(t_n + \frac{1}{2}\Delta t, x_n + \frac{1}{2}k_1, y_n + \frac{1}{2}l_1\right), \\
\\
k_3 &= \Delta t f\left(t_n + \frac{1}{2}\Delta t, x_n + \frac{1}{2}k_2, y_n + \frac{1}{2}l_2\right), \\
l_3 &= \Delta t g\left(t_n + \frac{1}{2}\Delta t, x_n + \frac{1}{2}k_2, y_n + \frac{1}{2}l_2\right), \\
\\
k_4 &= \Delta t f\left(t_n + \Delta t, x_n + k_3, y_n + l_3\right), \\
l_4 &= \Delta t g\left(t_n + \Delta t, x_n + k_3, y_n + l_3\right), \\
\\
x_{n+1} &= x_n + \frac{1}{6}\left(k_1 + 2k_2 + 2k_3 + k_4\right), \\
y_{n+1} &= y_n + \frac{1}{6}\left(l_1 + 2l_2 + 2l_3 + l_4\right).
\end{aligned}
$$

4.2 Boundary value problems

4.2.1 Shooting method

We consider the general ode of the form

$$\frac{d^2y}{dx^2} = f(x, y, dy/dx),$$

with two-point boundary conditions $y(0) = A$ and $y(1) = B$. We will first formulate the ode as an initial value problem. We have

$$\frac{dy}{dx} = z, \quad \frac{dz}{dx} = f(x, y, z).$$

The initial condition $y(0) = A$ is known, but the second initial condition $z(0) = b$ is unknown. Our goal is to determine b such that $y(1) = B$.

In fact, this is a root-finding problem for an appropriately defined function. We define the function $F = F(b)$ such that

$$F(b) = y(1) - B,$$

where $y(1)$ is the numerical value obtained from integrating the coupled first-order differential equations with $y(0) = A$ and $z(0) = b$. Our root-finding routine will want to solve $F(b) = 0$. (The method is called *shooting* because the slope of the solution curve for $y = y(x)$ at $x = 0$ is given by b, and the solution hits the value $y(1)$ at $x = 1$. This looks like pointing a gun and trying to shoot the target, which is B.)

To determine the value of b that solves $F(b) = 0$, we iterate using the secant method, given by

$$b_{n+1} = b_n - F(b_n)\frac{b_n - b_{n-1}}{F(b_n) - F(b_{n-1})}.$$

We need to start with two initial guesses for b, solving the ode for the two corresponding values of $y(1)$. Then the secant method will give us the next value of b to try, and we iterate until $|y(1) - B| < \text{tol}$, where tol is some specified tolerance for the error.

Chapter 5

Linear algebra

Consider the system of n linear equations and n unknowns, given by

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n &= b_1, \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n &= b_2, \\ \vdots \qquad\qquad & \qquad \vdots \\ a_{n1}x_1 + a_{n2}x_2 + \cdots + a_{nn}x_n &= b_n. \end{aligned}$$

We can write this system as the matrix equation

$$\mathrm{A}\mathbf{x} = \mathbf{b}, \tag{5.1}$$

with

$$\mathrm{A} = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix}, \quad \mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{pmatrix}.$$

This chapter details the numerical solution of (5.1).

5.1 Gaussian Elimination

The standard numerical algorithm to solve a system of linear equations is called Gaussian Elimination. We can illustrate this algorithm by example.

Consider the system of equations

$$\begin{aligned} -3x_1 + 2x_2 - x_3 &= -1, \\ 6x_1 - 6x_2 + 7x_3 &= -7, \\ 3x_1 - 4x_2 + 4x_3 &= -6. \end{aligned}$$

To perform Gaussian elimination, we form an Augmented Matrix by combining the matrix A with the column vector $\mathbf{b}$:

$$\begin{pmatrix} -3 & 2 & -1 & -1 \\ 6 & -6 & 7 & -7 \\ 3 & -4 & 4 & -6 \end{pmatrix}.$$

Row reduction is then performed on this matrix. Allowed operations are (1) multiply any row by a constant, (2) add multiple of one row to another row, (3) interchange the order of any rows. The goal is to convert the original matrix into an upper-triangular matrix.

We start with the first row of the matrix and work our way down as follows. First we multiply the first row by 2 and add it to the second row, and add the first row to the third row:

$$\begin{pmatrix} -3 & 2 & -1 & -1 \\ 0 & -2 & 5 & -9 \\ 0 & -2 & 3 & -7 \end{pmatrix}.$$

We then go to the second row. We multiply this row by -1 and add it to the third row:

$$\begin{pmatrix} -3 & 2 & -1 & -1 \\ 0 & -2 & 5 & -9 \\ 0 & 0 & -2 & 2 \end{pmatrix}.$$

The resulting equations can be determined from the matrix and are given by

$$\begin{aligned} -3x_1 + 2x_2 - x_3 &= -1 \\ -2x_2 + 5x_3 &= -9 \\ -2x_3 &= 2. \end{aligned}$$

These equations can be solved by backward substitution, starting from the last equation and working backwards. We have

$$\begin{aligned} -2x_3 = 2 &\rightarrow x_3 = -1 \\ -2x_2 = -9 - 5x_3 = -4 &\rightarrow x_2 = 2, \\ -3x_1 = -1 - 2x_2 + x_3 = -6 &\rightarrow x_1 = 2. \end{aligned}$$

Therefore,

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ -1 \end{pmatrix}.$$

5.2 LU decomposition

The process of Gaussian Elimination also results in the factoring of the matrix A to

$$\mathrm{A} = \mathrm{LU},$$

where L is a lower triangular matrix and U is an upper triangular matrix. Using the same matrix A as in the last section, we show how this factorization is realized. We have

$$\begin{pmatrix} -3 & 2 & -1 \\ 6 & -6 & 7 \\ 3 & -4 & 4 \end{pmatrix} \rightarrow \begin{pmatrix} -3 & 2 & -1 \\ 0 & -2 & 5 \\ 0 & -2 & 3 \end{pmatrix} = \mathrm{M}_1\mathrm{A},$$

where

$$\mathrm{M}_1\mathrm{A} = \begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} -3 & 2 & -1 \\ 6 & -6 & 7 \\ 3 & -4 & 4 \end{pmatrix} = \begin{pmatrix} -3 & 2 & -1 \\ 0 & -2 & 5 \\ 0 & -2 & 3 \end{pmatrix}.$$

Note that the matrix M_1 performs row elimination on the first column. Two times the first row is added to the second row and one times the first row is added to the third row. The entries of the column of M_1 come from $2 = -(6/-3)$ and $1 = -(3/-3)$ as required for row elimination. The number -3 is called the pivot.

The next step is

$$\begin{pmatrix} -3 & 2 & -1 \\ 0 & -2 & 5 \\ 0 & -2 & 3 \end{pmatrix} \to \begin{pmatrix} -3 & 2 & -1 \\ 0 & -2 & 5 \\ 0 & 0 & -2 \end{pmatrix} = M_2(M_1A),$$

where

$$M_2(M_1A) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} -3 & 2 & -1 \\ 0 & -2 & 5 \\ 0 & -2 & 3 \end{pmatrix} = \begin{pmatrix} -3 & 2 & -1 \\ 0 & -2 & 5 \\ 0 & 0 & -2 \end{pmatrix}.$$

Here, M_2 multiplies the second row by $-1 = -(-2/-2)$ and adds it to the third row. The pivot is -2.

We now have

$$M_2M_1A = U$$

or

$$A = M_1^{-1}M_2^{-1}U.$$

The inverse matrices are easy to find. The matrix M_1 multiples the first row by 2 and adds it to the second row, and multiplies the first row by 1 and adds it to the third row. To invert these operations, we need to multiply the first row by -2 and add it to the second row, and multiply the first row by -1 and add it to the third row. To check, with

$$M_1M_1^{-1} = I,$$

we have

$$\begin{pmatrix} 1 & 0 & 0 \\ 2 & 1 & 0 \\ 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Similarly,

$$M_2^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix}$$

Therefore,

$$L = M_1^{-1}M_2^{-1}$$

is given by

$$L = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -1 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -1 & 1 & 1 \end{pmatrix},$$

which is lower triangular. The off-diagonal elements of M_1^{-1} and M_2^{-1} are simply combined to form L. Our LU decomposition is therefore

$$\begin{pmatrix} -3 & 2 & -1 \\ 6 & -6 & 7 \\ 3 & -4 & 4 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -1 & 1 & 1 \end{pmatrix} \begin{pmatrix} -3 & 2 & -1 \\ 0 & -2 & 5 \\ 0 & 0 & -2 \end{pmatrix}.$$

Another nice feature of the LU decomposition is that it can be done by overwriting A, therefore saving memory if the matrix A is very large.

The LU decomposition is useful when one needs to solve $A\mathbf{x} = \mathbf{b}$ for $\mathbf{x}$ when A is fixed and there are many different $\mathbf{b}$'s. First one determines L and U using Gaussian elimination. Then one writes

$$(LU)\mathbf{x} = L(U\mathbf{x}) = \mathbf{b}.$$

We let

$$\mathbf{y} = U\mathbf{x},$$

and first solve

$$L\mathbf{y} = \mathbf{b}$$

for $\mathbf{y}$ by forward substitution. We then solve

$$U\mathbf{x} = \mathbf{y}$$

for $\mathbf{x}$ by backward substitution. If we count operations, we can show that solving $(LU)\mathbf{x} = \mathbf{b}$ is substantially faster once L and U are in hand than solving $A\mathbf{x} = \mathbf{b}$ directly by Gaussian elimination.

We now illustrate the solution of $LU\mathbf{x} = \mathbf{b}$ using our previous example, where

$$L = \begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -1 & 1 & 1 \end{pmatrix}, \quad U = \begin{pmatrix} -3 & 2 & -1 \\ 0 & -2 & 5 \\ 0 & 0 & -2 \end{pmatrix}, \quad \mathbf{b} = \begin{pmatrix} -1 \\ -7 \\ -6 \end{pmatrix}.$$

With $\mathbf{y} = U\mathbf{x}$, we first solve $L\mathbf{y} = \mathbf{b}$, that is

$$\begin{pmatrix} 1 & 0 & 0 \\ -2 & 1 & 0 \\ -1 & 1 & 1 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} = \begin{pmatrix} -1 \\ -7 \\ -6 \end{pmatrix}.$$

Using forward substitution

$$\begin{aligned} y_1 &= -1, \\ y_2 &= -7 + 2y_1 = -9, \\ y_3 &= -6 + y_1 - y_2 = 2. \end{aligned}$$

We now solve $U\mathbf{x} = \mathbf{y}$, that is

$$\begin{pmatrix} -3 & 2 & -1 \\ 0 & -2 & 5 \\ 0 & 0 & -2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} -1 \\ -9 \\ 2 \end{pmatrix}.$$

Using backward substitution,

$$\begin{aligned} -2x_3 &= 2 \rightarrow x_3 = -1, \\ -2x_2 &= -9 - 5x_3 = -4 \rightarrow x_2 = 2, \\ -3x_1 &= -1 - 2x_2 + x_3 = -6 \rightarrow x_1 = 2, \end{aligned}$$

and we have once again determined

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 2 \\ 2 \\ -1 \end{pmatrix}.$$

5.3 Partial pivoting

When performing Gaussian elimination, the diagonal element that one uses during the elimination procedure is called the pivot. To obtain the correct multiple, one uses the pivot as the divisor to the elements below the pivot. Gaussian elimination in this form will fail if the pivot is zero. In this situation, a row interchange must be performed.

Even if the pivot is not identically zero, a small value can result in big round-off errors. For very large matrices, one can easily lose all accuracy in the solution. To avoid these round-off errors arising from small pivots, row interchanges are made, and this technique is called partial pivoting (partial pivoting is in contrast to complete pivoting, where both rows and columns are interchanged). We will illustrate by example the LU decomposition using partial pivoting.

Consider

$$A = \begin{pmatrix} -2 & 2 & -1 \\ 6 & -6 & 7 \\ 3 & -8 & 4 \end{pmatrix}.$$

We interchange rows to place the largest element (in absolute value) in the pivot, or a_{11}, position. That is,

$$A \to \begin{pmatrix} 6 & -6 & 7 \\ -2 & 2 & -1 \\ 3 & -8 & 4 \end{pmatrix} = P_{12}A,$$

where

$$P_{12} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

is a permutation matrix that when multiplied on the left interchanges the first and second rows of a matrix. Note that $P_{12}^{-1} = P_{12}$. The elimination step is then

$$P_{12}A \to \begin{pmatrix} 6 & -6 & 7 \\ 0 & 0 & 4/3 \\ 0 & -5 & 1/2 \end{pmatrix} = M_1 P_{12} A,$$

where

$$M_1 = \begin{pmatrix} 1 & 0 & 0 \\ 1/3 & 1 & 0 \\ -1/2 & 0 & 1 \end{pmatrix}.$$

The final step requires one more row interchange:

$$M_1 P_{12} A \to \begin{pmatrix} 6 & -6 & 7 \\ 0 & -5 & 1/2 \\ 0 & 0 & 4/3 \end{pmatrix} = P_{23} M_1 P_{12} A = U.$$

Since the permutation matrices given by P are their own inverses, we can write our result as

$$(P_{23} M_1 P_{23}) P_{23} P_{12} A = U.$$

Multiplication of M on the left by P interchanges rows while multiplication on the right by P interchanges columns. That is,

$$P_{23}\begin{pmatrix} 1 & 0 & 0 \\ 1/3 & 1 & 0 \\ -1/2 & 0 & 1 \end{pmatrix} P_{23} = \begin{pmatrix} 1 & 0 & 0 \\ -1/2 & 0 & 1 \\ 1/3 & 1 & 0 \end{pmatrix} P_{23} = \begin{pmatrix} 1 & 0 & 0 \\ -1/2 & 1 & 0 \\ 1/3 & 0 & 1 \end{pmatrix}.$$

The net result on M_1 is an interchange of the nondiagonal elements $1/3$ and $-1/2$.

We can then multiply by the inverse of $(P_{23}M_1P_{23})$ to obtain

$$P_{23}P_{12}A = (P_{23}M_1P_{23})^{-1}U,$$

which we write as

$$PA = LU.$$

Instead of L, MATLAB will write this as

$$A = (P^{-1}L)U.$$

For convenience, we will just denote $(P^{-1}L)$ by L, but by L here we mean a permutated lower triangular matrix.

5.4 MATLAB programming

In MATLAB, to solve $A\mathbf{x} = \mathbf{b}$ for $\mathbf{x}$ using Gaussian elimination, one types

```
x=A\b;
```

where \ solves for $\mathbf{x}$ using the most efficient algorithm available, depending on the form of A. If A is a general $n \times n$ matrix, then first the LU decomposition of A is found using partial pivoting, and then x is determined from permuted forward and backward substitution. If A is upper or lower triangular, then forward or backward substitution (or their permuted version) is used directly.

If there are many different right-hand-sides, one would first directly find the LU decomposition of A using a function call, and then solve using \. That is, one would iterate for different $\mathbf{b}$'s the following expressions:

```
[L U]=lu(A);
y=L\b;
x=U\y;
```

where the second and third lines can be shortened to

```
x=U\(L\b);
```

where the parenthesis are required. In lecture, I will demonstrate these solutions in MATLAB using the matrix `A = [-3, 2, -1; 6, -6, 7; 3, -4, 4]` and the right-hand-side `b = [-1; -7; -6]`, which is the example that we did by hand.

Although we do not detail the algorithm here, MATLAB can also solve the linear algebra eigenvalue problem. Here, the mathematical problem to solve is given by

$$\mathrm{A}\mathbf{x} = \lambda\mathbf{x},$$

where A is a square matrix, λ are the eigenvalues, and the associated $\mathbf{x}$'s are the eigenvectors. The MATLAB subroutine that solves this problem is eig.m. To only find the eigenvalues of A, one types

```
lambda = eig(A); .
```

To find both the eigenvalues and eigenvectors, one types

```
[v,lambda] = eig(A); .
```

More information can be found from the MATLAB help pages. One of the nice features about programming in MATLAB is that no great sin is commited if one uses a built-in function without spending the time required to fully understand the underlying algorithm.

Chapter 6

Finite difference approximation

We introduce here numerical differentiation, also called finite difference approximation. This technique is commonly used to discretize and solve partial differential equations.

6.1 Finite difference formulas

Consider the Taylor series approximation for $y(x+h)$ and $y(x-h)$, given by

$$y(x+h) = y(x) + hy'(x) + \frac{1}{2}h^2y''(x) + \frac{1}{6}h^3y'''(x) + \frac{1}{24}h^4y''''(x) + \dots,$$
$$y(x-h) = y(x) - hy'(x) + \frac{1}{2}h^2y''(x) - \frac{1}{6}h^3y'''(x) + \frac{1}{24}h^4y''''(x) + \dots.$$

The standard definitions of the derivatives give the first-order approximations

$$y'(x) = \frac{y(x+h) - y(x)}{h} + \mathrm{O}(h),$$
$$y'(x) = \frac{y(x) - y(x-h)}{h} + \mathrm{O}(h).$$

The more widely-used second-order approximation is called the central-difference approximation and is given by

$$y'(x) = \frac{y(x+h) - y(x-h)}{2h} + \mathrm{O}(h^2).$$

The finite difference approximation to the second derivative can be found from considering

$$y(x+h) + y(x-h) = 2y(x) + h^2y''(x) + \frac{1}{12}h^4y''''(x) + \dots,$$

from which we find

$$y''(x) = \frac{y(x+h) - 2y(x) + y(x-h)}{h^2} + \mathrm{O}(h^2).$$

Often a second-order method is required for x on the boundaries of the domain. For a boundary point on the left, a second-order forward difference method requires the additional Taylor series

$$y(x+2h) = y(x) + 2hy'(x) + 2h^2y''(x) + \frac{4}{3}h^3y'''(x) + \dots.$$

We combine the Taylor series for $y(x+h)$ and $y(x+2h)$ to eliminate the term proportional to h^2:

$$y(x+2h) - 4y(x+h) = -3y(x) - 2hy'(x) + \mathrm{O}(\mathrm{h}^3).$$

Therefore,

$$y'(x) = \frac{-3y(x) + 4y(x+h) - y(x+2h)}{2h} + \mathrm{O}(h^2).$$

For a boundary point on the right, we send $h \to -h$ to find

$$y'(x) = \frac{3y(x) - 4y(x-h) + y(x-2h)}{2h} + \mathrm{O}(h^2).$$

6.2 Example: the Laplace equation

As an example of the finite difference technique, let us consider how to discretize the two dimensional Laplace equation

$$\left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\Phi = 0$$

on the rectangular domain $[0,1] \times [0,1]$. We assume here that the values of Φ are known on the boundaries of the domain. We form a two-dimensional grid with N intervals in each direction together with end points that lie on the boundaries by writing

$$\begin{aligned} x_i &= ih, & i &= 0, 1, \ldots, N, \\ y_j &= jh, & j &= 0, 1, \ldots, N, \end{aligned}$$

where $h = 1/N$. We will also denote the value of $\Phi(x_i, y_j)$ by $\Phi_{i,j}$. The finite difference approximation for the second derivatives at the interior point (x_i, y_j) then results in an equation that we write in the form

$$4\Phi_{i,j} - \Phi_{i+1,j} - \Phi_{i-1,j} - \Phi_{i,j+1} - \Phi_{i,j-1} = 0, \tag{6.1}$$

valid for $i = 1, 2, \ldots, N-1$ and $j = 1, 2, \ldots, N-1$. The boundary values $\Phi_{0,j}$, $\Phi_{N,j}$, $\Phi_{i,0}$, and $\Phi_{i,N}$ are assumed to be given.

One can observe that (6.1) represents a system of $(N-1)^2$ linear equations for $(N-1)^2$ unknowns. We can write this as a matrix equation if we decide how to order the unknowns $\Phi_{i,j}$ into a vector. The standard ordering is called natural ordering, and proceeds from the bottom of the grid along rows moving towards the top. This orders the unknowns as

$$\Phi = [\Phi_{1,1}, \Phi_{2,1}, \ldots, \Phi_{(N-1),1}, \ldots, \Phi_{1,(N-1)}, \Phi_{2(,N-1)}, \ldots, \Phi_{(N-1),(N-1)}]^T.$$

To illustrate the construction of the matrix equation, we consider the case $N = 3$, with two interior points in each direction. The four resulting linear equations with the boundary terms written on the right-hand-side are

$$\begin{aligned} 4\Phi_{1,1} - \Phi_{2,1} - \Phi_{1,2} &= \Phi_{0,1} + \Phi_{1,0}, \\ 4\Phi_{2,1} - \Phi_{1,1} - \Phi_{2,2} &= \Phi_{3,1} + \Phi_{2,0}, \\ 4\Phi_{1,2} - \Phi_{2,2} - \Phi_{1,1} &= \Phi_{0,2} + \Phi_{1,3}, \\ 4\Phi_{2,2} - \Phi_{1,2} - \Phi_{2,1} &= \Phi_{3,2} + \Phi_{2,3}; \end{aligned}$$

and the corresponding matrix equation is

$$\begin{pmatrix} 4 & -1 & -1 & 0 \\ -1 & 4 & 0 & -1 \\ -1 & 0 & 4 & -1 \\ 0 & -1 & -1 & 4 \end{pmatrix} \begin{pmatrix} \Phi_{1,1} \\ \Phi_{2,1} \\ \Phi_{1,2} \\ \Phi_{2,2} \end{pmatrix} = \begin{pmatrix} \Phi_{0,1} + \Phi_{1,0} \\ \Phi_{3,1} + \Phi_{2,0} \\ \Phi_{0,2} + \Phi_{1,3} \\ \Phi_{3,2} + \Phi_{2,3} \end{pmatrix}.$$

The pattern here may not be obvious, but the Laplacian matrix decomposes into 2-by-2 block matrices.

With some thought, a general result can be obtained for an N_x-by-N_y grid of internal points (with $N_x + 1$ and $N_y + 1$ intervals in the x- and y-directions, respectively). The Laplacian matrix then decomposes into N_x-by-N_x block matrices. The diagonal contains N_y of these N_x-by-N_x block matrices, each of which are tridiagonal with a 4 on the diagonal and a -1 on the off-diagonals. Immediately above and below these block matrices on the diagonal are $N_y - 1$ block matrices also of size N_x-by-N_x, each of which are diagonal with -1 along the diagonal.

MATLAB code for the Laplacian matrix can be found on the web in the function `sp_laplace.m`. This code was written by Bordner and Saied in 1995, and I have written a more modern and faster version of this code in `sp_laplace_new.m`.

An alternative solution method, which we will later make use of in §17.2.4, includes the boundary values in the solution vector. If one of the boundary conditions is at position n in this vector, then row n of the left-hand-side matrix will have just a one on the diagonal with all other elements equal to zero (i.e, a row of the identity matrix), and the corresponding element in the right-hand-side vector will have the value at the boundary. Here, with the total number of grid points (including the boundary points) in the x- and y directions given by N_x and N_y, the left-hand-side matrix is then generated using `A=sp_laplace_new(N_x,N_y)`, and all the rows corresponding to the boundary values are replaced with the corresponding rows of the identity matrix. This formulation may be slightly easier to code, and also easier to incorporate other more general boundary conditions.

Chapter 7

Iterative methods

7.1 Jacobi, Gauss-Seidel and SOR methods

Iterative methods are often used for solving a system of nonlinear equations. Even for linear systems, iterative methods have some advantages. They may require less memory and may be computationally faster. They are also easier to code. Here, without detailing the theoretical numerical analysis, we will simply explain the related iterative methods that go by the names of the Jacobi method, the Gauss-Seidel method, and the Successive Over Relaxation method (or SOR). We illustrate these three methods by showing how to solve the two-dimensional Poisson equation, an equation that we will later need to solve to determine the flow field past an obstacle.

The Poisson equation is given by

$$-\nabla^2 \Phi = f,$$

where

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

is the usual two-dimensional Laplacian. This could be a linear equation with f independent of Φ, or a nonlinear equation where f may be some nonlinear function of Φ.

After discretizing the Poisson equation using the centered finite difference approximation for the second derivatives, we obtain on a grid with uniform grid spacing h,

$$4\Phi_{i,j} - \Phi_{i+1,j} - \Phi_{i-1,j} - \Phi_{i,j+1} - \Phi_{i,j-1} = h^2 f_{i,j}. \tag{7.1}$$

The Jacobi method simply solves the discretized equation (7.1) for $\Phi_{i,j}$ iteratively. With superscripts indicating iteration steps, we have

$$\Phi_{i,j}^{(n+1)} = \frac{1}{4}\left(\Phi_{i+1,j}^{(n)} + \Phi_{i-1,j}^{(n)} + \Phi_{i,j+1}^{(n)} + \Phi_{i,j-1}^{(n)} + h^2 f_{i,j}^{(n)}\right). \tag{7.2}$$

In the old FORTRAN-77 scientific programming language, implementing the Jacobi method required the saving of Φ in two different arrays, one corresponding to the n-th iteration, and one corresponding to the $(n+1)$-st iteration. When the iteration was done with a single array, the method was called Gauss-Seidel. In the standard Gauss-Seidel method, the array was updated row-by-row and had the form

$$\Phi_{i,j}^{(n+1)} = \frac{1}{4}\left(\Phi_{i+1,j}^{(n)} + \Phi_{i-1,j}^{(n+1)} + \Phi_{i,j+1}^{(n)} + \Phi_{i,j-1}^{(n+1)} + h^2 f_{i,j}^{(n)}\right). \tag{7.3}$$

The Gauss-Seidel method had the double advantage of requiring less memory and converging more rapidly than the Jacobi method.

A variant of Gauss-Seidel that can also be found in textbooks is called red-black Gauss-Seidel. In this algorithm, the grid is viewed as a checkerboard with red and black squares. An updating

of $\Phi_{i,j}$ is done in two passes: in the first pass, $\Phi_{i,j}$ is updated only on the red squares; in the second pass, only on the black squares. Because of the structure of the Laplacian finite difference operator, when solving $\nabla^2\Phi = 0$ the updated values of Φ on the red squares depend only on the values of Φ on the black squares, and the updated values on the black squares depend only on the red squares.

Translating FORTRAN-77 directly into MATLAB, the Gauss-Seidel method in the standard form could be implemented on a uniform grid with $N-1$ and $M-1$ internal grid points in the x- and y-directions, respectively, as

```
for index2=2:M
   for index1=2:N
        phi(index1,index2)=0.25*(phi(index1+1,index2) ...
                           +phi(index1-1,index2)+phi(index1,index2+1)...
                           +phi(index1,index2-1)+h^2*f(index1,index2);
   end
end
```

Nowadays, however, programming languages such as MATLAB operate on vectors, which are optimized for modern computers, and the explicit loops that used to be the workhorse of FORTRAN-77 are now vectorized.

So to properly implement the Jacobi method, say, in MATLAB, one can predefine the vectors `index1` and `index2` that contain the index numbers of the first and second indices of the matrix variable `phi`. These definitions would be

```
index1=2:N; index2=2:M;
```

where `index1` and `index2` reference the internal grid points of the domain. The variable `phi` is assumed to be known on the boundaries of the domain corresponding to the indices `(1,1)`, `(N+1,1)`, `(1,M+1)`, and `(N+1,M+1)`. The Jacobi method in MATLAB can then be coded in one line as

```
phi(index1,index2)=0.25*(phi(index1+1,index2)+phi(index1-1,index2)...
      +phi(index1,index2+1)+phi(index1,index2-1)+h^2*f(index1,index2); .
```

The red-black Gauss-Seidel method could be implemented in a somewhat similar fashion. One must now predefine the vectors `even1`, `odd1`, `even2`, and `odd2`, with definitions

```
even1=2:2:N; even2=2:2:M;
odd1=3:2:N; odd2=3:2:M; .
```

The red-black Gauss-Seidel method then requires the following four coding lines to implement:

```
phi(even1,even2)=0.25*(phi(even1+1,even2)+phi(even1-1,even2)...
      +phi(even1,even2+1)+phi(even1,even2-1)+h^2*f(even1,even2);
phi(odd1,odd2)=0.25*(phi(odd1+1,odd2)+phi(odd1-1,odd2)...
      +phi(odd1,odd2+1)+phi(odd1,odd2-1)+h^2*f(odd1,odd2);
phi(even1,odd2)=0.25*(phi(even1+1,odd2)+phi(even1-1,odd2)...
      +phi(even1,odd2+1)+phi(even1,odd2-1)+h^2*f(even1,odd2);
phi(odd1,even2)=0.25*(phi(odd1+1,even2)+phi(odd1-1,even2)...
      +phi(odd1,even2+1)+phi(odd1,even2-1)+h^2*f(odd1,even2); .
```

Each iteration of the red-black Gauss-Seidel method will run slower than that of the Jacobi method, and the red-black Gauss-Seidel method will only be useful if the slower iterations are compensated by a faster convergence.

In practice, however, the Jacobi method or the red-black Gauss Seidel method is replaced by the corresponding Successive Over Relaxation method (SOR method). We will illustrate this method using the Jacobi method, though the better approach is to use red-black Gauss-Seidel.

The Jacobi method is first rewritten by adding and subtracting the diagonal term $\Phi_{i,j}^{(n)}$:

$$\Phi_{i,j}^{(n+1)} = \Phi_{i,j}^{(n)} + \frac{1}{4}\left(\Phi_{i+1,j}^{(n)} + \Phi_{i-1,j}^{(n)} + \Phi_{i,j+1}^{(n)} + \Phi_{i,j-1}^{(n)} - 4\Phi_{i,j}^{(n)} + h^2 f_{i,j}^{(n)}\right).$$

In this form, we see that the Jacobi method updates the value of $\Phi_{i,j}$ at each iteration. We can either magnify or diminish this update by introducing a relaxation parameter λ. We have

$$\Phi_{i,j}^{(n+1)} = \Phi_{i,j}^{(n)} + \frac{\lambda}{4}\left(\Phi_{i+1,j}^{(n)} + \Phi_{i-1,j}^{(n)} + \Phi_{i,j+1}^{(n)} + \Phi_{i,j-1}^{(n)} - 4\Phi_{i,j}^{(n)} + h^2 f_{i,j}^{(n)}\right),$$

which can be written more efficiently as

$$\Phi_{i,j}^{(n+1)} = (1-\lambda)\Phi_{i,j}^{(n)} + \frac{\lambda}{4}\left(\Phi_{i+1,j}^{(n)} + \Phi_{i-1,j}^{(n)} + \Phi_{i,j+1}^{(n)} + \Phi_{i,j-1}^{(n)} + h^2 f_{i,j}^{(n)}\right).$$

When used with Gauss-Seidel, a value of λ in the range $1 < \lambda < 2$ can often be used to accelerate convergence of the iteration. When $\lambda > 1$, the modifier *over* in Successive Over Relaxation is apt. When the right-hand-side of the Poisson equation is a nonlinear function of Φ, however, the $\lambda = 1$ Gauss-Seidel method may fail to converge. In this case, it may be reasonable to choose a value of λ less than one, and perhaps a better name for the method would be Successive Under Relaxation. When under relaxing, the convergence of the iteration will of course be slowed. But this is the cost that must sometimes be paid for stability.

7.2 Newton's method for a system of nonlinear equations

A system of nonlinear equations can be solved using a version of the iterative Newton's Method for root finding. Although in practice, Newton's method is often applied to a large nonlinear system, we will illustrate the method here for the simple system of two equations and two unknowns.

Suppose that we want to solve

$$f(x,y) = 0, \quad g(x,y) = 0,$$

for the unknowns x and y. We want to construct two simultaneous sequences $x_0, x_1, x_2, \ldots$ and $y_0, y_1, y_2, \ldots$ that converge to the roots. To construct these sequences, we Taylor series expand $f(x_{n+1}, y_{n+1})$ and $g(x_{n+1}, y_{n+1})$ about the point (x_n, y_n). Using the partial derivatives $f_x = \partial f/\partial x$, $f_y = \partial f/\partial y$, etc., the two-dimensional Taylor series expansions, displaying only the linear terms, are given by

$$f(x_{n+1}, y_{n+1}) = f(x_n, y_n) + (x_{n+1} - x_n) f_x(x_n, y_n) + (y_{n+1} - y_n) f_y(x_n, y_n) + \ldots$$

$$g(x_{n+1}, y_{n+1}) = g(x_n, y_n) + (x_{n+1} - x_n) g_x(x_n, y_n) + (y_{n+1} - y_n) g_y(x_n, y_n) + \ldots.$$

To obtain Newton's method, we take $f(x_{n+1}, y_{n+1}) = 0$, $g(x_{n+1}, y_{n+1}) = 0$, and drop higher-order terms above linear. Although one can then find a system of linear equations for x_{n+1} and y_{n+1}, it is more convenient to define the variables

$$\Delta x_n = x_{n+1} - x_n, \quad \Delta y_n = y_{n+1} - y_n.$$

The iteration equations will then be given by

$$x_{n+1} = x_n + \Delta x_n, \quad y_{n+1} = y_n + \Delta y_n;$$

and the linear equations to be solved for Δx_n and Δy_n are given by

$$\begin{pmatrix} f_x & f_y \\ g_x & g_y \end{pmatrix} \begin{pmatrix} \Delta x_n \\ \Delta y_n \end{pmatrix} = \begin{pmatrix} -f \\ -g \end{pmatrix},$$

where f, g, f_x, f_y, g_x, and g_y are all evaluated at the point (x_n, y_n). The two-dimensional case is easily generalized to n dimensions. The matrix of partial derivatives is called the Jacobian Matrix.

We illustrate Newton's Method by finding numerically the steady state solution of the Lorenz equations, given by

$$\begin{aligned} \sigma(y - x) &= 0, \\ rx - y - xz &= 0, \\ xy - bz &= 0, \end{aligned}$$

where x, y, and z are the unknown variables and σ, r, and b are the known parameters. Here, we have a three-dimensional homogeneous system $f = 0$, $g = 0$, and $h = 0$, with

$$\begin{aligned} f(x, y, z) &= \sigma(y - x), \\ g(x, y, z) &= rx - y - xz, \\ h(x, y, z) &= xy - bz. \end{aligned}$$

The partial derivatives can be computed to be

$$\begin{aligned} f_x &= -\sigma, & f_y &= \sigma, & f_z &= 0, \\ g_x &= r - z, & g_y &= -1, & g_z &= -x, \\ h_x &= y, & h_y &= x, & h_z &= -b. \end{aligned}$$

The iteration equation is therefore

$$\begin{pmatrix} -\sigma & \sigma & 0 \\ r - z_n & -1 & -x_n \\ y_n & x_n & -b \end{pmatrix} \begin{pmatrix} \Delta x_n \\ \Delta y_n \\ \Delta z_n \end{pmatrix} = - \begin{pmatrix} \sigma(y_n - x_n) \\ rx_n - y_n - x_n z_n \\ x_n y_n - b z_n \end{pmatrix},$$

with

$$\begin{aligned} x_{n+1} &= x_n + \Delta x_n, \\ y_{n+1} &= y_n + \Delta y_n, \\ z_{n+1} &= z_n + \Delta z_n. \end{aligned}$$

The sample MATLAB program that solves this system is in the m-file `newton_system.m`.

Chapter 8

Interpolation

Consider the following problem: Given the values of a known function $y = f(x)$ at a sequence of ordered points $x_0, x_1, \ldots, x_n$, find $f(x)$ for arbitrary x. When $x_0 \leq x \leq x_n$, the problem is called interpolation. When $x < x_0$ or $x > x_n$, the problem is called extrapolation.

With $y_i = f(x_i)$, the problem of interpolation is basically one of drawing a smooth curve through the known points $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$. This is not the same problem as drawing a smooth curve that approximates a set of data points that have experimental error. This latter problem is called least-squares approximation, which is considered in the next chapter.

It is possible to interpolate the $n + 1$ known points by a unique polynomial of degree n. When $n = 1$, the polynomial is a linear function; when $n = 2$, the polynomial is a quadratic function. Although low order polynomials are sometimes used when the number of points are few, higher-order polynomial interpolates tend to be unstable, and are not of much practical use.

Here, we will consider the more useful piece-wise polynomial interpolation. The two most used are piecewise linear interpolation, and cubic spline interpolation. The first makes use of linear polynomials, and the second cubic polynomials.

8.1 Piecewise linear interpolation

Here, we use linear polynomials. This is the default interpolation typically used when plotting data.

Suppose the interpolating function is $y = g(x)$, and as previously, there are $n + 1$ points to interpolate. We construct the function $g(x)$ out of n local linear polynomials. We write

$$g(x) = g_i(x), \quad \text{for } x_i \leq x \leq x_{i+1},$$

where

$$g_i(x) = a_i(x - x_i) + b_i,$$

and $i = 0, 1, \ldots, n - 1$.

We now require $y = g_i(x)$ to pass through the endpoints (x_i, y_i) and (x_{i+1}, y_{i+1}). We have

$$\begin{aligned} y_i &= b_i, \\ y_{i+1} &= a_i(x_{i+1} - x_i) + b_i. \end{aligned}$$

Therefore, the coefficients of $g_i(x)$ are determined to be

$$a_i = \frac{y_{i+1} - y_i}{x_{i+1} - x_i}, \quad b_i = y_i.$$

Although piecewise linear interpolation is widely used, particularly in plotting routines, it suffers from a discontinuity in the derivative at each point. This results in a function which may not look smooth if the points are too widely spaced. We next consider a more challenging algorithm that uses cubic polynomials.

8.2 Cubic spline interpolation

The $n+1$ points to be interpolated are again

$$(x_0, y_0), (x_1, y_1), \ldots (x_n, y_n).$$

Here, we use n piecewise cubic polynomials for interpolation,

$$g_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i, \quad i = 0, 1, \ldots, n-1,$$

with the global interpolation function written as

$$g(x) = g_i(x), \quad \text{for } x_i \leq x \leq x_{i+1}.$$

To achieve a smooth interpolation we impose that $g(x)$ and its first and second derivatives are continuous. The requirement that $g(x)$ is continuous (and goes through all $n+1$ points) results in the two constraints

$$g_i(x_i) = y_i, \quad i = 0 \text{ to } n-1, \tag{8.1}$$

$$g_i(x_{i+1}) = y_{i+1}, \quad i = 0 \text{ to } n-1. \tag{8.2}$$

The requirement that $g'(x)$ is continuous results in

$$g_i'(x_{i+1}) = g_{i+1}'(x_{i+1}), \quad i = 0 \text{ to } n-2. \tag{8.3}$$

And the requirement that $g''(x)$ is continuous results in

$$g_i''(x_{i+1}) = g_{i+1}''(x_{i+1}), \quad i = 0 \text{ to } n-2. \tag{8.4}$$

There are n cubic polynomials $g_i(x)$ and each cubic polynomial has four free coefficients; there are therefore a total of $4n$ unknown coefficients. The number of constraining equations from (8.1)-(8.4) is $2n + 2(n-1) = 4n - 2$. With $4n-2$ constraints and $4n$ unknowns, two more conditions are required for a unique solution. These are usually chosen to be extra conditions on the first $g_0(x)$ and last $g_{n-1}(x)$ polynomials. We will discuss these extra conditions later.

We now proceed to determine equations for the unknown coefficients of the cubic polynomials. The polynomials and their first two derivatives are given by

$$g_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i, \tag{8.5}$$

$$g_i'(x) = 3a_i(x - x_i)^2 + 2b_i(x - x_i) + c_i, \tag{8.6}$$

$$g_i''(x) = 6a_i(x - x_i) + 2b_i. \tag{8.7}$$

We will consider the four conditions (8.1)-(8.4) in turn. From (8.1) and (8.5), we have

$$d_i = y_i, \quad i = 0 \text{ to } n-1, \tag{8.8}$$

which directly solves for all of the d-coefficients.

To satisfy (8.2), we first define

$$h_i = x_{i+1} - x_i,$$

and

$$f_i = y_{i+1} - y_i.$$

Now, from (8.2) and (8.5), using (8.8), we obtain the n equations

$$a_i h_i^3 + b_i h_i^2 + c_i h_i = f_i, \quad i = 0 \text{ to } n-1. \tag{8.9}$$

From (8.3) and (8.6) we obtain the $n-1$ equations

$$3a_i h_i^2 + 2b_i h_i + c_i = c_{i+1}, \quad i = 0 \text{ to } n-2. \tag{8.10}$$

From (8.4) and (8.7) we obtain the $n-1$ equations

$$3a_i h_i + b_i = b_{i+1} \quad i = 0 \text{ to } n-2. \tag{8.11}$$

It will be useful to include a definition of the coefficient b_n, which is now missing. (The index of the cubic polynomial coefficients only go up to $n-1$.) We simply extend (8.11) up to $i = n-1$ and so write

$$3a_{n-1} h_{n-1} + b_{n-1} = b_n, \tag{8.12}$$

which can be viewed as a definition of b_n.

We now proceed to eliminate the sets of a- and c-coefficients (with the d-coefficients already eliminated in (8.8)) to find a system of linear equations for the b-coefficients. From (8.11) and (8.12), we can solve for the n a-coefficients to find

$$a_i = \frac{1}{3h_i}\left(b_{i+1} - b_i\right), \quad i = 0 \text{ to } n-1. \tag{8.13}$$

From (8.9), we can solve for the n c-coefficients as follows:

$$\begin{aligned} c_i &= \frac{1}{h_i}\left(f_i - a_i h_i^3 - b_i h_i^2\right) \\ &= \frac{1}{h_i}\left(f_i - \frac{1}{3h_i}\left(b_{i+1} - b_i\right) h_i^3 - b_i h_i^2\right) \\ &= \frac{f_i}{h_i} - \frac{1}{3} h_i \left(b_{i+1} + 2b_i\right), \quad i = 0 \text{ to } n-1. \end{aligned} \tag{8.14}$$

We can now find an equation for the b-coefficients by substituting (8.13) and (8.14) into (8.10):

$$3\left(\frac{1}{3h_i}(b_{i+1} - b_i)\right) h_i^2 + 2b_i h_i + \left(\frac{f_i}{h_i} - \frac{1}{3}h_i(b_{i+1} + 2b_i)\right) = \left(\frac{f_{i+1}}{h_{i+1}} - \frac{1}{3}h_{i+1}(b_{i+2} + 2b_{i+1})\right),$$

which simplifies to

$$\frac{1}{3}h_i b_i + \frac{2}{3}(h_i + h_{i+1})b_{i+1} + \frac{1}{3}h_{i+1}b_{i+2} = \frac{f_{i+1}}{h_{i+1}} - \frac{f_i}{h_i}, \tag{8.15}$$

an equation that is valid for $i = 0$ to $n-2$. Therefore, (8.15) represent $n-1$ equations for the $n+1$ unknown b-coefficients. Accordingly, we write the matrix equation for the b-coefficients, leaving the first and last row absent, as

$$\begin{pmatrix} \dots & \dots & \dots & \dots & \text{missing} & \dots & \dots \\ \frac{1}{3}h_0 & \frac{2}{3}(h_0+h_1) & \frac{1}{3}h_1 & \dots & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & \frac{1}{3}h_{n-2} & \frac{2}{3}(h_{n-2}+h_{n-1}) & \frac{1}{3}h_{n-1} \\ \dots & \dots & \dots & \dots & \text{missing} & \dots & \dots \end{pmatrix} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{n-1} \\ b_n \end{pmatrix}$$

$$= \begin{pmatrix} \text{missing} \\ \frac{f_1}{h_1} - \frac{f_0}{h_0} \\ \vdots \\ \frac{f_{n-1}}{h_{n-1}} - \frac{f_{n-2}}{h_{n-2}} \\ \text{missing} \end{pmatrix}.$$

Once the missing first and last equations are specified, the matrix equation for the b-coefficients can be solved by Gaussian elimination. And once the b-coefficients are determined, the a- and c-coefficients can also be determined from (8.13) and (8.14), with the d-coefficients already known from (8.8). The piecewise cubic polynomials, then, are known and $g(x)$ can be used for interpolation to any value x satisfying $x_0 \le x \le x_n$.

The missing first and last equations can be specified in several ways, and here we show the two ways that are allowed by the MATLAB function spline.m. The first way should be used when the derivative $g'(x)$ is known at the endpoints x_0 and x_n; that is, suppose we know the values of α and β such that

$$g_0'(x_0) = \alpha, \quad g_{n-1}'(x_n) = \beta.$$

From the known value of α, and using (8.6) and (8.14), we have

$$\begin{aligned} \alpha &= c_0 \\ &= \frac{f_0}{h_0} - \frac{1}{3}h_0(b_1 + 2b_0). \end{aligned}$$

Therefore, the missing first equation is determined to be

$$\frac{2}{3}h_0 b_0 + \frac{1}{3}h_0 b_1 = \frac{f_0}{h_0} - \alpha. \tag{8.16}$$

From the known value of β, and using (8.6), (8.13), and (8.14), we have

$$\begin{aligned} \beta &= 3a_{n-1}h_{n-1}^2 + 2b_{n-1}h_{n-1} + c_{n-1} \\ &= 3\left(\frac{1}{3h_{n-1}}(b_n - b_{n-1})\right)h_{n-1}^2 + 2b_{n-1}h_{n-1} + \left(\frac{f_{n-1}}{h_{n-1}} - \frac{1}{3}h_{n-1}(b_n + 2b_{n-1})\right), \end{aligned}$$

which simplifies to

$$\frac{1}{3}h_{n-1}b_{n-1} + \frac{2}{3}h_{n-1}b_n = \beta - \frac{f_{n-1}}{h_{n-1}}, \tag{8.17}$$

to be used as the missing last equation.

The second way of specifying the missing first and last equations is called the *not-a-knot* condition, which assumes that

$$g_0(x) = g_1(x), \quad g_{n-2}(x) = g_{n-1}(x).$$

Considering the first of these equations, from (8.5) we have

$$\begin{aligned} a_0(x - x_0)^3 + b_0(x - x_0)^2 + c_0(x - x_0) + d_0 \\ = a_1(x - x_1)^3 + b_1(x - x_1)^2 + c_1(x - x_1) + d_1. \end{aligned}$$

Now two cubic polynomials can be proven to be identical if at some value of x, the polynomials and their first three derivatives are identical. Our conditions of continuity at $x = x_1$ already require that at this value of x these two polynomials and their first two derivatives are identical. The polynomials themselves will be identical, then, if their third derivatives are also identical at $x = x_1$, or if

$$a_0 = a_1.$$

From (8.13), we have

$$\frac{1}{3h_0}(b_1 - b_0) = \frac{1}{3h_1}(b_2 - b_1),$$

or after simplification

$$h_1 b_0 - (h_0 + h_1)b_1 + h_0 b_2 = 0, \tag{8.18}$$

which provides us our missing first equation. A similar argument at $x = x_{n-1}$ also provides us with our last equation,

$$h_{n-1}b_{n-2} - (h_{n-2} + h_{n-1})b_{n-1} + h_{n-2}b_n = 0. \tag{8.19}$$

The MATLAB subroutines spline.m and ppval.m can be used for cubic spline interpolation (see also interp1.m). I will illustrate these routines in class and post sample code on the course web site.

8.3 Multidimensional interpolation

Suppose we are interpolating the value of a function of two variables,

$$z = f(x, y).$$

The known values are given by

$$z_{ij} = f(x_i, y_j),$$

with $i = 0, 1, \ldots, n$ and $j = 0, 1, \ldots, n$. Note that the (x, y) points at which $f(x, y)$ are known lie on a grid in the $x - y$ plane.

Let $z = g(x, y)$ be the interpolating function, satisfying $z_{ij} = g(x_i, y_j)$. A two-dimensional interpolation to find the value of g at the point (x, y) may be done by first performing $n + 1$ one-dimensional interpolations in y to find the value of g at the $n + 1$ points (x_0, y), (x_1, y), $\ldots$, (x_n, y), followed by a single one-dimensional interpolation in x to find the value of g at (x, y).

In other words, two-dimensional interpolation on a grid of dimension $(n + 1) \times (n + 1)$ is done by first performing $n + 1$ one-dimensional interpolations to the value y followed by a single one-dimensional interpolation to the value x. Two-dimensional interpolation can be generalized to higher dimensions. The MATLAB functions to perform two- and three-dimensional interpolation are interp2.m and interp3.m.

Chapter 9

Least-squares approximation

The method of least-squares is commonly used to fit a parameterized curve to experimental data. In general, the fitting curve is not expected to pass through the data points, making this problem substantially different from the one of interpolation.

We consider here only the most common situation: the fitting of a straight line through data with the same experimental error for all the data points. We assume that the data to be fitted are given by (x_i, y_i), with $i = 1$ to n.

We write for the fitting curve

$$y(x) = \alpha x + \beta.$$

The distance r_i from the data point (x_i, y_i) and the fitting curve is given by

$$\begin{aligned} r_i &= y_i - y(x_i) \\ &= y_i - (\alpha x_i + \beta). \end{aligned}$$

A least-squares fit minimizes the sum of the squares of the r_i's. This minimum can be shown to result in the most probable values of α and β.

We define

$$\begin{aligned} \rho &= \sum_{i=1}^{n} r_i^2 \\ &= \sum_{i=1}^{n} \big(y_i - (\alpha x_i + \beta)\big)^2. \end{aligned}$$

To minimize ρ with respect to α and β, we solve

$$\frac{\partial \rho}{\partial \alpha} = 0, \quad \frac{\partial \rho}{\partial \beta} = 0.$$

Taking the partial derivatives, we have

$$\begin{aligned} \frac{\partial \rho}{\partial \alpha} &= \sum_{i=1}^{n} 2(-x_i)\big(y_i - (\alpha x_i + \beta)\big) = 0, \\ \frac{\partial \rho}{\partial \beta} &= \sum_{i=1}^{n} 2(-1)\big(y_i - (\alpha x_i + \beta)\big) = 0. \end{aligned}$$

These equations form a system of two linear equations in the two unknowns α and β, which is evident when rewritten in the form

$$\begin{aligned} \alpha \sum_{i=1}^{n} x_i^2 + \beta \sum_{i=1}^{n} x_i &= \sum_{i=1}^{n} x_i y_i, \\ \alpha \sum_{i=1}^{n} x_i + \beta n &= \sum_{i=1}^{n} y_i. \end{aligned}$$

These equations can be solved numerically, and MATLAB provides a built-in subroutine called polyfit.m. With the data in the vectors x and y, the MATLAB call

```
p = polyfit(x,y,1);
```

returns the values $p(1) = \alpha$ and $p(2) = \beta$, which can then be used to draw the fitting line.

Part II

Dynamical systems and chaos

The second part of this course will include a discussion of dynamical systems theory and chaos. Our main vehicle for this discussion will be the motion of the one-dimensional driven, damped pendulum.

Chapter 10

The simple pendulum

We first consider the simple pendulum shown in Fig. 10.1. A mass is attached to a massless rigid rod, and is constrained to move along an arc of a circle centered at the pivot point. Suppose l is the fixed length of the connecting rod, and θ is the angle it makes with the vertical axis. We will derive the governing equations for the motion of the mass, and an equation which can be solved to determine the period of oscillation.

10.1 Governing equations

The governing equations for the pendulum are derived from Newton's equation, $F = ma$. Because the pendulum is constained to move along an arc, we can write Newton's equation directly for the displacement s of the pendulum along the arc with origin at the bottom and positive direction to the right.

The relevant force on the pendulum is the component of the gravitational force along the arc, and from Fig. 10.1 is seen to be

$$F_g = -mg\sin\theta, \tag{10.1}$$

where the negative sign signifies a force acting along the negative s direction when $0 < \theta < \pi$, and the positive s direction when $-\pi < \theta < 0$.

Newton's equation for the simple pendulum moving along the arc is therefore

$$m\ddot{s} = -mg\sin\theta.$$

Now, the relationship between the arc length s and the angle θ is given by $s = l\theta$, and therefore $\ddot{s} = l\ddot{\theta}$. The simple pendulum equation can then be written in terms of the angle θ as

$$\ddot{\theta} + \omega^2\sin\theta = 0, \tag{10.2}$$

with

$$\omega = \sqrt{g/l}. \tag{10.3}$$

The standard small angle approximation $\sin\theta \approx \theta$ results in the well-known equation for the simple harmonic oscillator,

$$\ddot{\theta} + \omega^2\theta = 0. \tag{10.4}$$

We have derived the equations of motion by supposing that the pendulum is constrained to move along the arc of a circle. Such a constraint is valid provided the pendulum mass is connected to a massless rigid rod. If the rod is replaced by a massless spring, say, then oscillations in the length of the spring can occur. Deriving the equations for this more general physical situation requires considering the vector form of Newton's equation.

With the origin of the coordinate system located at the base of the pendulum, we define the positive x-direction to be pointing down, and the positive y direction to be pointing to the right. The mass is located at the position vector $\mathbf{x} = (x, y)$. Both gravity and the tension force T acts on the mass, and the governing equations are given by

$$\begin{aligned} m\ddot{x} &= mg - T\cos\theta, \\ m\ddot{y} &= -T\sin\theta. \end{aligned}$$

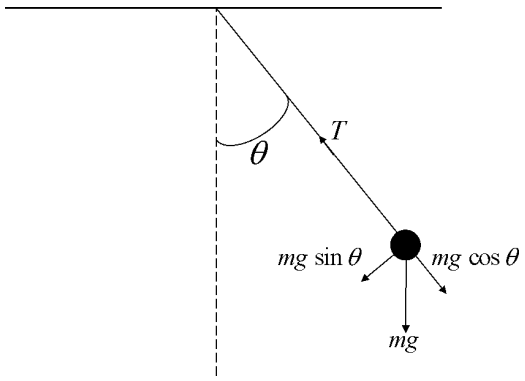


Figure 10.1: *Forces acting on the pendulum.*

It is informative to construct the equivalent governing equations in polar coordinates. To do so, we form the complex coordinate $z = x + iy$, and make use of the polar form for z, given by

$$z = re^{i\theta}.$$

The governing equations then become

$$\frac{d^2}{dt^2}\left(re^{i\theta}\right) = g - \frac{T}{m}e^{i\theta}. \tag{10.5}$$

The second derivative can be computed using the product and chain rules, and one finds

$$\frac{d^2}{dt^2}\left(re^{i\theta}\right) = \left(\left(\ddot{r} - r\dot{\theta}^2\right) + i\left(r\ddot{\theta} + 2\dot{r}\dot{\theta}\right)\right)e^{i\theta}.$$

Dividing both sides of (10.5) by $e^{i\theta}$, we obtain

$$\left(\ddot{r} - r\dot{\theta}^2\right) + i\left(r\ddot{\theta} + 2\dot{r}\dot{\theta}\right) = ge^{-i\theta} - \frac{T}{m}. \tag{10.6}$$

The two governing equations in polar coordinates, then, are determined by equating the real parts and the imaginary parts of (10.6), and we find

$$\ddot{r} - r\dot{\theta}^2 = g\cos\theta - \frac{T}{m}, \tag{10.7}$$

$$r\ddot{\theta} + 2\dot{r}\dot{\theta} = -g\sin\theta. \tag{10.8}$$

If the connector is a rigid rod, as we initially assumed, then $r = l$, $\dot{r} = 0$, and $\ddot{r} = 0$. The first equation (10.7) is then an equation for the tension T in the rod, and the second equation (10.8) reduces to our previously derived (10.2). If the connector is a spring, then Hooke's law may be applied. Suppose the spring constant is k and the unstretched spring has length l. Then

$$T = k(r - l)$$

in (10.7), and a pair of simultaneous second-order equations govern the motion. The stable equilibrium with the mass at rest at the bottom satisfies $\theta = 0$ and $r = l + mg/k$; i.e., the spring is stretched to counterbalance the hanging mass.

10.2 Period of motion

In general, the period of the simple pendulum depends on the amplitude of its motion. For small amplitude oscillations, the simple pendulum equation (10.2) reduces to the simple harmonic oscillator equation (10.4), and the period becomes independent of amplitude. The general solution of the simple harmonic oscillator equation can be written as

$$\theta(t) = A\cos(\omega t + \varphi).$$

Initial conditions on θ and $\dot{\theta}$ determine A and φ, and by redefining the origin of time, one can always choose $\varphi = 0$ and $A > 0$. With this choice, $A = \theta_m$, the maximum amplitude of the pendulum, and the analytical solution of (10.4) is

$$\theta(t) = \theta_m \cos\omega t,$$

where ω is called the angular frequency of oscillation. The period of motion is related to the angular frequency by

$$T = 2\pi/\omega,$$

and is independent of the amplitude θ_m.

If $\sin\theta \approx \theta$ is no longer a valid approximation, then we need to solve the simple pendulum equation (10.2). We first derive a closed form analytical expression, and then explain how to compute a numerical solution.

10.2.1 Analytical solution

A standard procedure for solving unfamiliar differential equations is to try and determine some combination of variables that is independent of time. Here, we can multiply (10.2) by $\dot{\theta}$ to obtain

$$\dot{\theta}\ddot{\theta} + \omega^2\dot{\theta}\sin\theta = 0. \tag{10.9}$$

Now,

$$\dot{\theta}\ddot{\theta} = \frac{d}{dt}\left(\frac{1}{2}\dot{\theta}^2\right), \quad \dot{\theta}\sin\theta = \frac{d}{dt}\left(-\cos\theta\right),$$

so that (10.9) can be written as

$$\frac{d}{dt}\left(\frac{1}{2}\dot{\theta}^2 - \omega^2\cos\theta\right) = 0. \tag{10.10}$$

The combination of variables in the parenthesis is therefore independent of time and is called an integral of motion. It is also said to be conserved, and (10.10) is called a conservation law. In physics, this integral of motion (multiplied by ml^2) is identified with the energy of the oscillator.

The value of this integral of motion at $t = 0$ is given by $-\omega^2\cos\theta_m$, so the derived conservation law takes the form

$$\frac{1}{2}\dot{\theta}^2 - \omega^2\cos\theta = -\omega^2\cos\theta_m, \tag{10.11}$$

which is a separable first-order differential equation.

We can compute the period of oscillation T as four times the time it takes for the pendulum to go from its initial height to the bottom. During this quarter cycle of oscillation, $d\theta/dt < 0$ so that from (10.11),

$$\frac{d\theta}{dt} = -\sqrt{2}\omega\sqrt{\cos\theta - \cos\theta_m}.$$

After separating and integrating over a quarter period, we have

$$\int_0^{T/4} dt = -\frac{\sqrt{2}}{2\omega}\int_{\theta_m}^{0}\frac{d\theta}{\sqrt{\cos\theta - \cos\theta_m}},$$

or

$$T = \frac{2\sqrt{2}}{\omega} \int_0^{\theta_m} \frac{d\theta}{\sqrt{\cos\theta - \cos\theta_m}}. \tag{10.12}$$

We can transform this equation for the period into a more standard form using a trigonometric identity and a substitution. The trig-identity is the well-known half-angle formula for sin, given by

$$\sin^2(\theta/2) = \frac{1}{2}(1 - \cos\theta).$$

Using this identity, we write

$$\cos\theta = 1 - 2\sin^2(\theta/2), \quad \cos\theta_m = 1 - 2\sin^2(\theta_m/2), \tag{10.13}$$

and substituting (10.13) into (10.12) results in

$$T = \frac{2}{\omega} \int_0^{\theta_m} \frac{d\theta}{\sqrt{\sin^2(\theta_m/2) - \sin^2(\theta/2)}}. \tag{10.14}$$

We now define the constant

$$a = \sin(\theta_m/2), \tag{10.15}$$

and perform the substitution

$$\sin\phi = \frac{1}{a}\sin(\theta/2). \tag{10.16}$$

Developing this substitution, we have

$$\cos\phi\, d\phi = \frac{1}{2a}\cos(\theta/2)\, d\theta. \tag{10.17}$$

Now,

$$\begin{aligned}\cos(\theta/2) &= \sqrt{1 - \sin^2(\theta/2)} \\ &= \sqrt{1 - a^2\sin^2\phi},\end{aligned}$$

so that (10.17) can be solved for $d\theta$:

$$d\theta = \frac{2a\cos\phi}{\sqrt{1 - a^2\sin^2\phi}} d\phi. \tag{10.18}$$

Using (10.16), the domain of integration $\theta \in [0, \theta_m]$ transforms into $\phi \in [0, \pi/2]$. Therefore, substituting (10.16) and (10.18) into (10.14) results in the standard equation for the period given by

$$T = \frac{4}{\omega} \int_0^{\pi/2} \frac{d\phi}{\sqrt{1 - a^2\sin^2\phi}}, \tag{10.19}$$

with a given by (10.15). The integral $T = T(a)$ is called the complete elliptic integral of the first kind.

For small amplitudes of oscillation, it is possible to determine the leading-order dependence of the period on θ_m. Now a is given by (10.15), so that a Taylor series to leading order yields

$$a^2 = \frac{1}{4}\theta_m^2 + \mathrm{O}(\theta_m^4).$$

Similarly, the Taylor series expansion of the integrand of (10.19) is given by

$$\frac{1}{\sqrt{1-a^2\sin^2\phi}} = 1 + \frac{1}{2}a^2\sin^2\phi + \mathrm{O}(a^4)$$
$$= 1 + \frac{1}{8}\theta_m^2\sin^2\phi + \mathrm{O}(\theta_m^4).$$

Therefore, from (10.19),

$$T = \frac{4}{\omega}\int_0^{\pi/2} d\phi \left(1 + \frac{1}{8}\theta_m^2\sin^2\phi\right) + \mathrm{O}(\theta_m^4).$$

Using

$$\int_0^{\pi/2} d\phi \sin^2\phi = \frac{\pi}{4},$$

we have

$$T = \frac{2\pi}{\omega}\left(1 + \frac{\theta_m^2}{16}\right) + \mathrm{O}(\theta_m^4). \tag{10.20}$$

10.2.2 Numerical solution

We discuss here two methods for computing the period of the pendulum $T = T(\theta_m)$ as a function of the maximum amplitude. The period can be found in units of ω^{-1}; that is, we compute ωT.

The first method makes use of our analytical work and performs a numerical integration of (10.19). Algorithms for numerical integration are discussed in Chapter 3. In particular, use can be made of adaptive Simpson's quadrature, implemented in the MATLAB function `quad.m`.

The second method, which is just as reasonable, solves the differential equation (10.2) directly. Nondimensionalizing using $\tau = \omega t$, equation (10.2) becomes

$$\frac{d^2\theta}{d\tau^2} + \sin\theta = 0. \tag{10.21}$$

To solve (10.21), we write this second-order equation as the system of two first-order equations

$$\frac{d\theta}{d\tau} = u,$$
$$\frac{du}{d\tau} = -\sin\theta,$$

with initial conditions $\theta(0) = \theta_m$ and $u(0) = 0$. We then determine the (dimensionless) time required for the pendulum to move to the position $\theta = 0$: this time will be equal to one-fourth of the period of motion.

Algorithms for integration of ordinary differential equations are discussed in Chapter 4. In particular, use can be made of a Runge-Kutta (4,5) formula, the Dormand-Prince pair, that is implemented in the MATLAB function `ode45.m`.

Perhaps the simplest way to compute the period is to make use of the Event Location Property of the MATLAB ode solver. Through the `odeset` option, it is possible to instruct `ode45.m` to end the time integration when the event $\theta = 0$ occurs, and to return the time at which this event takes place.

A graph of the dimensionless period ωT versus the amplitude θ_m is shown in Fig. 10.2. For comparison, the low-order analytical result of (10.20) is shown as the dashed line.

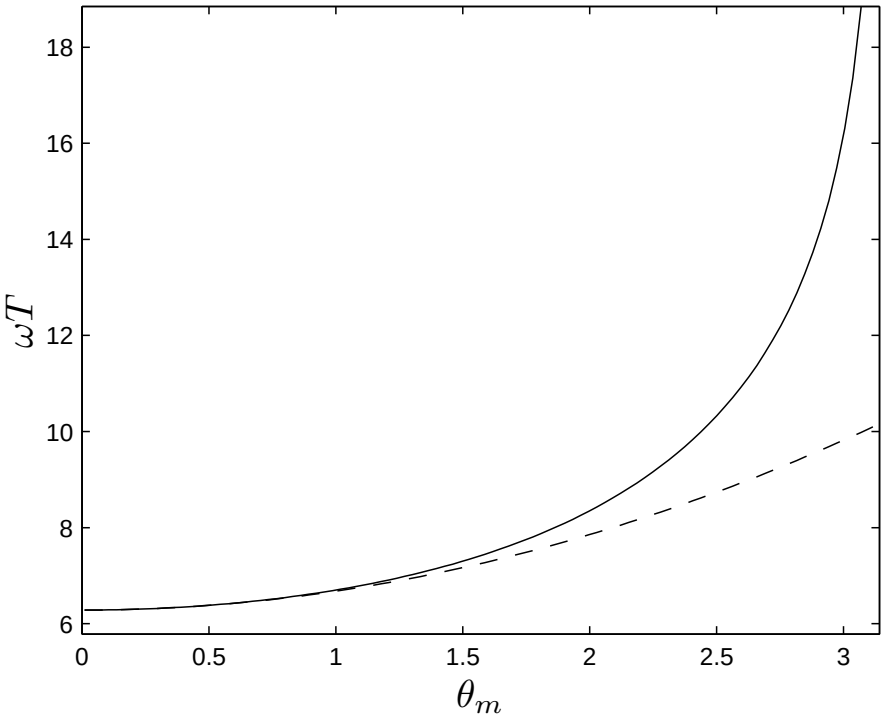


Figure 10.2: *The dimensionless period of the simple pendulum ωT versus amplitude θ_m. The solid line is the numerical result and the dashed line is a low-order approximation.*

Chapter 11

The damped, driven pendulum

The simple pendulum is the mathematical idealization of a frictionless pendulum. We now consider the effects of friction as well as an externally imposed periodic force. The frictional force is modeled as

$$F_f = -\gamma l\dot{\theta},$$

where the frictional force is opposite in sign to the velocity, and thus opposes motion. The positive parameter γ is called the coefficient of friction. The external periodic force is modeled as

$$F_e = F\cos\Omega t,$$

where F is the force's amplitude and Ω is the force's angular frequency. If we also include the gravitational force given by (10.1), Newton's equation can then be written as

$$\ddot{\theta} + \lambda\dot{\theta} + \omega^2\sin\theta = f\cos\Omega t, \tag{11.1}$$

where $\lambda = \gamma/m$, $f = F/ml$, and ω is defined in (10.3). An analytical solution of (11.1) is possible only for small oscillations. Indeed, the damped, driven pendulum can be chaotic when oscillations are large.

11.1 The linear pendulum

11.1.1 Damped pendulum

Here, we exclude the external force, and consider the damped pendulum using the small amplitude approximation $\sin\theta \approx \theta$. The governing equation becomes the linear, second-order, homogeneous differential equation given

$$\ddot{\theta} + \lambda\dot{\theta} + \omega^2\theta = 0, \tag{11.2}$$

which is usually discussed in detail in a first course on differential equations.

The characteristic equation of (11.2) is obtained by the ansatz $\theta(t) = \exp(\alpha t)$, which yields

$$\alpha^2 + \lambda\alpha + \omega^2 = 0, \tag{11.3}$$

with solution

$$\alpha_\pm = -\frac{1}{2}\lambda \pm \frac{1}{2}\sqrt{\lambda^2 - 4\omega^2}. \tag{11.4}$$

For convenience, we define $\beta = \lambda/2$ so that (11.4) becomes

$$\alpha_\pm = -\beta \pm \sqrt{\beta^2 - \omega^2}. \tag{11.5}$$

The discriminant of (11.5) is $\beta^2 - \omega^2$, and its sign determines the nature of the damped oscillations.

The underdamped pendulum satisfies $\beta < \omega$, and we write

$$\alpha_\pm = -\beta \pm i\omega_*,$$

where $\omega_* = \sqrt{\omega^2 - \beta^2}$ and $i = \sqrt{-1}$. In this case, the general solution of (11.2) is a damped oscillation given by

$$\theta(t) = e^{-\beta t}\left(A\cos\omega_* t + B\sin\omega_* t\right).$$

The overdamped pendulum satisfies $\beta > \omega$, and the general solution is an exponential decay and is given by

$$\theta(t) = c_1 e^{\alpha_+ t} + c_2 e^{\alpha_- t},$$

where both α_+ and α_- are negative.

The critically damped pendulum corresponds to the special case when $\beta = \omega$, and with $\alpha_+ = \alpha_- = \alpha < 0$, the general solution is given by

$$\theta(t) = (c_1 + c_2 t)\, e^{\alpha t}.$$

11.1.2 Driven pendulum

Here, we neglect friction but include the external periodic force. The small amplitude approximation results in the governing equation

$$\ddot{\theta} + \omega^2\theta = f\cos\Omega t. \tag{11.6}$$

An interesting solution occurs exactly at resonance, when the external forcing frequency Ω exactly matches the frequency ω of the unforced oscillator. Here, the inhomogeneous term of the differential equation is a solution of the homogeneous equation. With the initial conditions $\theta(0) = \theta_0$ and $\dot{\theta}(0) = 0$, the solution at resonance can be determined to be

$$\theta(t) = \theta_0\cos\omega t + \frac{f}{2\omega}t\sin\omega t,$$

which is a sum of a homogeneous solution (with coefficients determined to satisfy the initial conditions) plus the particular solution. The particular solution is an oscillation with an amplitude that increases linearly with time. Eventually, the small amplitude approximation used to derive (11.6) will become invalid.

An interesting computation solves the pendulum equation at resonance—replacing $\omega^2\theta$ in (11.6) by $\omega^2\sin\theta$—with the pendulum initially at rest at the bottom ($\theta_0 = 0$). What happens to the amplitude of the oscillation after its initial linear increase?

11.1.3 Damped, driven pendulum

Here, we consider both friction and an external periodic force. The small amplitude approximation of (11.1) is given by

$$\ddot{\theta} + \lambda\dot{\theta} + \omega^2\theta = f\cos\Omega t. \tag{11.7}$$

The general solution to (11.7) is determined by adding a particular solution to the general solution of the homogeneous equation. Because of friction, the homogeneous solutions decay to zero leaving at long times only the non-decaying particular solution. To find this particular solution, we note that the complex ode given by

$$\ddot{z} + \lambda\dot{z} + \omega^2 z = f e^{i\Omega t}, \tag{11.8}$$

with $z = x + iy$, represents two real odes given by

$$\ddot{x} + \lambda\dot{x} + \omega^2 x = f\cos\Omega t, \quad \ddot{y} + \lambda\dot{y} + \omega^2 y = f\sin\Omega t,$$

where the first equation is the same as (11.7). We can therefore solve the complex ode (11.8) for $z(t)$, and then take as our solution $\theta(t) = \mathrm{Re}(z)$.

With the ansatz $z_p = Ae^{i\Omega t}$, we have from (11.8)

$$-\Omega^2 A + i\lambda\Omega A + \omega^2 A = f,$$

or solving for A,

$$A = \frac{f}{(\omega^2 - \Omega^2) + i\lambda\Omega}. \tag{11.9}$$

The complex coefficient A determines both the amplitude and the phase of the oscillation. We first rewrite A by multiplying the numerator and denominator by the complex conjugate of the denominator:

$$A = \frac{f\big((\omega^2 - \Omega^2) - i\lambda\Omega\big)}{(\omega^2 - \Omega^2)^2 + \lambda^2\Omega^2}.$$

Now, using the polar form of a complex number, we have

$$(\omega^2 - \Omega^2) - i\lambda\Omega = \sqrt{(\omega^2 - \Omega^2)^2 + \lambda^2\Omega^2}e^{i\phi},$$

where $\tan\phi = \lambda\Omega/(\Omega^2 - \omega^2)$. Therefore, A can be rewritten as

$$A = \frac{fe^{i\phi}}{\sqrt{(\omega^2 - \Omega^2)^2 + \lambda^2\Omega^2}}.$$

With the particular solution given by $\theta(t) = \mathrm{Re}(Ae^{i\omega t})$, we have

$$\theta(t) = \left(\frac{f}{\sqrt{(\omega^2 - \Omega^2)^2 + \lambda^2\Omega^2}}\right) \mathrm{Re}\left(e^{i(\Omega t + \phi)}\right) \tag{11.10}$$

$$= \left(\frac{f}{\sqrt{(\omega^2 - \Omega^2)^2 + \lambda^2\Omega^2}}\right) \cos(\Omega t + \phi). \tag{11.11}$$

The amplitude of the pendulum's oscillation at long times is therefore given by

$$\frac{f}{\sqrt{(\omega^2 - \Omega^2)^2 + \lambda^2\Omega^2}},$$

and the phase shift of the oscillation relative to the external periodic force is given by ϕ.

For example, if the external forcing frequency is tuned to match the frequency of the unforced oscillator, that is, $\Omega = \omega$, then one obtains directly from (11.9) that $A = f/(i\lambda\omega)$, so that the asymptotic solution for $\theta(t)$ is given by

$$\theta(t) = \frac{f}{\lambda\omega}\sin\omega t. \tag{11.12}$$

The oscillator is observed to be $\pi/2$ out of phase with the external force, or in other words, the velocity of the oscillator, not the position, is in phase with the force.

The solution given by (11.12) shows that large amplitude oscillations can result by either increasing f, or decreasing λ or ω. As the amplitude of oscillation becomes large, the small amplitude approximation $\sin\theta \approx \theta$ may become inaccurate and the true pendulum solution may diverge from (11.12).

11.2 The nonlinear pendulum

As we already eluded, the fully nonlinear damped, driven pendulum can become chaotic. To study (11.1) numerically, or for that matter any other equation, the number of free parameters should be reduced to a minimum. This usually means that the governing equations should be nondimensionalized, and the dimensional parameters should be grouped into a minimum number of dimensionless parameters. How many dimensionless parameters will there be? The answer to this question is called the Buckingham Π Theorem.

The Buckingham Π Theorem: *If an equation involves n dimensional parameters that are specified in terms of k independent units, then the equation can be nondimensionalized to one involving $n-k$ dimensionless parameters.*

Now, the damped, driven pendulum equation (11.1) contains four dimensional parameters, λ, f, ω, and Ω, and has a single independent unit, namely time. Therefore, this equation can be nondimensionalized to an equation with only three dimensionless parameters. Namely, we nondimensionalize time using one of the dimensional parameters. Here, we choose ω, with units of inverse time, and write

$$\tau = \omega t,$$

where τ is now the dimensionless time. The damped, driven pendulum equation (11.1) therefore nondimensionalizes to

$$\frac{d^2\theta}{d\tau^2} + \left(\frac{\lambda}{\omega}\right)\frac{d\theta}{d\tau} + \sin\theta = \left(\frac{f}{\omega^2}\right)\cos\left(\left(\frac{\Omega}{\omega}\right)\tau\right), \tag{11.13}$$

and the remaining three dimensionless groupings of parameters are evidently

$$\frac{\lambda}{\omega}, \quad \frac{f}{\omega^2}, \quad \frac{\Omega}{\omega}.$$

We may give these three dimensionless groupings new names. Rather than introduce even more named parameters into the problem, I will now call the dimensionless time t, and reuse some of the other parameter names, with the understanding that the damped, driven pendulum equation that we will now study numerically is dimensionless. We will therefore study the equation

$$\ddot{\theta} + \frac{1}{q}\dot{\theta} + \sin\theta = f\cos\omega t, \tag{11.14}$$

with the now dimensionless parameters named q, f and ω.

Equation (11.14) is called a non-autonomous equation. For a differential equation to be called autonomous, the independent variable t must not appear explicitly. It is possible to write this second-order non-autonomous differential equation as a system of three first-order autonomous equations by introducing the dependent variable $\psi = \omega t$. We therefore have

$$\begin{aligned} \dot{\theta} &= u, \\ \dot{u} &= -\frac{1}{q}u - \sin\theta + f\cos\psi, \\ \dot{\psi} &= \omega. \end{aligned} \tag{11.15}$$

The necessary conditions for an autonomous system of differential equations to admit chaotic solutions are (1) the system has at least three independent dynamical variables, and; (2) the system contains at least one nonlinear coupling. Here, we see that the damped, driven pendulum

equation satisfies these conditions, where the three independent dynamical variables are θ, u and ψ, and there are two nonlinear couplings, $\sin\theta$ and $\cos\psi$, where we already know that the first nonlinear coupling is required for chaotic solutions.

But what exactly is chaos? What we are considering here is called deterministic chaos, that is chaotic solutions to deterministic equations such as a non-stochastic differential equation. Although there is no definitive definition of chaos, perhaps its most important characteristic is a solution's sensitivity to initial conditions. A small change in initial conditions can lead to a large deviation in a solution's behavior. A solution's sensitivity to initial conditions has been called the Butterfly Effect, where the image of a butterfly appeared in the title of a talk that one of the founders of the field, Edward Lorenz, gave in 1972: "Does the flap of a butterfly's wings in Brazil set off a tornado in Texas?"

We can easily observe that the small amplitude approximation of (11.14) can not admit chaotic solutions. Suppose we consider two solutions $\theta_1(t)$ and $\theta_2(t)$ to the approximate equations, these two solutions differing only in their initial conditions. We therefore have

$$\ddot{\theta}_1 + \frac{1}{q}\dot{\theta}_1 + \theta_1 = f\cos\omega t,$$
$$\ddot{\theta}_2 + \frac{1}{q}\dot{\theta}_2 + \theta_2 = f\cos\omega t.$$

If we define $\delta = \theta_2 - \theta_1$, then the equation satisfied by $\delta = \delta(t)$ is given by

$$\ddot{\delta} + \frac{1}{q}\dot{\delta} + \delta = 0,$$

which is the undriven, damped pendulum equation. Therefore, $\delta(t) \to 0$ for large times, and the solution for θ_2 and θ_1 eventually converge, despite different initial conditions. In other words, there is no sensitivity to initial conditions in the solution. Only for large amplitudes where the approximation $\sin\theta \approx \theta$ becomes invalid, are chaotic solutions possible.

We will next learn some of the concepts and tools required for a numerical exploration of chaos in dynamical systems.

Chapter 12

Concepts and tools

We introduce here the concepts and tools that are useful in studying nonlinear dynamical systems.

12.1 Fixed points and linear stability analysis

Consider the one-dimensional differential equation for $x = x(t)$ given by

$$\dot{x} = f(x). \tag{12.1}$$

We say that x_* is a fixed point, or equilibrium point, of (12.1) if $f(x_*) = 0$, so that at a fixed point, $\dot{x} = 0$. The name *fixed point* is apt since the solution to (12.1) with initial condition $x(0) = x_*$ is fixed at $x(t) = x_*$ for all time t.

A fixed point, however, can be stable or unstable. A fixed point is said to be stable if a small perturbation decays in time; it is said to be unstable if a small perturbation grows in time.

We can determine stability by a linear analysis. Let $x = x_* + \epsilon(t)$, where ϵ represents a small perturbation of the solution from the fixed point x_*. Because x_* is a constant, $\dot{x} = \dot{\epsilon}$; and because x_* is a fixed point, $f(x_*) = 0$. Taylor series expanding about $\epsilon = 0$, we have

$$\begin{aligned}\dot{\epsilon} &= f(x_* + \epsilon) \\ &= f(x_*) + \epsilon f'(x_*) + \dots \\ &= \epsilon f'(x_*) + \dots .\end{aligned}$$

The omitted terms in the Taylor series expansion are proportional to ϵ^2, and can be made negligible—at least over a short time interval—by taking $\epsilon(0)$ small. The differential equation to be considered, $\dot{\epsilon} = f'(x_*)\epsilon$, is therefore linear, and has the solution

$$\epsilon(t) = \epsilon(0) e^{f'(x_*)t}.$$

The perturbation of the fixed point solution $x(t) = x_*$ thus decays or grows exponentially depending on the sign of $f'(x_*)$. The stability condition on x_* is therefore

$$x_* \quad \text{is} \quad \begin{cases} \text{a stable fixed point if} & f'(x_*) < 0, \\ \text{an unstable fixed point if} & f'(x_*) > 0. \end{cases}$$

For the special case $f'(x_*) = 0$, we say the fixed point is marginally stable. We will see that bifurcations usually occur at parameter values where fixed points become marginally stable.

Difference equations, or maps, may be similarly analyzed. Consider the one-dimensional map given by

$$x_{n+1} = f(x_n). \tag{12.2}$$

We say that $x_n = x_*$ is a fixed point of the map if $x_* = f(x_*)$. The stability of this fixed point can then be determined by writing $x_n = x_* + \epsilon_n$ so that (12.2) becomes

$$\begin{aligned}x_* + \epsilon_{n+1} &= f(x_* + \epsilon_n) \\ &= f(x_*) + \epsilon_n f'(x_*) + \dots \\ &= x_* + \epsilon_n f'(x_*) + \dots .\end{aligned}$$

Therefore, to leading-order in ϵ,

$$\left|\frac{\epsilon_{n+1}}{\epsilon_n}\right| = \left|f'(x_*)\right|,$$

and the stability condition on x_* for a one-dimensional map is

$$x_* \quad \text{is} \quad \begin{cases} \text{a stable fixed point if} & |f'(x_*)| < 1, \\ \text{an unstable fixed point if} & |f'(x_*)| > 1. \end{cases}$$

Here, marginal stability occurs when $|f'(x_*)| = 1$, and bifurcations usually occur at parameter values where the fixed point becomes marginally stable. If $f'(x_*) = 0$, then the fixed point is called superstable. Perturbations to a superstable fixed point decay especially fast, making numerical calculations at superstable fixed points more rapidly convergent.

The tools of fixed point and linear stability analysis are also applicable to higher-order systems of equations. Consider the two-dimensional system of differential equations given by

$$\dot{x} = f(x, y), \quad \dot{y} = g(x, y). \tag{12.3}$$

The point (x_*, y_*) is said to be a fixed point of (12.3) if $f(x_*, y_*) = 0$ and $g(x_*, y_*) = 0$. Again, the local stability of a fixed point can be determined by a linear analysis. We let $x(t) = x_* + \epsilon(t)$ and $y(t) = y_* + \delta(t)$, where ϵ and δ are small independent perturbations from the fixed point. Making use of the two dimensional Taylor series of $f(x, y)$ and $g(x, y)$ about the fixed point, or equivalently about $(\epsilon, \delta) = (0, 0)$, we have

$$\begin{aligned}
\dot{\epsilon} &= f(x_* + \epsilon, y_* + \delta) \\
&= f_* + \epsilon \frac{\partial f_*}{\partial x} + \delta \frac{\partial f_*}{\partial y} + \ldots \\
&= \epsilon \frac{\partial f_*}{\partial x} + \delta \frac{\partial f_*}{\partial y} + \ldots, \\
\dot{\delta} &= g(x_* + \epsilon, y_* + \delta) \\
&= g_* + \epsilon \frac{\partial g_*}{\partial x} + \delta \frac{\partial g_*}{\partial y} + \ldots \\
&= \epsilon \frac{\partial g_*}{\partial x} + \delta \frac{\partial g_*}{\partial y} + \ldots,
\end{aligned}$$

where in the Taylor series f_*, g_* and the similarly marked partial derivatives all denote functions evaluated at the fixed point. Neglecting higher-order terms in the Taylor series, we thus have a system of odes for the perturbation, given in matrix form as

$$\frac{d}{dt}\begin{pmatrix} \epsilon \\ \delta \end{pmatrix} = \begin{pmatrix} \partial f_*/\partial x & \partial f_*/\partial y \\ \partial g_*/\partial x & \partial g_*/\partial y \end{pmatrix}\begin{pmatrix} \epsilon \\ \delta \end{pmatrix}. \tag{12.4}$$

The two-by-two matrix in (12.4) is called the Jacobian matrix at the fixed point. An eigenvalue analysis of the Jacobian matrix will typically yield two eigenvalues λ_1 and λ_2. These eigenvalues may be real and distinct, complex conjugate pairs, or repeated. The fixed point is stable (all perturbations decay exponentially) if both eigenvalues have negative real parts. The fixed point is unstable (some perturbations grow exponentially) if at least one eigenvalue has a positive real part. Fixed points can be further classified as stable or unstable nodes, unstable saddle points, stable or unstable spiral points, or stable or unstable improper nodes.

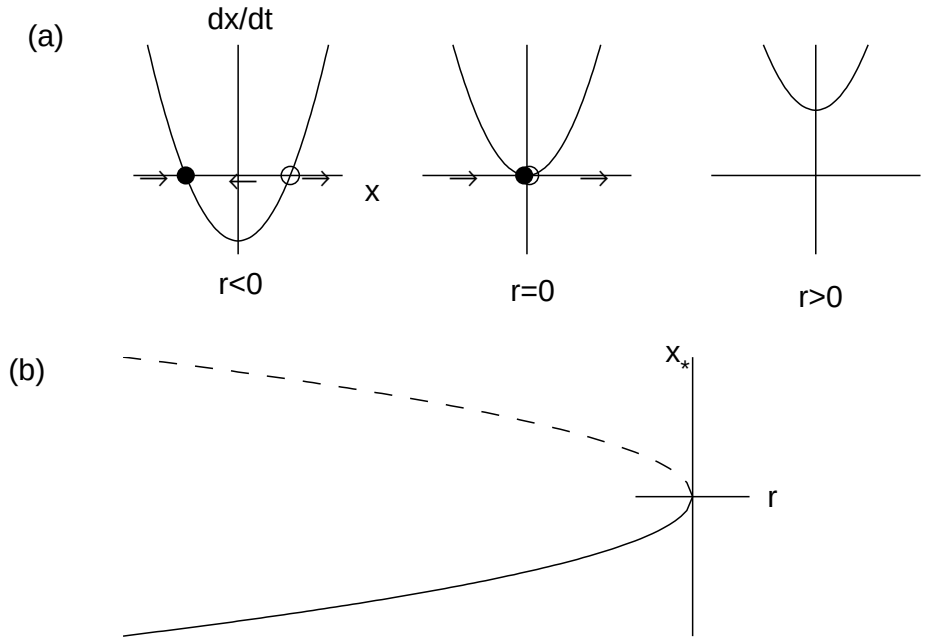


Figure 12.1: *Saddlenode bifurcation. (a) $\dot{x}$ versus x; (b) bifurcation diagram.*

12.2 Bifurcations

For nonlinear systems, small changes in the parameters of the system can result in qualitative changes in the dynamics. These qualitative changes are called bifurcations. Here we consider four classic bifurcations of one-dimensional nonlinear differential equations: the saddle-node bifurcation, the transcritical bifurcation, and the supercritical and subcritical pitchfork bifurcations. The differential equation that we will consider is in general written as

$$\dot{x} = f_r(x),$$

where the subscript r represents a parameter that results in a bifurcation when varied across zero. The simplest differential equations that exhibit these bifurcations are called the *normal forms*, and correspond to a local analysis (i.e., Taylor series expansion) of more general differential equations around the fixed point, together with a possible rescaling of x.

12.2.1 Saddle-node bifurcation

The saddle-node bifurcation results in fixed points being created or destroyed. The normal form for a saddle-node bifurcation is given by

$$\dot{x} = r + x^2.$$

The fixed points are $x_* = \pm\sqrt{-r}$. Clearly, two real fixed points exist when $r < 0$ and no real fixed points exist when $r > 0$. The stability of the fixed points when $r < 0$ are determined by the

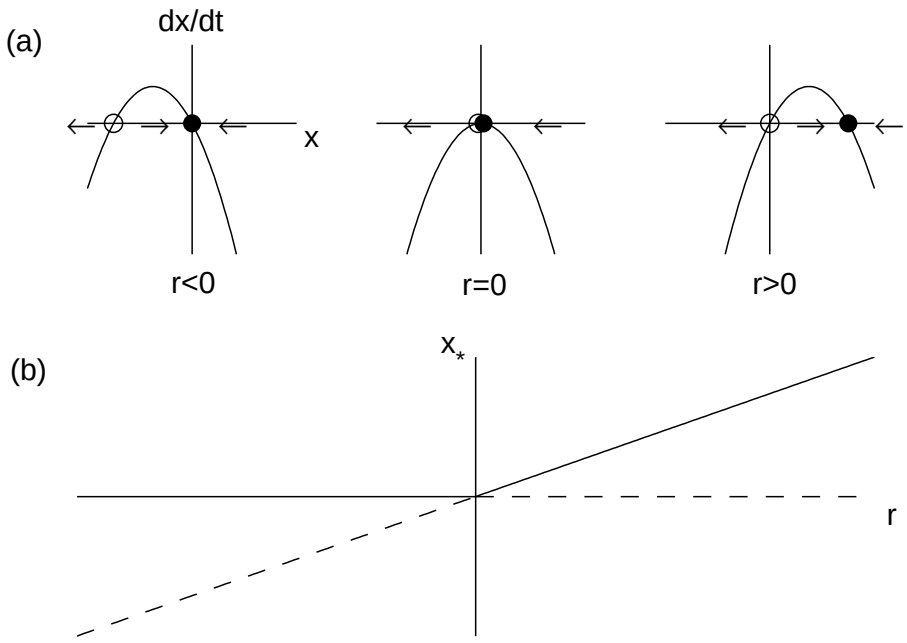


Figure 12.2: *Transcritical bifurcation. (a) $\dot{x}$ versus x; (b) bifurcation diagram.*

derivative of $f(x) = r + x^2$, given by $f'(x) = 2x$. Therefore, the negative fixed point is stable and the positive fixed point is unstable.

We can illustrate this bifurcation. First, in Fig. 12.1(a), we plot $\dot{x}$ versus x for the three parameter values corresponding to $r < 0$, $r = 0$ and $r > 0$. The values at which $\dot{x} = 0$ correspond to the fixed points, and arrows are drawn indicating how the solution $x(t)$ evolves (to the right if $\dot{x} > 0$ and to the left if $\dot{x} < 0$). The stable fixed point is indicated by a filled circle and the unstable fixed point by an open circle. Note that when $r = 0$, solutions converge to the origin from the left, but diverge from the origin on the right.

Second, we plot the standard bifurcation diagram in Fig. 12.1(b), where the fixed point x_* is plotted versus the bifurcation parameter r. As is the custom, the stable fixed point is denoted by a solid line and the unstable fixed point by a dashed line. Note that the two fixed points collide and annihilate at $r = 0$, and there are no fixed points for $r > 0$.

12.2.2 Transcritical bifurcation

A transcritical bifurcation occurs when there is an exchange of stabilities between two fixed points. The normal form for a transcritical bifurcation is given by

$$\dot{x} = rx - x^2.$$

The fixed points are $x_* = 0$ and $x_* = r$. The derivative of the right-hand-side is $f'(x) = r - 2x$, so that $f'(0) = r$ and $f'(r) = -r$. Therefore, for $r < 0$, $x_* = 0$ is stable and $x_* = r$ is unstable, while for $r > 0$, $x_* = r$ is stable and $x_* = 0$ is unstable. The two fixed points thus exchange stability as r passes through zero. The transcritical bifurcation is illustrated in Fig. 12.2.

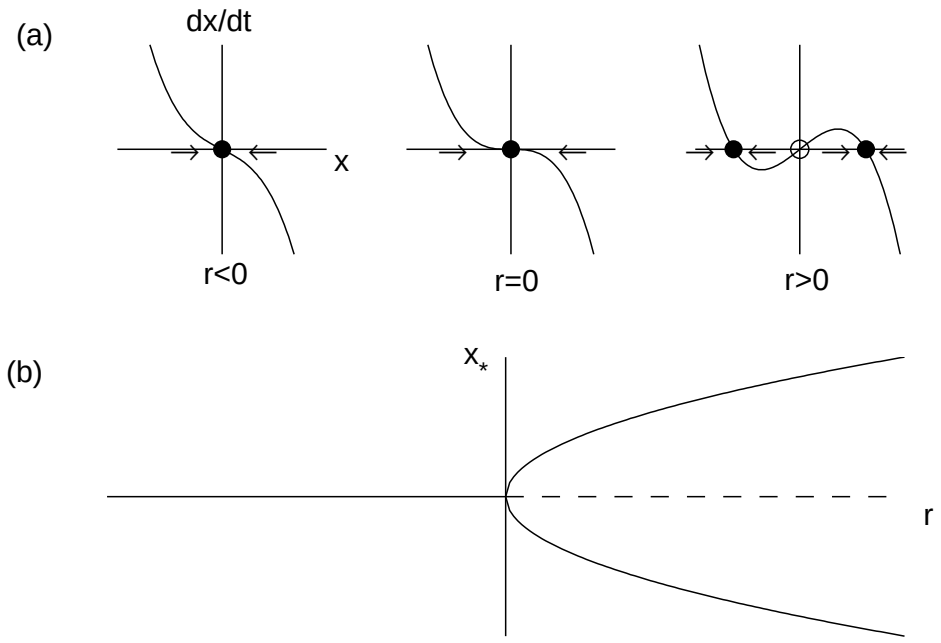


Figure 12.3: *Supercritical pitchfork bifurcation. (a) $\dot{x}$ versus x; (b) bifurcation diagram.*

12.2.3 Pitchfork bifurcations

The pitchfork bifurcations occur when one fixed point becomes three at the bifurcation point. Pitchfork bifurcations are usually associated with the physical phenomena called symmetry breaking. Pitchfork bifurcations come in two types. In the supercritical pitchfork bifurcation, the stability of the original fixed point changes from stable to unstable and a new pair of stable fixed points are created above (super-) the bifurcation point. In the subcritical bifurcation, the stability of the original fixed point again changes from stable to unstable but a new pair of now unstable fixed points are created below (sub-) the bifurcation point.

Supercritical pitchfork bifurcation

The normal form for the supercritical pitchfork bifurcation is given by

$$\dot{x} = rx - x^3. \tag{12.5}$$

Note that the linear term results in exponential growth when $r > 0$ and the nonlinear term stabilizes this growth. The fixed points are $x_* = 0$ and $x_* = \pm\sqrt{r}$, the latter fixed points existing only when $r > 0$. The derivative of f is $f'(x) = r - 3x^2$ so that $f'(0) = r$ and $f'(\pm\sqrt{r}) = -2r$. Therefore, the fixed point $x_* = 0$ is stable for $r < 0$ and unstable for $r > 0$ while the fixed points $x = \pm\sqrt{r}$ exist and are stable for $r > 0$. Notice that the fixed point $x_* = 0$ becomes unstable as r crosses zero and two new stable fixed points $x_* = \pm\sqrt{r}$ are born. The supercritical pitchfork bifurcation is illustrated in Fig. 12.3.

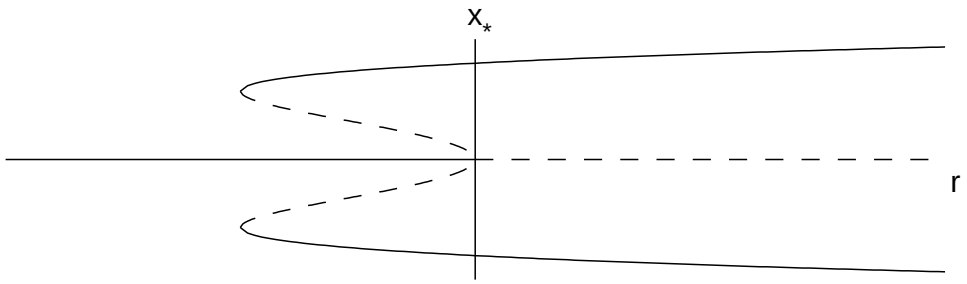


Figure 12.4: *Subcritical pitchfork bifurcation diagram.*

The pitchfork bifurcation illustrates the physics of symmetry breaking. The differential equation (12.5) is invariant under the transformation $x \to -x$. Fixed point solutions of this equation that obey this same symmetry are called symmetric; fixed points that do not are called asymmetric. Here, $x_* = 0$ is the symmetric fixed point and $x = \pm\sqrt{r}$ are asymmetric. Asymmetric fixed points always occur in pairs, and mirror each others stability characteristics. Only the initial conditions determine which asymmetric fixed point is asymptotically attained.

Subcritical pitchfork bifurcation

In the subcritical case, the cubic term is destabilizing. The normal form (to order x^3) is

$$\dot{x} = rx + x^3.$$

The fixed points are $x_* = 0$ and $x_* = \pm\sqrt{-r}$, the latter fixed points existing only when $r \leq 0$. The derivative of the right-hand-side is $f'(x) = r + 3x^2$ so that $f'(0) = r$ and $f'(\pm\sqrt{-r}) = -2r$. Therefore, the fixed point $x_* = 0$ is stable for $r < 0$ and unstable for $r > 0$ while the fixed points $x = \pm\sqrt{-r}$ exist and are unstable for $r < 0$. There are no stable fixed points when $r > 0$.

The absence of stable fixed points for $r > 0$ indicates that the neglect of terms of higher-order in x than x^3 in the normal form may be unwarranted. Keeping to the intrinsic symmetry of the equations (only odd powers of x so that the equation is invariant when $x \to -x$) we can add a stabilizing nonlinear term proportional to x^5. The extended normal form (to order x^5) is

$$\dot{x} = rx + x^3 - x^5,$$

and is somewhat more difficult to analyze. The fixed points are solutions of

$$x(r + x^2 - x^4) = 0.$$

The fixed point $x_* = 0$ exists for all r, and four additional fixed points can be found from the solutions of the quadratic equation in x^2:

$$x_* = \pm\sqrt{\frac{1}{2}(1 \pm \sqrt{1+4r})}.$$

These fixed points exist only if x_* is real. Clearly, for the inner square-root to be real, $r \geq -1/4$. Also observe that $1 - \sqrt{1+4r}$ becomes negative for $r > 0$. We thus have three intervals in r to

consider, and these regions and their fixed points are

$$\begin{aligned}
r < -\frac{1}{4}: &\quad x_* = 0 \quad \text{(one fixed point)};\\
-\frac{1}{4} < r < 0: &\quad x_* = 0, \quad x_* = \pm\sqrt{\frac{1}{2}(1 \pm \sqrt{1+4r})} \quad \text{(five fixed points)};\\
r > 0: &\quad x_* = 0, \quad x_* = \pm\sqrt{\frac{1}{2}(1 + \sqrt{1+4r})} \quad \text{(three fixed points)}.
\end{aligned}$$

Stability is determined from $f'(x) = r + 3x^2 - 5x^4$. Now, $f'(0) = r$ so $x_* = 0$ is stable for $r < 0$ and unstable for $r > 0$. The calculation for the other four roots can be simplified by noting that x_* satisfies $r + x_*^2 - x_*^4 = 0$, or $x_*^4 = r + x_*^2$. Therefore,

$$\begin{aligned}
f'(x_*) &= r + 3x_*^2 - 5x_*^4\\
&= r + 3x_*^2 - 5(r + x_*^2)\\
&= -4r - 2x_*^2\\
&= -2(2r + x_*^2).
\end{aligned}$$

With $x_*^2 = \frac{1}{2}(1 \pm \sqrt{1+4r})$, we have

$$\begin{aligned}
f'(x_*) &= -2\left(2r + \frac{1}{2}(1 \pm \sqrt{1+4r})\right)\\
&= -\left((1+4r) \pm \sqrt{1+4r}\right)\\
&= -\sqrt{1+4r}\left(\sqrt{1+4r} \pm 1\right).
\end{aligned}$$

Clearly, the plus root is always stable since $f'(x_*) < 0$. The minus root exists only for $-\frac{1}{4} < r < 0$ and is unstable since $f'(x_*) > 0$. We summarize the stability of the various fixed points:

$$\begin{aligned}
r < -\frac{1}{4}: &\quad x_* = 0 \quad \text{(stable)};\\
-\frac{1}{4} < r < 0: &\quad x_* = 0, \quad \text{(stable)}\\
&\quad x_* = \pm\sqrt{\frac{1}{2}(1 + \sqrt{1+4r})} \quad \text{(stable)};\\
&\quad x_* = \pm\sqrt{\frac{1}{2}(1 - \sqrt{1+4r})} \quad \text{(unstable)};\\
r > 0: &\quad x_* = 0 \quad \text{(unstable)}\\
&\quad x_* = \pm\sqrt{\frac{1}{2}(1 + \sqrt{1+4r})} \qquad \text{(stable)}.
\end{aligned}$$

The bifurcation diagram is shown in Fig. 12.4. Notice that there in addition to a subcritical pitchfork bifurcation at the origin, there are two symmetric saddlenode bifurcations that occur when $r = -1/4$.

We can imagine what happens to the solution to the ode as r increases from negative values, supposing there is some noise in the system so that $x(t)$ fluctuates around a stable fixed point. For $r < -1/4$, the solution $x(t)$ fluctuates around the stable fixed point $x_* = 0$. As r increases into the range $-1/4 < r < 0$, the solution will remain close to the stable fixed point $x_* = 0$. However, a so-called catastrophe occurs as soon as $r > 0$. The $x_* = 0$ fixed point is lost and the

solution will jump up (or down) to the only remaining fixed point. A similar catastrophe can happen as r decreases from positive values. In this case, the jump occurs as soon as $r < -1/4$. Since the behavior of $x(t)$ is different depending on whether we increase or decrease r, we say that the system exhibits hysteresis. The existence of a subcritical pitchfork bifurcation can be very dangerous in engineering applications since a small change in the physical parameters of a problem can result in a large change in the equilibrium state. Physically, this can result in the collapse of a structure.

12.2.4 Hopf bifurcations

A new type of bifurcation can occur in two dimensions. Suppose there is some control parameter μ. Furthermore, suppose that for $\mu < 0$, a two-dimensional system approaches a fixed point by exponentially-damped oscillations. We know that the Jacobian matrix at the fixed point with $\mu < 0$ will have complex conjugate eigenvalues with negative real parts. Now suppose that when $\mu > 0$ the real parts of the eigenvalues become positive so that the fixed point becomes unstable. This change in stability of the fixed point is called a *Hopf bifurcation*. The Hopf bifurcations also come in two types: a supercritical Hopf bifurcation and a subcritical Hopf bifurcation. For the supercritical Hopf bifurcation, as μ increases slightly above zero, the resulting oscillation around the now unstable fixed point is quickly stabilized at small amplitude, and one obtains a limit cycle. For the subcritical Hopf bifurcation, as μ increases slightly above zero, the limit cycle immediately jumps to large amplitude.

12.3 Phase portraits

The phase space of a dynamical system consists of the independent dynamical variables. For example, the phase space of the damped, driven pendulum equations given by (11.15) is three dimensional and consists of θ, u and ϕ. The unforced equations have only a two-dimensional phase space, consisting of θ and u.

An important feature of paths in phase space is that they can never cross except at a fixed point, which may be stable if all paths go into the point, or unstable if some paths go out. This is a consequence of the uniqueness of the solution to a differential equation. Every point on the phase-space diagram represents a possible initial condition for the equations, and must have a unique trajectory associated with that initial condition.

The simple pendulum is a conservative system, exhibiting a conservation law for energy, and this implies a conservation of phase space area (or volume). The damped pendulum is nonconservative, however, and this implies a shrinking of phase space area (or volume).

Indeed, it is possible to derive a general condition to determine whether phase space volume is conserved or shrinks. We consider here, for convenience, equations in a three-dimensional phase space given by

$$\dot{x} = F(x,y,z), \quad \dot{y} = G(x,y,z), \quad \dot{z} = H(x,y,z).$$

We can consider a small volume of phase space $\Gamma = \Gamma(t)$, given by

$$\Gamma(t) = \Delta x \Delta y \Delta z, \tag{12.6}$$

where

$$\Delta x = x_1(t) - x_0(t), \quad \Delta y = y_1(t) - y_0(t), \quad \Delta z = z_1(t) - z_0(t).$$

The initial phase-space volume at time t represents a box with one corner at the point $(x_0(t), y_0(t), z_0(t))$ and the opposing corner at $(x_1(t), y_1(t), z_1(t))$. This initial phase-space box then evolves over time.

To determine how an edge emanating from (x_0, y_0, z_0) with length Δx evolves, we write using a first-order Taylor series expansion in Δt

$$x_0(t + \Delta t) = x_0(t) + \Delta t F(x_0, y_0, z_0),$$
$$x_1(t + \Delta t) = x_1(t) + \Delta t F(x_1, y_0, z_0).$$

Therefore,

$$\Delta x(t + \Delta t) = \Delta x(t) + \Delta t\big(F(x_1, y_0, z_0) - F(x_0, y_0, z_0)\big). \tag{12.7}$$

Subtracting $\Delta x(t)$ from both sides and dividing by Δt, we have as $\Delta t \to 0$,

$$\frac{d}{dt}(\Delta x) = \big(F(x_1, y_0, z_0) - F(x_0, y_0, z_0)\big).$$

With Δx small, we therefore obtain to first order in Δx,

$$\frac{d}{dt}(\Delta x) = \Delta x \frac{\partial F}{\partial x},$$

where the partial derivative is evaluated at the point $(x_0(t), y_0(t), z_0(t))$.

Similarly, we also have

$$\frac{d}{dt}(\Delta y) = \Delta y \frac{\partial G}{\partial y}, \quad \frac{d}{dt}(\Delta z) = \Delta z \frac{\partial H}{\partial z}.$$

Since

$$\frac{d\Gamma}{dt} = \frac{d(\Delta x)}{dt}\Delta y \Delta z + \Delta x \frac{d(\Delta y)}{dt}\Delta z + \Delta x \Delta y \frac{d(\Delta z)}{dt},$$

we have

$$\frac{d\Gamma}{dt} = \left(\frac{\partial F}{\partial x} + \frac{\partial G}{\partial y} + \frac{\partial H}{\partial z}\right)\Gamma, \tag{12.8}$$

valid for the evolution of an infinitesimal box.

We can conclude from (12.8) that phase-space volume is conserved if the divergence on the right-hand-side vanishes, or that phase-space volume contracts exponentially if the divergence is negative.

For example, we can consider the equations for the damped, driven pendulum given by (11.15). The divergence in this case is derived from

$$\frac{\partial}{\partial \theta}(u) + \frac{\partial}{\partial u}\left(-\frac{1}{q}u - \sin\theta + f\cos\psi\right) + \frac{\partial}{\partial \psi}(\omega) = -\frac{1}{q};$$

and provided $q > 0$ (i.e., the pendulum is damped), phase-space volume contracts. For the undamped pendulum (where $q \to \infty$), phase-space volume is conserved.

12.4 Limit cycles

The asymptotic state of a nonlinear system may be a limit cycle instead of a fixed point. A stable limit cycle is a periodic solution on which all nearby solutions converge. Limit cycles can also be unstable. Determining the existence of a limit cycle from a nonlinear system of equations through analytical means is usually impossible, but limit cycles are easily found numerically. The damped, driven pendulum equations has no fixed points, but does have limit cycles.

12.5 Attractors and basins of attraction

An attractor is a stable configuration in phase space. For example, an attractor can be a stable fixed point or a limit cycle. The damped, non-driven pendulum converges to a stable fixed point corresponding to the pendulum at rest at the bottom. The damped, driven pendulum for certain values of the parameters converges to a limit cycle. We will see that the chaotic pendulum also has an attractor of a different sort, called a strange attractor.

The domain of attraction of a stable fixed point, or of a limit cycle, is called the attractor's basin of attraction. The basin of attraction consists of all initial conditions for which solutions asymptotically converge to the attractor. For nonlinear systems, basins of attraction can have complicated geometries.

12.6 Poincaré sections

The Poincaré section is a method to reduce by one or more dimensions the phase space diagrams of higher dimensional systems. Most commonly, a three-dimensional phase space can be reduced to two-dimensions, for which a plot may be easier to interpret. For systems with a periodic driving force such as the damped, driven pendulum system given by (11.15), the two-dimensional phase space (θ, u) can be viewed stroboscopically, with the strobe period taken to be the period of forcing, that is $2\pi/\omega$. For the damped, driven pendulum, the third dynamical variable ψ satisfies $\psi = \psi_0 + \omega t$, and the plotting of the phase-space variables (θ, u) at the times t_0, $t_0 + 2\pi/\omega$, $t_0 + 4\pi/\omega$, etc., is equivalent to plotting the values of (θ, u) every time the phase-space variable ψ in incremented by 2π.

For systems without periodic forcing, a plane can be placed in the three-dimensional phase space, and a point can be plotted on this plane every time the orbit passes through it. For example, if the dynamical variables are x, y and z, a plane might be placed at $z = 0$ and the values of (x, y) plotted every time z passes through zero. Usually, one also specifies the direction of passage, so that a point is plotted only when $\dot{z} > 0$, say.

12.7 Fractal dimensions

The attractor of the chaotic pendulum is called strange because it occupies a fractional dimension of phase space. To understand what is meant by fractional dimensions, we first review some classical fractals.

12.7.1 Classical fractals

Cantor set

Perhaps the most famous classical fractal is the Cantor set. To construct the Cantor set, we start with the line segment $S_0 = [0, 1]$. We then remove the middle one-third of this line segment, and denote the remaining set as $S_1 = [0, 1/3] \cup [2/3, 1]$. Then we remove the middle one-third of both remaining line segments, and denote this set as $S_2 = [0, 1/9] \cup [2/9, 1/3] \cup [2/3, 7/9] \cup [8/9, 1]$. Repeating this process ad infinitum, the Cantor set is defined as

$$S = \lim_{n \to \infty} S_n. \tag{12.9}$$

An illustration showing how to construct the Cantor set is shown in Fig. 12.5.

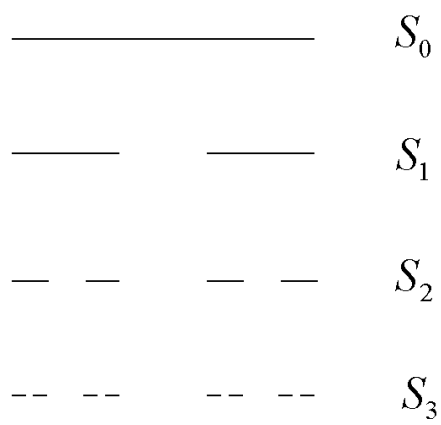


Figure 12.5: *Construction of the Cantor set.*

The Cantor set has some unusual properties. First, we compute its length. Let l_n denote the length of the set S_n. Clearly, $l_0 = 1$. Since S_1 is constructed by removing the middle third of S_0, we have $l_1 = 2/3$. To construct S_2, we again remove the middle third of the two line segments of S_1, reducing the length of S_1 by another factor of $2/3$, so that $l_2 = (2/3)^2$, and so on. We obtain $l_n = (2/3)^n$, and the $\lim_{n\to\infty} l_n = 0$. Therefore, the Cantor set has zero length.

The Cantor set, then, does not consist of line segments. Yet, the Cantor set is certainly not an empty set. We can see that at stage 1, there are two interior endpoints $1/3$ and $2/3$, and these will never be removed; at stage 2, there are the additional interior endpoints $1/9$, $2/9$, $7/9$ and $8/9$; and at stage 3, we add eight more interior endpoints. We see, then, that at stage k we add 2^k more interior endpoints. An infinite but countable number of endpoints are therefore included in the Cantor set.

But the Cantor set consists of much more than just these endpoints, and we will in fact show that the Cantor set is an uncountable set. Recall from analysis, that an infinite set of points can be either countable or uncountable. A countable set of points is a set that can be put in a one-to-one correspondence with the set of natural numbers. As is well known, the infinite set of all rational numbers is countable whereas the infinite set of real numbers is uncountable. By listing the endpoints of the Cantor set above, each stage adding 2^k more endpoints, we have shown that the set of all endpoints is a countable set.

In order to prove that the Cantor set is uncountable, it is helpful to make use of the base 3 representation of numbers. Recall that any number N, given by

$$N = \ldots \mathrm{a} * B^3 + \mathrm{b} * B^2 + \mathrm{c} * B + \mathrm{d} * B^0 + \mathrm{e} * B^{-1} + \mathrm{f} * B^{-2} + \mathrm{g} * B^{-3} + \ldots,$$

can be written in base B as

$$N = \ldots \mathrm{abcd.efg} \ldots \quad (\text{base } B),$$

where the period separating the integer part of the number from the fractional part is in general called a radix point. For base 10, of course, the radix point is called a decimal point, and for base 2, a binary point.

Now, consider the Cantor set. In the first stage, all numbers lying between $1/3$ and $2/3$ are removed. The remaining numbers, then, must be of the form $0.0\ldots$ (base 3) or $0.2\ldots$ (base 3), since (almost) all the numbers having the form $0.1\ldots$ (base 3) have been removed. The single exception is the endpoint $1/3 = 0.1$ (base 3), but this number can also be written as $1/3 = 0.0\overline{2}$ (base 3), where the bar over a number or numbers signifies an infinite repetition. In the second stage, all numbers lying between $1/9$ and $2/9$, say, are removed, and these numbers are of the form $0.01\ldots$ (base 3). We can see, then, that the Cantor set can be defined as the set of all numbers on the unit interval that contain no 1's in their base 3 representation.

Using base 3 notation, it is easy to find a number that is not an endpoint, yet is in the Cantor set. For example, the number $1/4$ is not an endpoint. We can convert $1/4$ to base 3 by the following calculation:

$$\begin{aligned}\frac{1}{4} \times 3 &= 0.75,\\ 0.75 \times 3 &= 2.25,\\ 0.25 \times 3 &= 0.75,\\ 0.75 \times 3 &= 2.25,\end{aligned}$$

and so on, from which we find that $1/4 = 0.\overline{02}$ (base 3). The number $1/4$, therefore, has no 1's in its base 3 expansion and so is an element of the Cantor set.

We can now prove that the Cantor set is an uncountable set. The proof is similar to that used to prove that the real numbers are uncountable. If we suppose that the Cantor set is countable, then we can list all of its elements using a base 3 expansion; that is,

$$\begin{aligned}x_1 &= 0.x_{11}x_{12}x_{13}\ldots \quad \text{(base 3)},\\ x_2 &= 0.x_{21}x_{22}x_{23}\ldots \quad \text{(base 3)},\\ x_3 &= 0.x_{31}x_{32}x_{33}\ldots \quad \text{(base 3)},\end{aligned}$$

and so on, where all the x_{ij}'s must be either a 0 or a 2 (i.e., no 1's are allowed). It is now simple to name a number y not on this list. We have

$$y = 0.y_{11}y_{22}y_{33}\ldots,$$

where $y_{11} \neq x_{11}$, $y_{22} \neq x_{22}$, $y_{33} \neq x_{33}$, etc.. That is, if $x_{11} = 0$, then $y_{11} = 2$; if $x_{11} = 2$, then $y_{11} = 0$, and so on. By constructing a number not on the list, we have obtained a reductio ad absurdum, and we can conclude that the Cantor set is uncountable.

Note that the endpoints of the Cantor set are just those numbers written in base 3 that end with either all 0's or all 2's, and these indeed form only a very small subset of the entire Cantor set.

From the base 3 representation, we can now see that the Cantor set has the following interesting characteristic. On the one hand, any small interval around a point in the Cantor set contains another point in the Cantor set. On the other hand, there is an interval between any two points in the Cantor set that is not in the Cantor set. For example, take the point $1/4 = 0.\overline{02}$ in the Cantor set, and the interval $[1/4 - 3^{-4}, 1/4 + 3^{-4}]$. In base 3, the interval is given by $[0.0201\overline{02}, 0.0210\overline{02}]$. There are, of course, an infinite number of points in the Cantor set in this interval, one of them being 0.0202 to the left of $1/4$, and 0.02022 to the right of $1/4$. If the interval was established using 3^{-n}, for any n, we can still find other points in the interval that are in the Cantor set. Also, between any two points in the Cantor set, there is an interval that was removed during the

construction of the Cantor set. So the Cantor set consists of neither line segments nor a set of discrete points.

The Cantor set is also self-similar, meaning that it contains smaller copies of itself at all scales.

The Cantor set is called a fractal, and is said to have a fractal dimension. Integer dimensions are more familiar to us: we say that a line has dimension 1; an area, dimension 2; and a volume, dimension 3.

For self-similar sets, we can place the definition of dimension on a more mathematical basis. A line segment, a square, and a cube can also be considered self-similar. Suppose that a self-similar set S is composed of m copies of itself scaled down by a factor of r. As examples, the line segment $[0,1]$ is composed of two copies of itself scaled down by a factor of two; namely, the segments $[0,1/2]$ and $[1/2,1]$. A square is composed of four copies of itself scaled down by a factor of two. And a cube is composed of eight copies of itself scaled down by a factor of two. We write

$$m = r^D,$$

where D is called the similarity dimension of the set. With $r = 2$, the line segment has $m = 2$, the square has $m = 4$, and the cube has $m = 8$, yielding the usual dimensions $D = 1$, 2 and 3, respectively.

Now the Cantor set is composed of two copies of itself scaled down by a factor of three. Therefore,

$$2 = 3^D,$$

and the similarity dimension of the Cantor set is given by $D = \log 2/\log 3 \approx 0.6309$, which has dimension smaller than that of a line, but larger than that of a point.

Sierpinski triangle

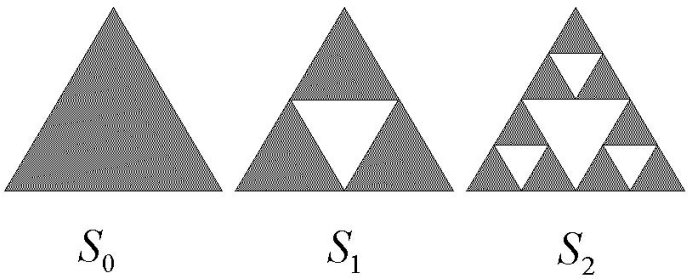


Figure 12.6: *Construction of the Sierpinski triangle.*

Other classical fractals can be constructed similarly to the Cantor set. The Sierpinski triangle starts with an equilateral triangle, S_0, with unit sides. We then draw lines connecting the three midpoints of the three sides. The just formed equilateral middle triangle of side length $1/2$ is

then removed from S_0 to form the set S_1. This process is repeated on the remaining equilateral triangles, as illustrated in Fig. 12.6. The Sierpinski triangle S is defined as the limit of this repetition; that is, $S = \lim_{n\to\infty} S_n$.

The Sierpinski triangle is composed of three copies of itself scaled down by a factor of two, so that

$$3 = 2^D,$$

and the similarity dimension is $D = \log 3/\log 2 \approx 1.5850$, which has dimension between a line and an area.

Similar to the Cantor set, the Sierpinksi triangle, though existing in a two-dimensional space, has zero area. If we let A_i be the area of the set S_i, then since area is reduced by a factor of $3/4$ with each iteration, we have

$$A_n = \left(\frac{3}{4}\right)^n A_0,$$

so that $A = \lim_{n\to\infty} A_n = 0$.

Sierpinski carpet

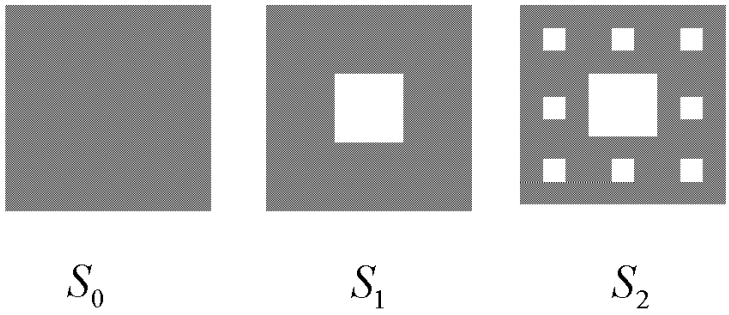


Figure 12.7: *Construction of the Sierpinski Carpet*

The Sierpinski carpet starts with a square, S_0, with unit sides. Here, we remove a center square of side length $1/3$; the remaining set is called S_1. This process is then repeated as illustrated in Fig. 12.7.

Here, the Sierpinski carpet is composed of eight copies of itself scaled down by a factor of three, so that

$$8 = 3^D, \tag{12.10}$$

and the similarity dimension is $D = \log 8/\log 3 \approx 1.8928$, somewhat larger than the Sierpinski triangle. One can say, then, that the Sierpinski carpet is more space filling than the Sierpinski triangle, though it still has zero area.

Koch curve

Like the Cantor set, we start with the unit interval, but now we replace the middle one-third by two line segments of length $1/3$, as illustrated in 12.8, to form the set S_1. This process is then repeated on the four line segments of length $1/3$ to form S_2, and so on, as illustrated in Fig. 12.8.

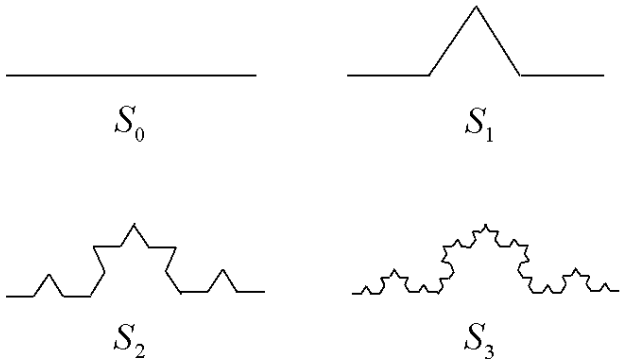


Figure 12.8: *Construction of the Koch curve.*

Here, the Koch curve is composed of four copies of itself scaled down by a factor of three, so that

$$4 = 3^D, \tag{12.11}$$

and the similarity dimension is $D = \log 4 / \log 3 \approx 1.2619$. The Koch curve therefore has a dimension lying between a line and an area. Indeed, with each iteration, the length of the Koch curve increases by a factor of $4/3$ so that its length is infinite.

Koch snow flake

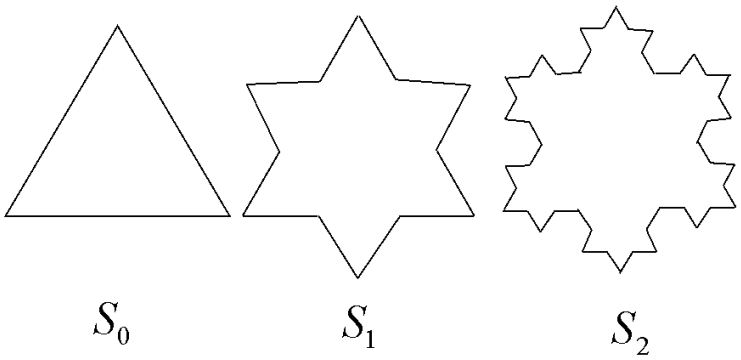


Figure 12.9: *Construction of the Koch snow flake.*

Here, we start with an equilateral triangle. Similar to the Koch curve, we replace the middle one-third of each side by two line segments, as illustrated in 12.9. The boundary of the Koch snow flake has the same fractal dimension as the Koch curve, and is of infinite length. The area bounded by the boundary is obviously finite, however, and can be shown to be $8/5$ of the area of the original triangle. Interestingly, here an infinite perimeter encloses a finite area.

12.7.2 Correlation Dimension

Modern studies of dynamical systems have discovered sets named strange attractors. These sets share some of the characteristics of classical fractals: they are not space filling yet have structure at arbitrarily small scale. However, they are not perfectly self-similar in the sense of the classical fractals, having arisen from the chaotic evolution of some dynamical system. Here, we attempt to generalize the definition of dimension so that it is applicable to strange attractors and relatively easy to compute. The definition and numerical algorithm described here was first proposed in a paper by Grassberger and Procaccia (1993).

Definitions

Consider a set S of N points. Denote the points in this set as $\mathbf{x}_i$, with $i = 1, 2, \ldots, N$ and denote the distance between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ as r_{ij}. Note that there are $N(N-1)/2$ distinct distances. We define the correlation integral $C(r)$ to be

$$C(r) = \lim_{N\to\infty} \frac{2}{N(N-1)} \times \{\text{number of distinct distances } r_{ij} \text{ less than } r\}.$$

If $C(r) \propto r^D$ over a wide range of r when N is sufficiently large, then D is to be called the correlation dimension of the set S. Note that with this normalization, $C(r) = 1$ for r larger than the largest possible distance between two points on the set S.

The correlation dimension agrees with the similarity dimension for an exactly self-similar set. As a specific example, consider the Cantor set. Suppose N points that lie on the Cantor set are chosen at random to be in S. Since every point on the set is within a distance $r = 1$ of every other point, one has $C(1) = 1$. Now, approximately one-half of the points in S lie between 0 and $1/3$, and the other half lie between $2/3$ and 1. Therefore, only $1/2$ of the possible distinct distances r_{ij} will be less than $1/3$ (as $N \to \infty$), so that $C(1/3) = 1/2$. Continuing in this fashion, we find in general that $C(1/3^s) = 1/2^s$. With $r = 1/3^s$, we have

$$s = -\frac{\log r}{\log 3},$$

so that

$$\begin{aligned} C(r) &= 2^{\log r/\log 3} \\ &= \exp(\log 2 \log r/\log 3) \\ &= r^{\log 2/\log 3}, \end{aligned}$$

valid for small values of r. We have thus found a correlation dimension, $D = \log 2/\log 3$, in agreement with the previously determined similarity dimension.

Numerical computation

Given a finite set S of N points, we want to formulate a fast numerical algorithm to compute $C(r)$ over a sufficiently wide range of r to accurately determine the correlation dimension D from a log-log plot of $C(r)$ versus r. A point $\mathbf{x}_i$ in the set S may be a real number, or may be a coordinate pair. If the points come from a Poincaré section, then the fractal dimension of the Poincaré section will be one less than the actual fractal dimension of the attractor in the full phase space.

Define the distance r_{ij} between two points $\mathbf{x}_i$ and $\mathbf{x}_j$ in S to be the standard Euclidean distance. To obtain an accurate value of D, one needs to throw away an initial transient before collecting points for the set S.

Since we are interested in a graph of $\log C$ versus $\log r$, we compute $C(r)$ at points r that are evenly spaced in $\log r$. For example, we can compute $C(r)$ at the points $r_s = 2^s$, where s takes on integer values.

First, one counts the number of distinct distance r_{ij} that lie in the interval $r_{s-1} \leq r_{ij} < r_s$. Denote this count by $M(s)$. An approximation to the correlation integral $C(r_s)$ using the N data points is then obtained from

$$C(r_s) = \frac{2}{N(N-1)} \sum_{s'=-\infty}^{s} M(s').$$

One can make use of a built-in MATLAB function to speed up the computation. The function call

```
[F,E] = log2(x);
```

returns the floating point number F and the integer E such that `x = F*2^E`, where $0.5 \leq |\mathrm{F}| < 1$ and E is an integer. To count each r_{ij} in the appropriate bin $M(s)$, one can compute all the values of r_{ij} and place them into the Matlab array `r`. Then one uses the vector form of `log2.m` to compute the corresponding values of `s`:

```
[~,s] = log2(r); .
```

One can then increment the count in a Matlab array `M` that corresponds to the particular integer values of `s`. The Matlab function `cumsum.m` can then be used to compute the Matlab array `C`.

Finally, a least-squares analysis of the data is required to compute the fractal dimension D. By directly viewing the log-log plot, one can choose which adjacent points to fit a straight line through, and then use the method of least-squares to compute the slope. The MATLAB function `polyfit.m` can determine the best fit line and the corresponding slope. Ideally, one would also compute a statistical error associated with this slope, and such an error should go to zero as the number of points computed approaches infinity.

Chapter 13

Pendulum dynamics

13.1 Phase portrait of the undriven pendulum

The undriven pendulum has only a two dimensional space, forming a phase plane where it is easy to visualize the phase portraits. The phase portrait of the small amplitude simple pendulum is particularly easy to draw. The dimensionless form of the simple pendulum equation is

$$\ddot{\theta} + \theta = 0,$$

and the solutions for $\theta = \theta(t)$ and $u = \dot{\theta}(t)$ are given by

$$\theta(t) = \theta_m \cos(t + \varphi), \quad u(t) = -\theta_m \sin(t + \varphi).$$

The phase portrait in the θ-u phase plane consists of concentric circles, with clockwise motion along these circles, as seen in the small diameter circles of Fig. 13.1. As the amplitude increases, the approximation $\sin\theta \approx \theta$ loses validity, and the relevant dimensionless equation becomes

$$\ddot{\theta} + \sin\theta = 0.$$

Beyond the small diameter circles of Fig. 13.1, one observes that the connected lines become elongated in u as the amplitude increases, implying the pendulum is slowed down when the amplitude becomes large (i.e., the period of the pendulum is lengthened). Eventually, a final closed curve is drawn, called the separatrix, that separates the pendulum's motion from periodic to rotary, the latter motion corresponding to a pendulum that swings over the top in a circular motion. Exactly on the separatrix trajectory, the pendulum comes to rest at the angle π, or 180°, which is an unstable fixed point of the pendulum.

The simple pendulum is a conservative system, exhibiting a conservation law for energy, and this implies a conservation of phase space area (or volume). If one evolves over time a given initial area of the phase space of Fig. 13.1, the area of this phase space will be conserved. When the pendulum is damped, however, the area of this phase space will shrink to zero.

13.2 Basin of attraction of the undriven pendulum

We consider the damped, undriven pendulum with governing equation

$$\ddot{\theta} + \frac{1}{q}\dot{\theta} + \sin\theta = 0,$$

and the stable fixed point given by $(\theta, \dot{\theta}) = (0, 0)$. How can we determine the basin of attraction of this fixed point? Of course, with $\dot{\theta} = 0$, only the values $-\pi < \theta < \pi$ will lie in the basin of attraction. But we also need to compute the basin of attraction for nonzero initial values of $\dot{\theta}$.

As is often the case, to devise a numerical algorithm it is best to appeal directly to the physics. We want to find the borderline of initial conditions between either the attractive fixed points $(0, 0)$ and $(2\pi, 0)$, or the attractive fixed points $(0, 0)$ and $(-2\pi, 0)$. The initial conditions just on the border result in the pendulum reaching either the unstable fixed points $(\pi, 0)$ or $(-\pi, 0)$. A small

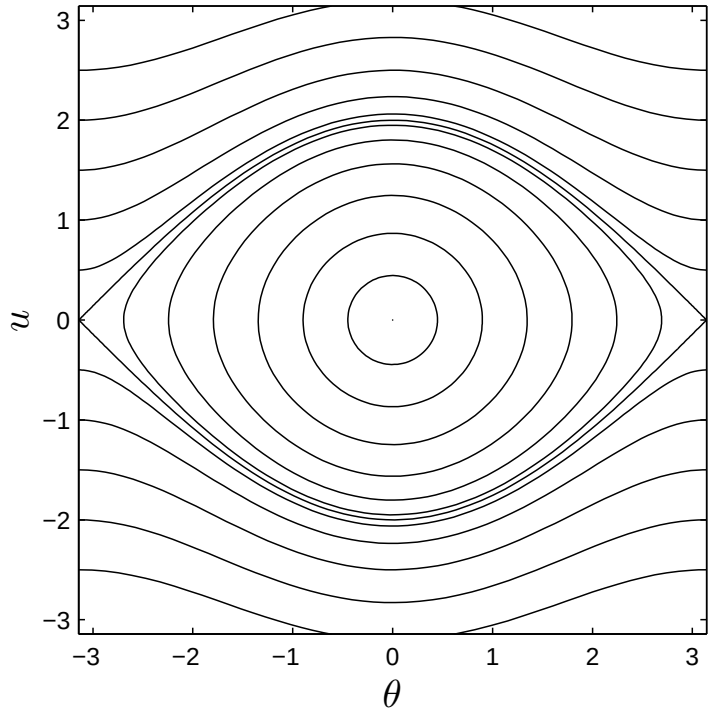


Figure 13.1: *Phase portrait of the simple pendulum.*

perturbation from $(\pi, 0)$ will result in one of the attractive points $(0, 0)$ or $(2\pi, 0)$; from $(-\pi, 0)$, one of the attractive points $(0, 0)$ or $(-2\pi, 0)$.

The algorithm then initializes the calculation at either the unstable fixed point $(\pi, 0)$ or $(-\pi, 0)$, with a small perturbation in either the position or velocity that results in the final attracting point being $(0, 0)$. We can call these values final conditions because the differential equations are then integrated backward in time to determine all possible initial conditions that can result in these final conditions. You can convince yourself that the four final conditions that should be used are the two pairs $(\pi, -\epsilon)$, $(\pi - \epsilon, 0)$, and $(-\pi, \epsilon)$, $(-\pi + \epsilon, 0)$, with ϵ very small. The first of each pair corresponds to the immediately previous motion being rotary, and the second of each pair corresponds to the immediately previous motion being oscillatory.

A graph of the basins of attraction of $(0, 0)$ for $q = 4$, corresponding to an underdamped pendulum (critical damping is $q = 1/2$) is shown in Fig. 13.2. The attracting fixed point is marked by an 'x', and the region inside the two curved lines is the basin of attraction.

13.3 Spontaneous symmetry-breaking bifurcation

An interesting supercritical pitchfork bifurcation occurs in the pendulum equations, where at the bifurcation point one stable limit cycle splits into two. The single limit cycle displays a symmetry that is no longer respected individually by the two new limit cycles. This type of pitchfork bifurcation is a manifestation of what is called spontaneous symmetry breaking.

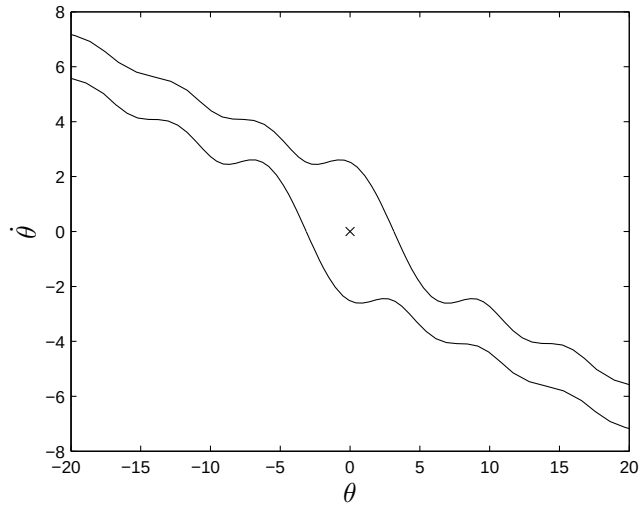


Figure 13.2: *Basin of attraction of* $(\theta, \dot{\theta}) = (0, 0)$ *for the unforced, underdamped pendulum with* $q = 4$*. The cross marks the attracting fixed point.*

The pendulum equation (11.14), written again here, is given by

$$\ddot{\theta} + \frac{1}{q}\dot{\theta} + \sin\theta = f\cos\omega t. \tag{13.1}$$

Using $\sin(-\theta) = -\sin\theta$ and $\cos(\omega t - \pi) = -\cos\omega t$, the pendulum equation can be seen to be invariant under the transformation

$$\theta \to -\theta, \quad t \to t - \pi/\omega, \tag{13.2}$$

with the physical interpretation that the equations of motion make no distinction between the right and left sides of the vertical, a consequence of the symmetry of both the pendulum and the external force.

Consider again the asymptotic small amplitude solution given by (11.10), which in dimensionless variables is

$$\theta(t) = \frac{f}{\sqrt{(1-\omega^2)^2 + (\omega/q)^2}}\cos(\omega t + \phi),$$

with

$$\tan\phi = \frac{\omega/q}{\omega^2 - 1}.$$

We can observe that this solution is also invariant under (13.2): the small amplitude solution obeys $\theta(t) = -\theta(t - \pi/\omega)$. We say that this solution is symmetric, meaning it obeys the same symmetry as the governing equations; that is, the motion of the small amplitude pendulum is symmetrical about the vertical.

In general, if $\theta = \theta_1(t)$ is a solution of (13.1), then so is $\theta = \theta_2(t)$, where $\theta_2(t) = -\theta_1(t - \pi/\omega)$. We can prove this mathematically. Assume $\theta = \theta_1(t)$ satisfies (13.1). We then show that $\theta = \theta_2$

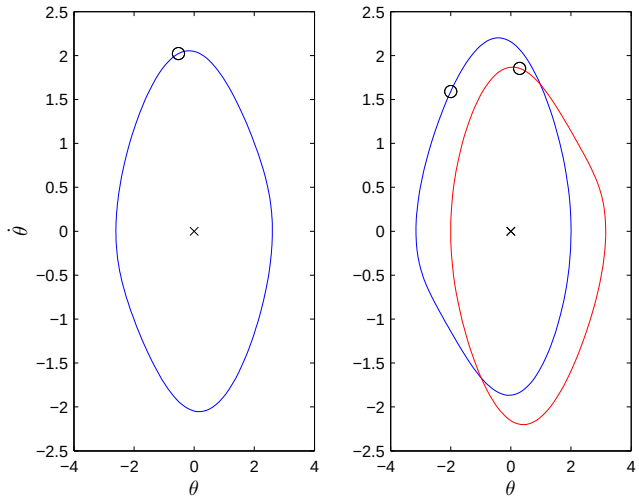


Figure 13.3: *Phase-space projection of the pendulum solution before and after spontaneous symmetry breaking. The 'x' represents the pendulum at rest at the bottom. The 'o's denote the positions* $(\theta, \dot{\theta})$ *and* $(-\theta, -\dot{\theta})$ *at the times* $t = 0$ *and* $t = \pi/\omega$*, respectively. In both plots,* $f = 1.5$*,* $\omega = 2/3$*. (a)* $q = 1.24$ *and the solution is observed to be symmetric; (b)* $q = 1.3$ *and one pair of asymmetric solutions is observed.*

also satisfies (13.1) by the following calculation:

$$\begin{aligned}\ddot{\theta}_2(t) + \frac{1}{q}\dot{\theta}_2(t) + \sin(\theta_2(t)) &= -\ddot{\theta}_1(t - \pi/\omega) - \frac{1}{q}\dot{\theta}_1(t - \pi/\omega) - \sin(\theta_1(t - \pi/\omega)) \\ &= -f\cos(\omega t - \pi) \\ &= f\cos\omega t.\end{aligned}$$

If the two solutions $\theta_1(t)$ and $\theta_2(t)$ are equal, then we say that this solution is symmetric. Here, we are considering asymptotic solutions that are independent of the initial conditions, since the initial conditions themselves can also break the symmetry. However, if $\theta_1(t)$ and $\theta_2(t)$ are not equal, we say that these solutions are asymmetric, and that spontaneous symmetry breaking has occurred. Evidently, spontaneous symmetry breaking is a decidedly nonlinear phenomena. After symmetry breaking occurs, the asymmetric solutions must occur in pairs, and the bifurcation point looks like a pitchfork bifurcation, and can be super- or sub-critical. Any subsequent bifurcation that occurs to one of the asymmetric solutions must also be mirrored by the other.

Spontaneous symmetry breaking occurs in the pendulum dynamics at the dimensionless parameter values $f = 1.5$, $\omega = 2/3$, and approximately $q = 1.246$. For q just less than 1.246, the single stable asymptotic solution for $\theta = \theta(t)$ is symmetric, and for q just greater than 1.246, there exists a pair of stable asymptotic solutions that are asymmetric.

By projecting the phase-space trajectory onto the $\theta - \dot{\theta}$ plane, in Fig. 13.3 we display both the symmetric solution when $q = 1.24$ and the two asymmetric solutions when $q = 1.30$. To explicitly observe the symmetry of the solutions, we mark the value of $(\theta, \dot{\theta})$ at the time $t = n2\pi/\omega$, with n a positive integer (equivalent to the time $t = 0$), and the value of $(-\theta, -\dot{\theta})$ at the time $t = n2\pi/\omega + \pi/\omega$ (equivalent to the time $t = \pi/\omega$). For a symmetric solution, these two points mark the same point on the trajectory, and for asymmetric solutions they mark

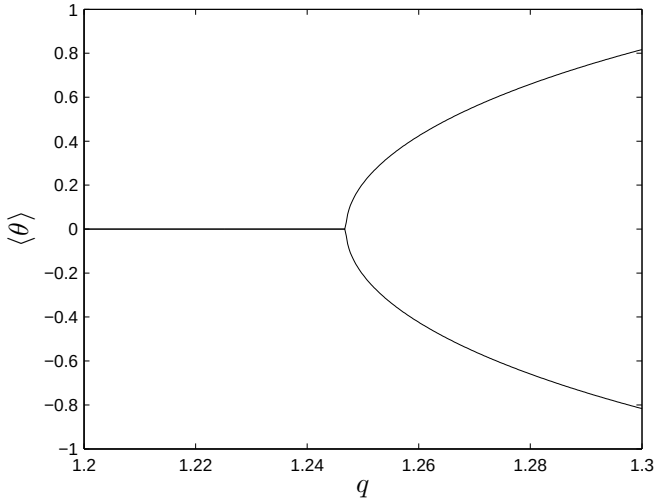


Figure 13.4: *Bifurcation diagram exhibiting spontaneous symmetry breaking. Here,* $f = 1.5$*,* $\omega = 2/3$*, and* q *is the control parameter. We plot* $\langle\theta\rangle$ *versus* q*.*

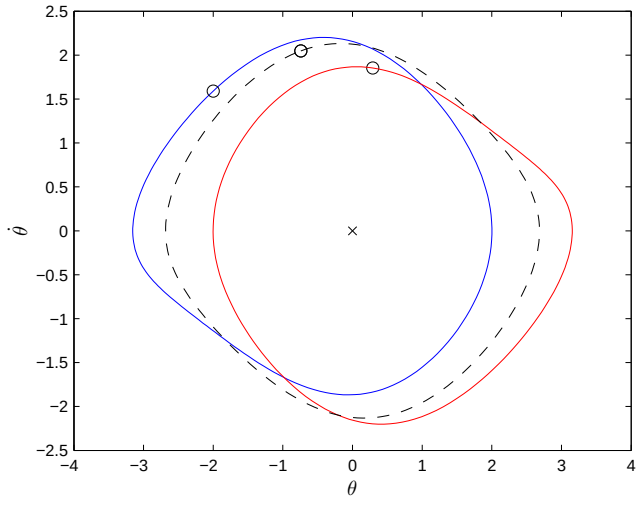


Figure 13.5: *Phase-space projection of the pendulum solution after spontaneous symmetry breaking, as in Fig. 13.3b. The unstable symmetric limit cycle is the dashed-line curve.*

points on different trajectories. Notice that after the occurrence of symmetry breaking, one of the asymptotic solutions undergoes an oscillation centered to the right of the vertical, and the other, centered to the left.

We can graph a bifurcation diagram associated with the symmetry breaking of the solutions. We fix $f = 1.5$ and $\omega = 2/3$ and vary q across the bifurcation point $q = 1.246$. We need to distinguish the symmetric from the asymmetric limit cycles, and one method is to compute the average value of $\theta(t)$ over one period of oscillation; that is,

$$\langle\theta\rangle = \frac{1}{T}\int_0^T \theta dt,$$

where $T = 2\pi/\omega$. For the symmetric solution, $\langle\theta\rangle = 0$, whereas $\langle\theta\rangle$ takes on both positive and negative values after symmetry breaking occurs. In Fig. 13.4, we plot the value of $\langle\theta\rangle$ versus q. At a value of approximately $q = 1.246$, spontaneous symmetry breaking occurs and the stable symmetric limit cycle splits into two asymmetric limit cycles, in what is evidently a supercritical pitchfork bifurcation.

The symmetric limit cycle still exists after the bifurcation point, and though unstable, can also be computed. To compute the unstable cycle, one could determine the values of θ and $\dot\theta$ at $t = 0$ that lie on this cycle, and then integrate over one period (over which the instability doesn't have sufficient time to develop). The problem of determining the correct initial conditions can be cast as a problem in multidimensional root-finding. The key idea is that a symmetric limit cycle satisfies $\theta(t) = -\theta(t - \pi/\omega)$. We therefore determine the solution vector of initial conditions $(\theta(0), \dot\theta(0))$ that satisfies the two equations

$$\theta(0) + \theta(\pi/\omega) = 0,$$
$$\dot\theta(0) + \dot\theta(\pi/\omega) = 0,$$

where $\theta(\pi/\omega)$ and $\dot\theta(\pi/\omega)$ are determined by integrating from $t = 0$ the differential equations using `ode45.m` with the initial conditions $(\theta(0), \dot\theta(0))$. Root-finding can be done using either a two-dimensional version of the secant rule or, more simply, the built-in MATLAB function `fsolve.m`. Convergence to the roots is robust, and an initial guess for the roots can be taken, for instance, as $(\theta(0), \dot\theta(0)) = (0, 0)$. A plot of the two stable asymmetric limit cycles, and the unstable symmetric limit cycle when $f = 1.5$, $\omega = 2/3$ and $q = 1.3$ is shown in Fig. 13.5.

13.4 Period-doubling bifurcations

As q increases further, another series of bifurcations occur, called period doubling bifurcations. These too are supercritical pitchfork bifurcations. These bifurcations happen simultaneously to the solutions with positive and negative values of I, and we need only consider one of these branches here. Before the first bifurcation occurs, the pendulum has period $2\pi/\omega$—the same period as the external force—and the phase-space trajectory forms a closed loop when the equations are integrated over a single period. After the first period-doubling bifurcation, which occurs approximately at $q = 1.348$, the period of the pendulum becomes twice the period of the external force. In Fig. 13.6, we plot a phase-space projection onto the $\theta - \dot\theta$ plane before and after the first period-doubling bifurcation. Before the bifurcation, we say the pendulum has period one, and after the bifurcation we say the pendulum has period two. Note that for the pendulum of period two, the phase-space trajectory loops around twice before closing, each loop corresponding to one period of the external force. To further illustrate the period-two oscillation, in Fig. 13.7 we plot the time series of $\theta(t)$ versus t over exactly two periods of the external force. Evidently, the pendulum motion is now periodic with period twice the period of the external force (the period of the external force is approximately 9.4).

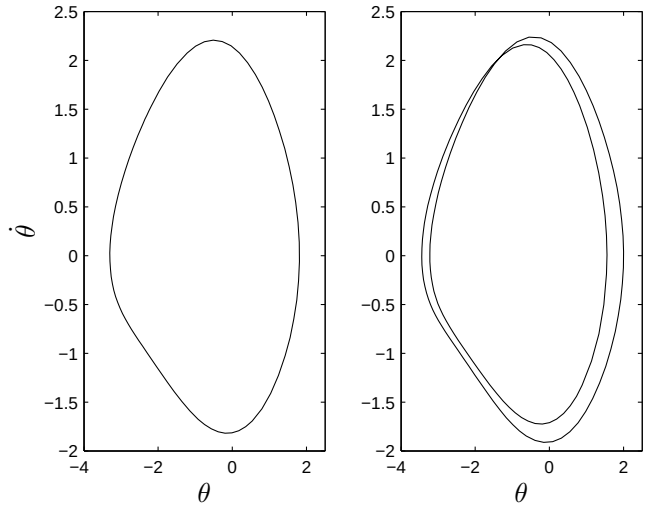


Figure 13.6: *Phase-space projection of the pendulum solution before and after the first period-doubling bifurcation. In both plots,* $f = 1.5$, $\omega = 2/3$. *(a)* $q = 1.34$ *and the oscillation has period one; (b)* $q = 1.36$ *and the oscillation has period two.*

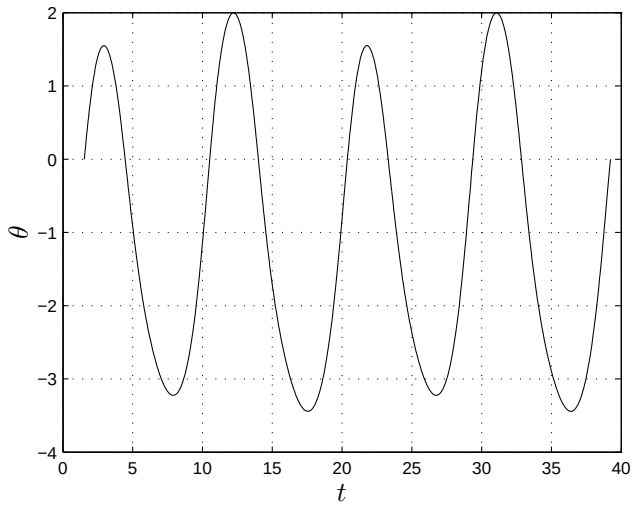


Figure 13.7: *Time series of the pendulum oscillation with period two. Here,* $f = 1.5$, $\omega = 2/3$, *and* $q = 1.36$.

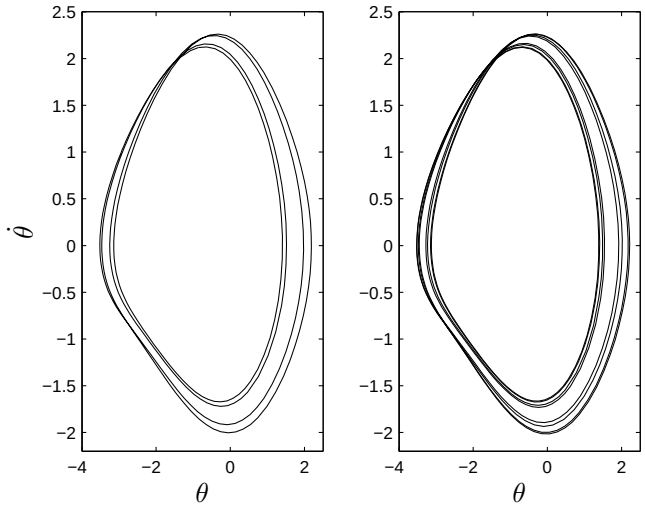


Figure 13.8: *Phase-space projection of the pendulum solution before and after the third period-doubling bifurcation. In both plots,* $f = 1.5$, $\omega = 2/3$. *(a)* $q = 1.3740$ *and the oscillation has period four; (b)* $q = 1.3755$ *and the oscillation has period eight.*

The period-doubling bifurcations continue with increasing q. The second doubling from period two to period four occurs at approximately $q = 1.370$ and the third doubling from period four to period eight occurs at approximately $q = 1.375$. The $\theta - \dot{\theta}$ phase-space projections for period four and period eight are shown in Fig. 13.8.

A bifurcation diagram can be plotted that illustrates these period-doubling bifurcations. We use a Poincaré section to plot the value of θ at the times corresponding to $2\pi n/\omega$, with n an integer. The control parameter for the bifurcation is q, and in Fig. 13.9, we plot the bifurcation diagram for $1.34 < q < 1.38$. In general, the angle θ should be mapped into the interval $-\pi < \theta < \pi$, but for these parameter values there is no rotary motion. Period-doubling bifurcations are observed, and eventually the pendulum becomes aperiodic. Additional windows of periodicity in the aperiodic regions of q are also apparent.

The aperiodic behavior of the pendulum observed in Fig. 13.9 corresponds to a chaotic pendulum. If only the pendulum exhibited the period-doubling route to chaos, then the results shown in Fig. 13.9, though interesting, would be of lesser importance. But in fact many other nonlinear systems also exhibit this route to chaos, and there are some universal features of Fig. 13.9, first discovered by Feigenbaum in 1975. One of these features is now called the Feigenbaum constant, δ.

To compute the Feigenbaum constant, one first defines q_n to be the value of q at which the pendulum motion bifurcates from period 2^{n-1} to period 2^n. We have already mentioned that $q_1 \approx 1.348$, $q_2 \approx 1.370$, and $q_3 \approx 1.375$. We now define

$$\delta_n = \frac{q_{n+1} - q_n}{q_{n+2} - q_{n+1}}, \tag{13.3}$$

and the Feigenbaum constant as

$$\delta = \lim_{n \to \infty} \delta_n. \tag{13.4}$$

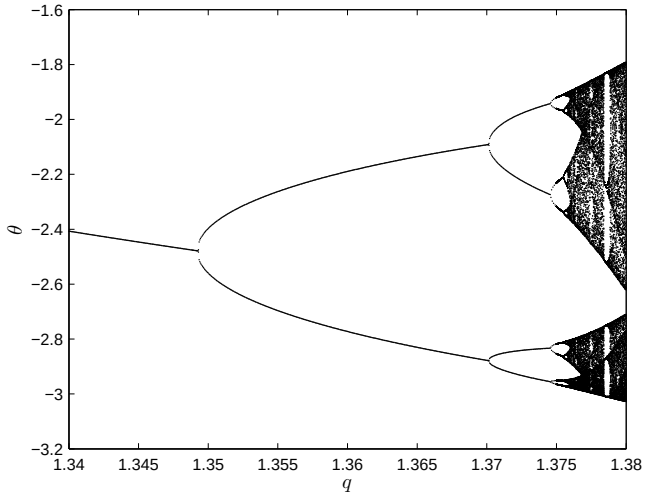


Figure 13.9: *Bifurcation diagram for period-doubling in the pendulum. A Poincaré section plots θ at the times $t = 2\pi n/\omega$, with n an integer. Here, $f = 1.5$, $\omega = 2/3$, and $1.34 < q < 1.38$*

Note that for the pendulum, we can already compute

$$\delta_1 = \frac{1.370 - 1.348}{1.375 - 1.370} = 4.400.$$

The meaning of δ can be better elucidated by writing

$$q_{n+2} - q_{n+1} = \frac{q_{n+1} - q_n}{\delta_n};$$

and by continuing to iterate this equation, we obtain

$$q_{n+2} - q_{n+1} = \frac{q_2 - q_1}{\delta_1 \delta_2 \cdots \delta_n}.$$

With all the δ_n's approximately equal to δ, we then have the scaling

$$q_{n+2} - q_{n+1} \propto \delta^{-n}.$$

A δ larger than one would then insure that the bifurcations occur increasingly closer together, so that an infinite period (and chaos) is eventually attained. The value of δ can be computed to high accuracy and has been found to be

$$\delta = 4.669201609102990\ldots,$$

and our value of $\delta_1 = 4.4$ is a rough approximation.

13.5 Period doubling in the logistic map

Feigenbaum originally discovered the period-doubling route to chaos by studying a simple one-dimensional map. A one-dimensional map with a single control parameter μ can be written as

$$x_{n+1} = f_\mu(x_n), \tag{13.5}$$

where $f_\mu(x)$ is some specified function. A one-dimensional map is iterated, starting with some initial value x_0, to obtain the sequence $x_1, x_2, x_3, \ldots$. If the sequence converges to x_*, then x_* is a stable fixed point of the map.

The specific one-dimensional map we will study here is the logistic map, with

$$f_\mu(x) = \mu x(1-x). \tag{13.6}$$

The logistic map is perhaps the simplest nonlinear equation that exhibits the period-doubling route to chaos. To constrain the values of x_n to lie between zero and unity, we assume that $0 < \mu < 4$ and that $0 < x_0 < 1$.

A period-1 cycle for the logistic map corresponds to a stable fixed point. Stable fixed points are solutions of the equation $x = f_\mu(x)$, or

$$x = \mu x(1-x).$$

The two solutions are given by $x_* = 0$ and $x_* = 1 - 1/\mu$. The first fixed point $x_* = 0$ must be stable for $0 < \mu < 1$, being the only fixed point lying between zero and one that exists in this range. To determine the stability of the second fixed point, we make use of the linear stability analysis discussed in §12.1.

For a one-dimensional map, x_* is a stable fixed point of (13.5) if $|f'_\mu(x_*)| < 1$. For the logistic map given by (13.6), $f'_\mu(0) = \mu$ so that $x_* = 0$ is stable for $0 < \mu < 1$ as we have already surmised, and for the second fixed point

$$f'_\mu(1 - 1/\mu) = 2 - \mu.$$

Therefore, we find that $x_* = 1 - 1/\mu$ is stable for $1 < \mu < 3$.

What happens when μ becomes larger than three? We will now show that the first period-doubling bifurcation occurs at $\mu = 3$. Because of the simplicity of the logistic map, we can determine explicitly the period-2 cycle. Consider the following composite map:

$$g_\mu(x) = f_\mu\big(f_\mu(x)\big). \tag{13.7}$$

Fixed points of this map will consist of both period-1 cycles and period-2 cycles of the original map (13.6). If $x = x_*$ is a fixed point of the composite map (13.7), then x_* satisfies the equation

$$x = g_\mu(x).$$

A period-2 cycle of the map $f_\mu(x)$ necessarily corresponds to two distinct fixed points of the composite map (13.7). We denote these two fixed points by x_0 and x_1, which satisfy

$$x_1 = f_\mu(x_0), \quad x_0 = f_\mu(x_1).$$

We will call x_0, x_1 the orbit of the period-2 cycle, with the later generalization of calling x_0, x_1, $\ldots$, x_{2^n-1} the orbit of the period-2^n cycle.

The period-2 cycle can now be determined analytically by solving

$$\begin{aligned} x &= f_\mu\big(f_\mu(x)\big) \\ &= f_\mu\big(\mu x(1-x)\big) \\ &= \mu\big(\mu x(1-x)\big)\big(1-\mu x(1-x)\big), \end{aligned}$$

which is a quartic equation for x. Two solutions corresponding to period-1 cycles are known: $x_* = 0$ and $x_* = 1 - 1/\mu$. Factoring out these two solutions, the second by using long division, results in the quadratic equation given by

$$\mu^2 x^2 - \mu(\mu+1)x + (\mu+1) = 0.$$

The period-2 cycle, then, corresponds to the two roots of this quadratic equation; that is,

$$x_0 = \frac{1}{2\mu}\left((\mu+1) + \sqrt{(\mu+1)(\mu-3)}\right), \quad x_1 = \frac{1}{2\mu}\left((\mu+1) - \sqrt{(\mu+1)(\mu-3)}\right). \tag{13.8}$$

These roots are valid solutions for $\mu \geq 3$. Exactly at $\mu = 3$, the period-2 cycle satisfies $x_0 = x_1 = 2/3$, which coincides with the value of the period-1 cycle $x_* = 1 - 1/\mu = 2/3$. At $\mu = 3$, then, we expect the fixed point of the composite map corresponding to a period-1 cycle to go unstable via a supercritical pitchfork bifurcation to a pair of stable fixed points, corresponding to a period-2 cycle.

When does this period-2 cycle become unstable? At $\mu = 3$ and $x_* = 2/3$, we have

$$\begin{aligned} f'_\mu(x_*) &= \mu(1 - 2x_*) \\ &= -1, \end{aligned}$$

so that the period-1 cycle becomes unstable when the derivative of the map function attains the value of -1, and it is reasonable to expect that the period-2 cycle also becomes unstable when

$$g'_\mu(x_0) = -1, \quad g'_\mu(x_1) = -1. \tag{13.9}$$

Now,

$$\begin{aligned} g'_\mu(x_0) &= f'_\mu\big(f_\mu(x_0)\big) f'_\mu(x_0) \\ &= f'_\mu(x_1) f'_\mu(x_0), \end{aligned}$$

and

$$\begin{aligned} g'_\mu(x_1) &= f'_\mu\big(f_\mu(x_1)\big) f'_\mu(x_1) \\ &= f'_\mu(x_0) f'_\mu(x_1). \end{aligned}$$

The two equations of (13.9) are therefore identical, and the period-2 cycle will go unstable at the value of μ satisfying

$$f'_\mu(x_0) f'_\mu(x_1) = -1,$$

where x_0 and x_1 are given by (13.8).

The equation for μ, then, is given by

$$\mu^2(1 - 2x_0)(1 - 2x_1) = -1,$$

or

$$\mu^2\big(1 - 2(x_0 + x_1) + 4x_0x_1\big) + 1 = 0. \tag{13.10}$$

Now, from (13.8),

$$x_0 + x_1 = \frac{\mu+1}{\mu}, \quad x_0x_1 = \frac{\mu+1}{\mu^2}. \tag{13.11}$$

Substitution of (13.11) into (13.10) results, after simplification, in the quadratic equation

$$\mu^2 - 2\mu - 5 = 0,$$

with the only positive solution given by

$$\begin{aligned} \mu &= 1 + \sqrt{6} \\ &\approx 3.449490. \end{aligned}$$

We therefore expect the period-2 cycle to bifurcate to a period-4 cycle at $\mu \approx 3.449490$.

n	μ_n
1	3
2	3.449490...
3	3.544090...
4	3.564407...
5	3.568759...
6	3.569692...
7	3.569891...
8	3.569934...

Table 13.1: The first eight values of μ_n at which period-doubling bifurcations occur.

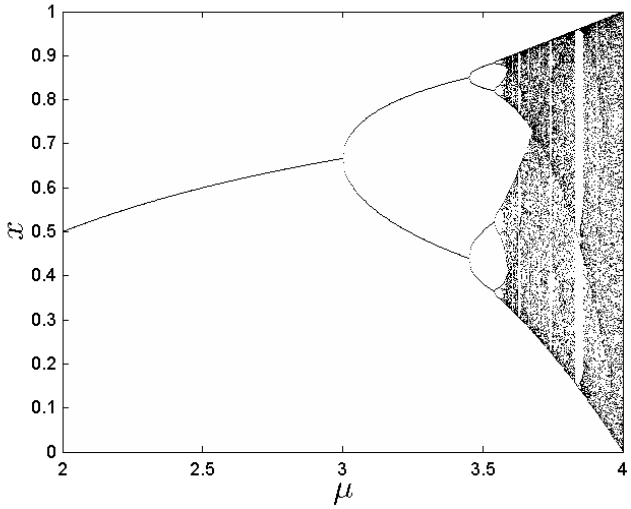


Figure 13.10: *Bifurcation diagram for period-doubling in the logistic map.*

If we define μ_n to be the value of μ at which the period-2^{n-1} cycle bifurcates to a period-2^n cycle, then we have determined analytically that $\mu_1 = 3$ and $\mu_2 = 1 + \sqrt{6}$. We list in Table 13.1, the first eight approximate values of μ_n, computed numerically.

We can compute a bifurcation diagram for the logistic map. For $2 < \mu < 4$, we plot the iterates from the map, discarding initial transients. The bifurcation diagram is shown in Fig. 13.10. Notice the uncanny similarity between the bifurcation diagram for the logistic map, and the one we have previously computed for the damped, driven pendulum equation, Fig. 13.9. Also note that the computation of Fig. 13.10, being on the order of seconds, is substantially faster than that of Fig. 13.9, which took about an hour, because a one-dimensional map is much faster to compute than the Poincaré section of a pair of coupled first-order differential equations.

13.6 Computation of the Feigenbaum constant

Period doubling in the logistic map enables an accurate computation of the Feigenbaum constant δ, defined as

$$\delta = \lim_{n\to\infty} \delta_n, \tag{13.12}$$

where

$$\delta_n = \frac{\mu_{n+1} - \mu_n}{\mu_{n+2} - \mu_{n+1}}. \tag{13.13}$$

Table 13.1 lists the known first eight values of μ_n at the bifurcation points. These values, and those at even larger values of n, are in fact very difficult to compute with high precision because of the slow convergence of the iterates at the bifurcation points. Rather, we will instead compute the values of μ at what are called superstable cycles. This now well-known method for computing δ was first described by Keith Briggs (1989).

Recall that for the general one-dimensional map

$$x_{n+1} = f_\mu(x_n),$$

a perturbation ϵ_n to a fixed point x_* decays as

$$\epsilon_{n+1} = f'_\mu(x_*)\epsilon_n.$$

At a so-called superstable fixed point, however, $f'_\mu(x_*) = 0$ and the perturbation decays very much faster as

$$\epsilon_{n+1} = \frac{1}{2} f''_\mu(x_*)\epsilon_n^2.$$

What are the superstable fixed points of the logistic map? Now, the values x_i that are in the orbit of a period-2^n cycle are fixed points of the composite map

$$x_{n+1} = g_\mu(x_n), \tag{13.14}$$

where $g_\mu = f_\mu \circ f_\mu \circ \cdots \circ f_\mu$, where the composition is repeated 2^n times. The orbit of a superstable cycle, then, consists of superstable fixed points of (13.14). If the period-2^n cycle has orbit $x_0, x_1, \ldots, x_{2^n-1}$, we have $f_\mu(x_0) = x_1$, $f_\mu(x_1) = x_2$, $\ldots$, $f_\mu(x_{2^n-1}) = x_0$, and by the chain rule,

$$g'_\mu(x_i) = f'_\mu(x_0) f'_\mu(x_1) \cdots f'_\mu(x_{2^n-1}),$$

for all x_i in the orbit of the period-2^n cycle. With

$$f'_\mu(x) = \mu(1 - 2x),$$

we have $g'_\mu(x) = 0$ for $x = 1/2$. Therefore, if $x_0 = 1/2$, say, is in the orbit of a period-2^n cycle, then this cycle is superstable.

At the bifurcation point creating a period-2^n cycle, the cycle has marginal stability and $g'_\mu(x_0) = 1$. As μ increases, $g'_\mu(x_0)$ decreases, and eventually the period-2^n cycle loses stability when $g'_\mu(x_0) = -1$. At some intermediate value of μ, then, there exists a value of x_0 with $g'_\mu(x_0) = 0$, and here we may assign $x_0 = 1/2$. Therefore, every period-2^n cycle contains a value of μ for which $x_0 = 1/2$ is in the orbit of the cycle.

Accordingly, we define m_n to be the value of μ at which $x_0 = 1/2$ is in the orbit of the period-2^n cycle. We can modify the definition of the Feigenbaum constant (13.13) to be

$$\delta_n = \frac{m_{n+1} - m_n}{m_{n+2} - m_{n+1}}. \tag{13.15}$$

Though the values of δ_n computed from (13.13) and (13.15) will differ slightly, the values as $n \to \infty$ should be the same.

We can easily determine the first two values of m_n. For the period-1 cycle, we have

$$\frac{1}{2} = m_0 \frac{1}{2}\left(1 - \frac{1}{2}\right),$$

or $m_0 = 2$, as confirmed from Fig. 13.10. To determine m_1, we make use of the period-2 cycle given by (13.7), and Fig. (13.10), which shows that the smaller root passes through $x_0 = 1/2$. Therefore,

$$\frac{1}{2} = \frac{1}{2m_1}\left((m_1+1) - \sqrt{(m_1+1)(m_1-3)}\right);$$

and solving for m_1, we obtain the quadratic equation

$$m_1^2 - 2m_1 - 4 = 0,$$

with solution $m_1 = 1 + \sqrt{5} \approx 3.2361$. Further values of m_n will be computed numerically.

To determine m_n, we need to solve the equation $G(\mu) = 0$, where

$$G(\mu) = g_\mu(1/2) - \frac{1}{2}, \tag{13.16}$$

and where as before, $g_\mu(x)$ is the composition of $f_\mu(x)$ repeated 2^n times. The roots of (13.16) are given by $m_0, m_1, \ldots m_n$, so that the desired root is the largest one.

We shall use Newton's method, §2.2, to solve (13.16). To implement Newton's method, we need to compute both $G(\mu)$ and $G'(\mu)$. Define $N = 2^n$. Then using the logistic map

$$x_{n+1} = \mu x_n(1 - x_n), \tag{13.17}$$

and iterating with $x_0 = 1/2$, we obtain $x_1, x_2, \ldots, x_N$. This orbit is superstable if $x_N = 1/2$. Therefore, we have

$$G(\mu) = x_N - 1/2.$$

Moreover,

$$G'(\mu) = x'_N,$$

where the derivative is with respect to μ. From (13.17), we have

$$\begin{aligned} x'_{n+1} &= x_n(1 - x_n) + \mu x'_n(1 - x_n) - \mu x_n x'_n \\ &= x_n(1 - x_n) + \mu x'_n(1 - 2x_n). \end{aligned}$$

Since we always choose $x_0 = 1/2$, independent of μ, we have as a starting value $x'_0 = 0$. Therefore, to compute both x_N and x'_N, we iterate 2^n times the coupled map equations

$$\begin{aligned} x_{n+1} &= \mu x_n(1 - x_n), \\ x'_{n+1} &= x_n(1 - x_n) + \mu x'_n(1 - 2x_n), \end{aligned}$$

n	m_n	δ_n
0	2	4.7089430135405
1	$1+\sqrt{5}$	4.6807709980107
2	3.4985616993277	4.6629596111141
3	3.5546408627688	4.6684039259164
4	3.5666673798563	4.6689537409802
5	3.5692435316371	4.6691571813703
6	3.5697952937499	4.6691910014915
7	3.5699134654223	4.6691994819801
8	3.5699387742333	4.6692010884670
9	3.5699441946081	4.6692015881423
10	3.5699453554865	4.6692023902759
11	3.5699456041111	4.6691974782669
12	3.5699456573588	4.6693329633696
13	3.5699456687629	
14	3.5699456712052	

Table 13.2: The first fourteen values of m_n, and estimates of the Feigenbaum delta.

with initial values $x_0 = 1/2$ and $x_0' = 0$. Newton's method then solves for m_n by iterating

$$\mu^{(i+1)} = \mu^{(i)} - \frac{x_N - 1/2}{x_N'},$$

until convergence of $\mu^{(i)}$ to m_n. In double precision, we have been able to achieve a precision of about 14 digits, which we find can be obtained in fewer than 5 iterations of Newton's method.

For Newton's method to converge to m_n, we need a good guess for $\mu^{(0)}$. We can use the previous best estimate for the Feigenbaum delta to predict m_n. From (13.15), we find

$$\mu^{(0)} = m_{n-1} + \frac{m_{n-1} - m_{n-2}}{\delta_{n-2}}.$$

Although we can not compute δ_{n-2} without knowing m_n, we can nevertheless use the estimate $\delta_{n-2} \approx \delta_{n-3}$. The computation, then, starts with $n = 2$, and we can begin by taking $\delta_{-1} = 4.4$, so that, for example,

$$\begin{aligned} \mu^{(0)} &= 3.2361 + \frac{3.2361 - 2}{4.4} \\ &= 3.5170. \end{aligned}$$

Using this algorithm, we have produced Table 13.2 for m_n, with corresponding calculations of δ_n. As the values of m_n converge, the corresponding values of δ_n begin to lose precision. It would appear that our best estimate from the table is $\delta \approx 4.66920$, compared to the known value of $\delta = 4.669201609102990\ldots$, computed by a different algorithm capable of achieving higher precision.

13.7 Strange attractor of the chaotic pendulum

After the period-doubling cascade, the pendulum motion becomes chaotic. We choose parameter values $q = 4$, $f = 1.5$, and $\omega = 2/3$ in the chaotic regime, and after discarding an initial transient

of 256 forcing periods, compute a Poincaré section of the phase-space trajectory in the θ-$\dot{\theta}$ plane, sampling points every forcing period. The values of the periodic variable θ are mapped onto the interval $-\pi < \theta < \pi$. The full Poincaré section consisting of 50,000 points is shown in the top drawing of Fig. 13.11, and a blowup of the points within the drawn rectangle (from a sample of 200,000 points over the entire attractor) is shown in the bottom drawing.

From Fig. 13.11, it appears that the Poincaré section has structure on all scales, which is reminiscent of the classical fractals discussed in §12.7. The set of points shown in Fig. 13.11 is called a strange attractor, and will be seen to have a fractional correlation dimension.

Using the algorithm for computing a correlation dimension discussed in §12.7, we draw a log-log plot of the correlation integral $C(r)$ versus r, shown in Fig. 13.12. A least-squares fit of a straight line to the middle region of the plot yields a correlation dimension for the Poincaré section of approximately $D = 1.25$.

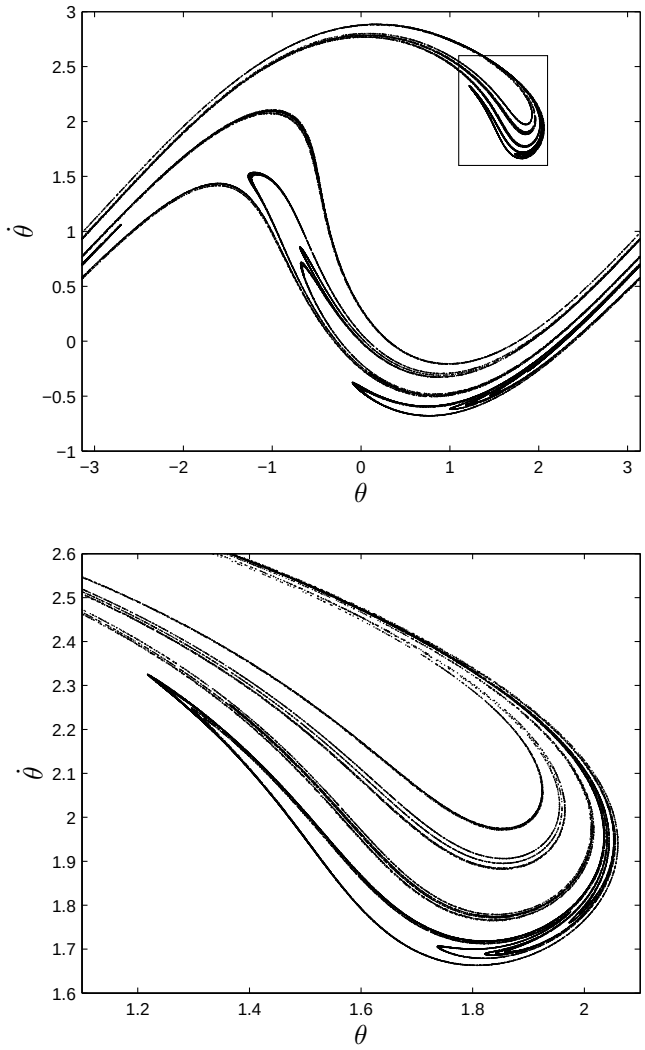


Figure 13.11: *A Poincaré section of the chaotic pendulum. The values of the parameters are* $q = 4$, $f = 1.5$, *and* $\omega = 2/3$. *The top figure shows the entire attractor and the bottom figure is a blow-up of the region inside the drawn rectangle.*

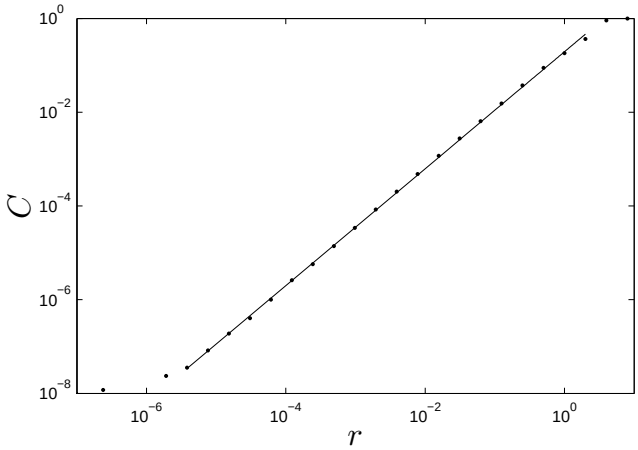


Figure 13.12: *The correlation integral $C(r)$ versus r for the strange attractor shown in Fig. 13.11, using a 13,000 point sample. The least-squares line on the log-log plot yields a correlation dimension of the Poincaré section of approximately $D = 1.25$.*

Part III

Computational fluid dynamics

The third part of this course considers a problem in computational fluid dynamics (cfd). Namely, we consider the steady two-dimensional flow past a rectangle or a circle.

Chapter 14

Derivation of the governing equations

We derive here the governing equations for the velocity $\mathbf{u} = \mathbf{u}(\mathbf{x}, t)$ of a flowing fluid.

14.1 Multi-variable calculus

Fluid flows typically take place in three-dimensional space, and the governing equations will contain derivatives in all three directions. The mathematics learned in a multi-variable calculus course will therefore be useful. Here, I summarize some of this mathematics.

14.1.1 Vector algebra

Examples of vectors will be the position vector $\mathbf{x}$ and the velocity vector $\mathbf{u}$. We will use the Cartesian coordinate system to write vectors in terms of their components as

$$\mathbf{x} = (x, y, z), \quad \mathbf{u} = (u, v, w),$$

or sometimes as

$$\mathbf{x} = (x_1, x_2, x_3), \quad \mathbf{u} = (u_1, u_2, u_3).$$

Another notation makes use of the cartesian unit vectors, $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$, and $\hat{\mathbf{z}}$:

$$\mathbf{x} = x\hat{\mathbf{x}} + y\hat{\mathbf{y}} + z\hat{\mathbf{z}}, \quad \mathbf{u} = u\hat{\mathbf{x}} + v\hat{\mathbf{y}} + w\hat{\mathbf{z}}.$$

The velocity $\mathbf{u}$ is called a vector field because it is a vector that is a function of the position vector $\mathbf{x}$.

The dot product between two vectors $\mathbf{u}$ and $\mathbf{v}$ is given by

$$\begin{aligned}\mathbf{u} \cdot \mathbf{v} &= u_1 v_1 + u_2 v_2 + u_3 v_3 \\ &= u_i v_i,\end{aligned}$$

where in the last expression we use the Einstein summation convention: when an index occurs twice in a single term, it is summed over. From hereon, the Einstein summation convention will be assumed unless explicitly stated otherwise.

The cross product between two vectors is given by a determinant:

$$\begin{aligned}\mathbf{u} \times \mathbf{v} &= \begin{vmatrix} \hat{\mathbf{x}} & \hat{\mathbf{y}} & \hat{\mathbf{z}} \\ u_1 & u_2 & u_3 \\ v_1 & v_2 & v_3 \end{vmatrix} \\ &= (u_2 v_3 - u_3 v_2, u_3 v_1 - u_1 v_3, u_1 v_2 - u_2 v_1).\end{aligned}$$

The cross-product of two vectors is a vector, and the components of the cross product can be written more succinctly using the Levi-Civita tensor, defined as

$$\epsilon_{ijk} = \begin{cases} 1 & \text{if } (i,j,k) \text{ is an even permutation of } (1,2,3), \\ -1 & \text{if } (i,j,k) \text{ is an odd permutation of } (1,2,3), \\ 0 & \text{if any index is repeated.} \end{cases}$$

Using the Levi-Civita tensor, the i-th component of the cross product can be written as

$$(\mathbf{u} \times \mathbf{v})_i = \epsilon_{ijk} u_j v_k.$$

Another useful tensor is the Kronecker delta, defined as

$$\delta_{ij} = \begin{cases} 0 & \text{if } i \neq j, \\ 1 & \text{if } i = j. \end{cases}$$

Note that $v_i \delta_{ij} = v_j$ and that $\delta_{ii} = 3$. A useful identity between the Levi-Civita tensor and the Kronecker delta is given by

$$\epsilon_{ijk} \epsilon_{imn} = \delta_{jm} \delta_{kn} - \delta_{jn} \delta_{km}.$$

Gauss's theorem (or the divergence theorem) and Stokes' theorem are usually introduced in a course on multi-variable calculus. We will state these theorems here.

First, Gauss's theorem. Let V be a three-dimensional volume bounded by a smooth surface S, and let $\mathbf{F}$ be a vector field in V. Then Gauss's theorem states that

$$\int_S \mathbf{F} \cdot \hat{\mathbf{n}} dS = \int_V \boldsymbol{\nabla} \cdot \mathbf{F} dV, \tag{14.1}$$

where $\hat{\mathbf{n}}$ is the outward facing unit normal vector to the bounding surface S.

Second, Stokes' theorem. Let S be a smooth surface bounded by a simple closed curve C with positive orientation. Then Stokes' theorem states that

$$\oint_C \mathbf{F} \cdot d\mathbf{r} = \int_S (\boldsymbol{\nabla} \times \mathbf{F}) \cdot \hat{\mathbf{n}} dS. \tag{14.2}$$

14.2 Continuity equation

We consider a control volume V of fluid bounded by a smooth surface S. The continuity equation expresses the conservation of mass. The time-derivative of the total mass of the fluid contained in the volume V is equal to the (negative) of the total mass of fluid that flows out of the boundary of V; that is;

$$\frac{d}{dt} \int_V \rho(\mathbf{x}, t) dV = - \int_S \rho(\mathbf{x}, t) \mathbf{u}(\mathbf{x}, t) \cdot \hat{\mathbf{n}} dS. \tag{14.3}$$

The integral on the right-hand-side represents the flux of mass through the boundary S and has units of mass per unit time. We now apply the divergence theorem to the integral on the right-hand-side:

$$\int_S \rho(\mathbf{x}, t) \mathbf{u}(\mathbf{x}, t) \cdot \hat{\mathbf{n}} dS = \int_V \boldsymbol{\nabla} \cdot (\rho \mathbf{u}) dV. \tag{14.4}$$

Combining the left- and right-hand-sides, we have

$$\int_V \left(\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \mathbf{u}) \right) dV = 0. \tag{14.5}$$

Because the control volume is arbitrary, the integrand must vanish identically, and we thus obtain the continuity equation

$$\frac{\partial \rho}{\partial t} + \boldsymbol{\nabla} \cdot (\rho \mathbf{u}) = 0. \tag{14.6}$$

We will only be considering here incompressible fluids, where we may assume that $\rho(\mathbf{x}, t)$ is a constant, independent of both space and time. The continuity equation (14.6) then becomes an equation for conservation of fluid volume, and is given by

$$\boldsymbol{\nabla} \cdot \mathbf{u} = 0. \tag{14.7}$$

This equation is called the incompressibility condition.

14.3 Momentum equation

14.3.1 Material derivative

The Navier-Stokes equation is derived from applying Newton's law $F = ma$ to a fluid flow. We first consider the acceleration of a fluid element. The velocity of the fluid at a fixed position $\mathbf{x}$ is given by $\mathbf{u}(\mathbf{x}, t)$, but the fluid element is not at a fixed position but follows the fluid in motion. Now a general application of the chain rule yields

$$\frac{d}{dt}\mathbf{u}(\mathbf{x}, t) = \frac{\partial \mathbf{u}}{\partial t} + \frac{\partial \mathbf{u}}{\partial x_j}\frac{\partial x_j}{\partial t}.$$

If the position $\mathbf{x}$ is fixed, then $\partial x_j/\partial t = 0$. But if $\mathbf{x} = \mathbf{x}(\mathbf{t})$ represents the position of the fluid element, then $\partial x_j/\partial t = u_j$. The latter assumption is called the material derivative and is written as

$$\begin{aligned}\frac{D\mathbf{u}}{Dt} &= \frac{\partial \mathbf{u}}{\partial t} + u_j\frac{\partial \mathbf{u}}{\partial x_j} \\ &= \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\boldsymbol{\nabla})\mathbf{u},\end{aligned}$$

and represents the acceleration of a fluid element as it flows with the fluid. Instead of the mass of the fluid element, we consider the mass per unit volume, and the right-hand-side of $F = ma$ becomes

$$\rho\left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u}\cdot\boldsymbol{\nabla})\mathbf{u}\right).$$

We now need find the forces per unit volume acting on the flowing fluid element. We consider both pressure forces and viscous forces.

14.3.2 Pressure forces

We consider the normal pressure forces acting on two opposing faces of a control volume of fluid. With A the area of the rectangular face at fixed y, and dy the depth, the volume of the box is Ady, and the net pressure force per unit volume acting on the control volume in the y-direction is given by

$$\begin{aligned}f_p &= \frac{pA - (p+dp)A}{Ady} \\ &= -\frac{dp}{dy}.\end{aligned}$$

Similar considerations for the x and z directions yield the pressure force vector per unit volume to be

$$\begin{aligned}\mathbf{f}_p &= -\left(\frac{\partial p}{\partial x}, \frac{\partial p}{\partial y}, \frac{\partial p}{\partial z}\right) \\ &= -\boldsymbol{\nabla} p.\end{aligned}$$

14.3.3 Viscous forces

The viscosity of a fluid measures its internal resistance to flow. Consider a fluid confined between two very large plates of surface area A, separated by a small distance dy. Suppose the bottom

plate is stationary and the top plate move with velocity du in the x- direction. The applied force per unit area required to keep the top surface in motion is empirically given by

$$\frac{F}{A} = \mu \frac{du}{dy},$$

where μ is called the dynamic viscosity. Of course there is also an opposite force required to keep the bottom surface stationary. The difference between these two forces is the net viscous force on the fluid element. Taking the difference, the resulting net force per unit area will be proportional to the second derivative of the velocity. Now the viscous forces act in all directions and on all the faces of the control volume. Without going into further technical details, we present the general form (for a so-called Newtonian fluid) of the viscous force vector per unit volume:

$$\begin{aligned}\mathbf{f}_v &= \mu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2}, \frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} + \frac{\partial^2 v}{\partial z^2}, \frac{\partial^2 w}{\partial x^2} + \frac{\partial^2 w}{\partial y^2} + \frac{\partial^2 w}{\partial z^2} \right) \\ &= \mu \nabla^2 \mathbf{u}.\end{aligned}$$

14.3.4 Navier-Stokes equation

Putting together all the terms, the Navier-Stokes equation is written as

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \boldsymbol{\nabla}) \mathbf{u} \right) = -\boldsymbol{\nabla} p + \mu \nabla^2 \mathbf{u}.$$

Now, instead of the dynamic viscosity μ, one usually defines the kinematic viscosity $\nu = \mu/\rho$. The governing equations of fluid mechanics for a so-called incompressible Newtonian fluid, then, are given by both the continuity equation and the Navier-Stokes equation; that is,

$$\boldsymbol{\nabla} \cdot \mathbf{u} = 0, \tag{14.8}$$

$$\frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \boldsymbol{\nabla}) \mathbf{u} = -\frac{1}{\rho} \boldsymbol{\nabla} p + \nu \nabla^2 \mathbf{u}. \tag{14.9}$$

14.3.5 Boundary conditions

Boundary conditions must be prescribed when flows contact solid surfaces. We will assume rigid, impermeable surfaces. If $\hat{\mathbf{n}}$ is the normal unit vector to the surface, and if there is no motion of the surface in the direction of its normal vector, then the condition of impermeability yields

$$\mathbf{u} \cdot \hat{\mathbf{n}} = 0.$$

We will also assume the no-slip condition: a viscous fluid should have zero velocity relative to a solid surface. In other words, a stationary or moving solid surface drags along the fluid touching it with the same velocity. The no-slip condition can be expressed mathematically as

$$\mathbf{u} \times \hat{\mathbf{n}} = \mathbf{V} \times \hat{\mathbf{n}},$$

where $\mathbf{u}$ is the velocity of the fluid, $\mathbf{V}$ is the velocity of the surface, and $\hat{\mathbf{n}}$ is the normal vector to the surface.

Boundary conditions may also be prescribed far from any wall or obstacle. The free-stream boundary condition states that $\mathbf{u} = \mathbf{U}$ at infinity, where $\mathbf{U}$ is called the free-stream velocity.

Chapter 15

Laminar flow

Smoothly flowing fluids, with the fluid flowing in undisrupted layers, are called laminar flows. There are several iconic laminar flows, whose velocity fields are readily found by solving the continuity and Navier-Stokes equations.

15.1 Plane Couette flow

Plane Couette flow consists of a fluid flowing between two infinite plates separated by a distance d. The lower plate is stationary, and the upper plate is moving to the right with velocity U. The pressure p is constant and the fluid is incompressible.

We look for a steady solution for the velocity field of the form

$$\mathbf{u}(x, y, z) = \big(u(y), 0, 0\big).$$

The incompressibility condition is automatically satisfied, and the first-component of the Navier-Stokes equation reduces to

$$\nu \frac{\partial^2 u}{\partial y^2} = 0.$$

Applying the boundary conditions $u(0) = 0$ and $u(d) = U$ on the lower and upper plates, the laminar flow solution is given by

$$u(y) = \frac{Uy}{d}.$$

15.2 Channel flow

Channel flow, or Poiseuille flow, also consists of a fluid flowing between two infinite plates separated by a distance d, but with both plates stationary. Here, there is a constant pressure gradient along the x-direction in which the fluid flows. Again, we look for a steady solution for the velocity field of the form

$$\mathbf{u}(x, y, z) = \big(u(y), 0, 0\big),$$

and with

$$p = p(x)$$

and

$$\frac{dp}{dx} = -G, \tag{15.1}$$

with G a positive constant. The first-component of the Navier-Stokes equation becomes

$$-\frac{1}{\rho}\frac{dp}{dx} + \nu \frac{d^2 u}{dy^2} = 0. \tag{15.2}$$

Using (15.1) in (15.2) leads to

$$\frac{d^2 u}{dy^2} = -\frac{G}{\nu\rho},$$

which can be solved using the no-slip boundary conditions $u(0) = u(d) = 0$. We find

$$u(y) = \frac{Gd^2}{2\nu\rho}\left(\frac{y}{d}\right)\left(1 - \frac{y}{d}\right).$$

The maximum velocity of the fluid occurs at the midline, $y = d/2$, and is given by

$$u_{\max} = \frac{Gd^2}{8\nu\rho}.$$

15.3 Pipe flow

Pipe flow consists of flow through a pipe of circular cross-section radius R, with a constant pressure gradient along the pipe length. With the pressure gradient along the x-direction, we look for a steady solution of the velocity field of the the form

$$\mathbf{u} = \big(u(y,z), 0, 0\big).$$

With the constant pressure gradient defined as in (15.1), the Navier-Stokes equation reduces to

$$\frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 u}{\partial z^2} = -\frac{G}{\nu\rho}. \tag{15.3}$$

The use of polar coordinates in the y-z plane can aid in solving (15.3). With

$$u = u(r),$$

we have

$$\frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} = \frac{1}{r}\frac{d}{dr}\left(r\frac{d}{dr}\right),$$

so that (15.3) becomes the differential equation

$$\frac{d}{dr}\left(r\frac{du}{dr}\right) = -\frac{Gr}{\nu\rho},$$

with no-slip boundary condition $u(R) = 0$. The first integration from 0 to r yields

$$r\frac{du}{dr} = -\frac{Gr^2}{2\nu\rho};$$

and after division by r, the second integration from r to R yields

$$u(r) = \frac{GR^2}{4\nu\rho}\left(1 - \left(\frac{r}{R}\right)^2\right).$$

The maximum velocity occurs at the pipe midline, $r = 0$, and is given by

$$u_{\max} = \frac{GR^2}{4\nu\rho}.$$

Chapter 16

Stream function, vorticity equations

16.1 Stream function

A streamline at time t is defined as the curve whose tangent is everywhere parallel to the velocity vector. With $d\mathbf{x}$ along the tangent, we have

$$\mathbf{u} \times d\mathbf{x} = 0;$$

and with $\mathbf{u} = (u, v, w)$ and $d\mathbf{x} = (dx, dy, dz)$, the cross product yields the three equations

$$v dz = w dy, \quad u dz = w dx, \quad u dy = v dx, \tag{16.1}$$

or equivalently,

$$\frac{dx}{u} = \frac{dy}{v} = \frac{dz}{w}.$$

Streamlines have the following two properties. They cannot intersect except at a point of zero velocity, and as streamlines converge the fluid speed increases. The latter is a consequence of the incompressibility of the fluid: as the same flow rate of fluid passes through a smaller cross-sectional area, the fluid velocity must increase.

Related to streamlines is the stream function. We specialize here to a two dimensional flow, with

$$\mathbf{u} = \big(u(x,y), v(x,y), 0\big).$$

The incompressibility condition becomes

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0,$$

which can be satisfied by defining the scalar stream function $\psi = \psi(x, y)$ by

$$u(x,y) = \frac{\partial \psi}{\partial y}, \quad v(x,y) = -\frac{\partial \psi}{\partial x}.$$

Now, the differential of the stream function $\psi(x, y)$ satisfies

$$\begin{aligned} d\psi &= \frac{\partial \psi}{\partial x} dx + \frac{\partial \psi}{\partial y} dy \\ &= -v dx + u dy, \end{aligned}$$

which from (16.1) is equal to zero along streamlines. Thus the contour curves of constant ψ represent the streamlines of the flow field, and can provide a good visualization of a fluid flow in two dimensions.

16.2 Vorticity

The vector vorticity field is defined from the vector velocity field by

$$\boldsymbol{\omega} = \boldsymbol{\nabla} \times \mathbf{u}. \tag{16.2}$$

The vorticity is a measure of the local rotation of the fluid as can be seen from an application of Stokes' theorem:

$$\begin{aligned}\int_S \boldsymbol{\omega} \cdot \hat{\mathbf{n}} dS &= \int_S (\boldsymbol{\nabla} \times \mathbf{u}) \cdot \hat{\mathbf{n}} dS \\ &= \oint_C \mathbf{u} \cdot d\mathbf{r}.\end{aligned}$$

Flows without vorticity are called irrotational, or potential flow, and vorticity is sometimes called swirl.

The governing equation for vorticity may be found by taking the curl of the Navier-Stokes equation; that is,

$$\boldsymbol{\nabla} \times \left\{ \frac{\partial \mathbf{u}}{\partial t} + (\mathbf{u} \cdot \boldsymbol{\nabla})\mathbf{u} \right\} = \boldsymbol{\nabla} \times \left\{ -\frac{1}{\rho} \boldsymbol{\nabla} p + \nu \nabla^2 \mathbf{u} \right\}.$$

Computing term-by-term, we have

$$\begin{aligned}\boldsymbol{\nabla} \times \left\{ \frac{\partial \mathbf{u}}{\partial t} \right\} &= \frac{\partial}{\partial t} (\boldsymbol{\nabla} \times \mathbf{u}) \\ &= \frac{\partial \boldsymbol{\omega}}{\partial t}.\end{aligned}$$

And because the curl of a gradient is zero,

$$\boldsymbol{\nabla} \times \left\{ -\frac{1}{\rho} \boldsymbol{\nabla} p \right\} = 0.$$

Also,

$$\begin{aligned}\boldsymbol{\nabla} \times \left\{ \nu \nabla^2 \mathbf{u} \right\} &= \nu \nabla^2 (\boldsymbol{\nabla} \times \mathbf{u}) \\ &= \nu \nabla^2 \boldsymbol{\omega}.\end{aligned}$$

The remaining term to compute is the curl of the convection term in the Navier-Stokes equation. We first consider the following equality (where the subscript i signifies the i-th component of the vector):

$$\begin{aligned}\{\mathbf{u} \times (\boldsymbol{\nabla} \times \mathbf{u})\}_i &= \epsilon_{ijk} u_j \epsilon_{klm} \frac{\partial u_m}{\partial x_l} \\ &= \epsilon_{kij} \epsilon_{klm} u_j \frac{\partial u_m}{\partial x_l} \\ &= (\delta_{il}\delta_{jm} - \delta_{im}\delta_{jl}) u_j \frac{\partial u_m}{\partial x_l} \\ &= u_m \frac{\partial u_m}{\partial x_i} - u_l \frac{\partial u_i}{\partial x_l} \\ &= \frac{1}{2} \frac{\partial}{\partial x_i} u_m u_m - u_l \frac{\partial u_i}{\partial x_l}.\end{aligned}$$

Therefore, in vector form,

$$\mathbf{u} \times (\boldsymbol{\nabla} \times \mathbf{u}) = \frac{1}{2}\boldsymbol{\nabla}(\mathbf{u}^2) - (\mathbf{u} \cdot \boldsymbol{\nabla})\mathbf{u}.$$

This identity allows us to write

$$(\mathbf{u} \cdot \boldsymbol{\nabla})\mathbf{u} = \frac{1}{2}\boldsymbol{\nabla}(\mathbf{u}^2) - \mathbf{u} \times (\boldsymbol{\nabla} \times \mathbf{u}).$$

Taking the curl of both sides and making use of the curl of a gradient equals zero and $\boldsymbol{\nabla} \times \mathbf{u} = \boldsymbol{\omega}$, results in

$$\begin{aligned} \boldsymbol{\nabla} \times \{(\mathbf{u} \cdot \boldsymbol{\nabla})\mathbf{u}\} &= -\boldsymbol{\nabla} \times (\mathbf{u} \times \boldsymbol{\omega}) \\ &= \boldsymbol{\nabla} \times (\boldsymbol{\omega} \times \mathbf{u}) \,. \end{aligned}$$

Combining all the above terms, we have thus obtained the vorticity equation

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + \boldsymbol{\nabla} \times (\boldsymbol{\omega} \times \mathbf{u}) = \nu \nabla^2 \boldsymbol{\omega}. \tag{16.3}$$

An alternative form of the vorticity equation rewrites the convection term to explicitly include the substantive derivative. We have

$$\begin{aligned} \{\boldsymbol{\nabla} \times (\boldsymbol{\omega} \times \mathbf{u})\}_i &= \epsilon_{ijk} \frac{\partial}{\partial x_j} \epsilon_{klm} \omega_l u_m \\ &= \epsilon_{kij}\epsilon_{klm} \frac{\partial}{\partial x_j}(\omega_l u_m) \\ &= (\delta_{il}\delta_{jm} - \delta_{im}\delta_{jl}) \frac{\partial}{\partial x_j}(\omega_l u_m) \\ &= \frac{\partial}{\partial x_m}(\omega_i u_m) - \frac{\partial}{\partial x_l}(\omega_l u_i) \\ &= u_m \frac{\partial w_i}{\partial x_m} - \omega_l \frac{\partial u_i}{\partial x_l}, \end{aligned}$$

where to obtain the last equality we have used both $\partial u_m / \partial x_m = 0$ and $\partial \omega_l / \partial x_l = 0$. Therefore, in vector form,

$$\boldsymbol{\nabla} \times (\boldsymbol{\omega} \times \mathbf{u}) = (\mathbf{u} \cdot \boldsymbol{\nabla})\boldsymbol{\omega} - (\boldsymbol{\omega} \cdot \boldsymbol{\nabla})\mathbf{u}.$$

The vorticity equation can then be rewritten as

$$\frac{\partial \boldsymbol{\omega}}{\partial t} + (\mathbf{u} \cdot \boldsymbol{\nabla})\boldsymbol{\omega} = (\boldsymbol{\omega} \cdot \boldsymbol{\nabla})\mathbf{u} + \nu \nabla^2 \boldsymbol{\omega}. \tag{16.4}$$

Compared to the Navier-Stokes equation, there is an extra term, called the vortex stretching term, on the right-hand-side of (16.4).

16.3 Two-dimensional Navier-Stokes equation

We have already seen that in two dimensions, the incompressibility condition is automatically satisfied by defining the stream function $\psi(\mathbf{x}, t)$. Also in two dimensions, the vorticity can be

reduced to a scalar field. With $\mathbf{u} = \big(u(x,y), v(x,y)\big)$, we have

$$\begin{aligned}\boldsymbol{\omega} &= \boldsymbol{\nabla} \times \mathbf{u} \\ &= \begin{vmatrix} \hat{\mathbf{x}} & \hat{\mathbf{y}} & \hat{\mathbf{z}} \\ \partial/\partial x & \partial/\partial y & \partial/\partial z \\ u(x,y) & v(x,y) & 0 \end{vmatrix} \\ &= \hat{\mathbf{z}} \left(\frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \right) \\ &= \omega(x,y)\hat{\mathbf{z}},\end{aligned}$$

where we have now defined the scalar field ω to be the third component of the vector vorticity field. Making use of the stream function, we then have

$$\begin{aligned}\omega &= \frac{\partial v}{\partial x} - \frac{\partial u}{\partial y} \\ &= -\frac{\partial^2 \psi}{\partial x^2} - \frac{\partial^2 \psi}{\partial y^2}.\end{aligned}$$

Therefore, in vector form, we have

$$\nabla^2 \psi = -\omega, \tag{16.5}$$

where

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$$

is the two-dimensional Laplacian.

Now, with $\boldsymbol{\omega} = \omega(x,y)\hat{\mathbf{z}}$, the third component of the vorticity equation (16.4) becomes

$$\frac{\partial \omega}{\partial t} + (\mathbf{u} \cdot \boldsymbol{\nabla})\omega = \nu \nabla^2 \omega,$$

where the vortex stretching term can be seen to vanish. We can also write

$$\begin{aligned}\mathbf{u} \cdot \boldsymbol{\nabla} &= u \frac{\partial}{\partial x} + v \frac{\partial}{\partial y} \\ &= \frac{\partial \psi}{\partial y} \frac{\partial}{\partial x} - \frac{\partial \psi}{\partial x} \frac{\partial}{\partial y}.\end{aligned}$$

The vorticity equation in two dimensions then becomes

$$\frac{\partial \omega}{\partial t} + \left(\frac{\partial \psi}{\partial y} \frac{\partial \omega}{\partial x} - \frac{\partial \psi}{\partial x} \frac{\partial \omega}{\partial y} \right) = \nu \nabla^2 \omega.$$

For a stationary flow, this equation becomes the Poisson equation,

$$\nabla^2 \omega = \frac{1}{\nu} \left(\frac{\partial \psi}{\partial y} \frac{\partial \omega}{\partial x} - \frac{\partial \psi}{\partial x} \frac{\partial \omega}{\partial y} \right). \tag{16.6}$$

We have thus obtained for a stationary flow two coupled Poisson equations for $\psi(x,y)$ and $\omega(x,y)$ given by (16.5) and (16.6).

The pressure field $p(x,y)$ decouples from these two Poisson equations, but can be determined if desired. We take the divergence of the Navier-Stokes equation, (14.9), and application of the incompressibility condition, (14.8), yields

$$\boldsymbol{\nabla} \cdot \frac{\partial \mathbf{u}}{\partial t} = 0, \quad \boldsymbol{\nabla} \cdot \nabla^2 \mathbf{u} = 0,$$

resulting in

$$\nabla^2 p = -\rho \boldsymbol{\nabla} \cdot \big((\mathbf{u} \cdot \boldsymbol{\nabla}) \mathbf{u} \big). \tag{16.7}$$

We would like to eliminate the velocity vector field in (16.7) in favor of the stream function. We compute

$$\begin{aligned} \boldsymbol{\nabla} \cdot (\mathbf{u} \cdot \boldsymbol{\nabla}) \mathbf{u} &= \frac{\partial}{\partial x_i} \left(u_j \frac{\partial u_i}{\partial x_j} \right) \\ &= \frac{\partial u_j}{\partial x_i} \frac{\partial u_i}{\partial x_j} + u_j \frac{\partial^2 u_i}{\partial x_i \partial x_j}. \end{aligned}$$

The second term on the right-hand-side vanishes because of the incompressibility condition. We therefore have

$$\begin{aligned} \boldsymbol{\nabla} \cdot (\mathbf{u} \cdot \boldsymbol{\nabla}) \mathbf{u} &= \frac{\partial u_j}{\partial x_i} \frac{\partial u_i}{\partial x_j} \\ &= \left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial v}{\partial x} \frac{\partial u}{\partial y} \right) + \left(\frac{\partial u}{\partial y} \frac{\partial v}{\partial x} \right) + \left(\frac{\partial v}{\partial y} \right)^2 \\ &= \left(\frac{\partial^2 \psi}{\partial x \partial y} \right)^2 - 2 \frac{\partial^2 \psi}{\partial x^2} \frac{\partial^2 \psi}{\partial y^2} + \left(\frac{\partial^2 \psi}{\partial x \partial y} \right)^2 \\ &= -2 \left[\frac{\partial^2 \psi}{\partial x^2} \frac{\partial^2 \psi}{\partial y^2} - \left(\frac{\partial^2 \psi}{\partial x \partial y} \right)^2 \right]. \end{aligned}$$

Using (16.7), the pressure field therefore satisfies the Poisson equation

$$\nabla^2 p = 2\rho \left[\frac{\partial^2 \psi}{\partial x^2} \frac{\partial^2 \psi}{\partial y^2} - \left(\frac{\partial^2 \psi}{\partial x \partial y} \right)^2 \right].$$

Chapter 17

Steady, two-dimensional flow past an obstacle

We now consider a classic problem in computational fluid dynamics: the steady, two-dimensional flow past an obstacle. Here, we consider flows past a rectangle and a circle.

First, we consider flow past a rectangle. The simple Cartisian coordinate system is most suitable for this problem, and the boundaries of the internal rectangle can be aligned with the computational grid. Second, we consider flow past a circle. Here, the polar coordinate system is most suitable, and this introduces some additional analytical complications to the problem formulation. Nevertheless, we will see that the computation of flow past a circle may in fact be simpler than flow past a rectangle. Although flow past a rectangle contains two dimensionless parameters, flow past a circle contains only one. Furthermore, flow past a circle may be solved within a rectangular domain having no internal boundaries.

17.1 Flow past a rectangle

The free stream velocity is given by $\mathbf{u} = U\hat{\mathbf{x}}$ and the rectangular obstacle is assumed to have width W and height H. Now, the stream function has units of velocity times length, and the vorticity has units of velocity divided by length. The steady, two-dimensional Poisson equations for the stream function and the vorticity, given by (16.5) and (16.6), may be nondimensionalized using the velocity U and the length W. The resulting dimensionless equations can be written as

$$-\nabla^2\psi = \omega, \tag{17.1}$$

$$-\nabla^2\omega = \mathrm{Re}\left(\frac{\partial\psi}{\partial x}\frac{\partial\omega}{\partial y} - \frac{\partial\psi}{\partial y}\frac{\partial\omega}{\partial x}\right), \tag{17.2}$$

where the dimensionless parameter Re is called the Reynolds number. An additional dimensionless parameter arises from the aspect ratio of the rectangular obstacle, and is denoted by a. These two dimensionless parameters are defined by

$$\mathrm{Re} = \frac{UW}{\nu}, \quad a = \frac{W}{H}. \tag{17.3}$$

A solution for the scalar stream function and scalar vorticity field will be sought for different values of the Reynolds number Re at a fixed aspect ratio a.

17.1.1 Finite difference approximation

We construct a rectangular grid for a numerical solution. We will make use of square grid cells, and write

$$x_i = ih, \qquad i = 0, 1, \ldots, N_x; \tag{17.4a}$$

$$y_j = jh, \qquad j = 0, 1, \ldots, N_y, \tag{17.4b}$$

where N_x and N_y are the number of grid cells spanning the x- and y-directions, and h is the side length of a grid cell.

To obtain an accurate solution, we require the boundaries of the obstacle to lie exactly on the boundaries of the grid cell. The width of the obstacle in our dimensionless formulation is unity, and we place the front of the obstacle at $x = mh$ and the back of the obstacle at $x = (m + I)h$, where we must have

$$hI = 1.$$

With I specified, the grid spacing is determined by $h = 1/I$.

We will only look for steady solutions for the flow field that are symmetric about the midline of the obstacle. Assuming symmetry, we need only solve for the flow field in the upper half of the domain. We place the center line of the obstacle at $y = 0$ and the top of the rectangle at hJ. The dimensionless half-height of the obstacle is given by $1/2a$, so that

$$hJ = \frac{1}{2a}.$$

Forcing the rectangle to lie on the grid lines constrains the choice of aspect ratio and the values of I and J such that

$$a = \frac{I}{2J}.$$

Reasonable values of a to consider are $a = \ldots, 1/4, 1/2, 1, 2, 4, \ldots$, etc, and I and J can be adjusted accordingly.

The physics of the problem is specified through the two dimensionless parameters Re and a. The numerics of the problem is specified by the parameters N_x, N_y, h, and the placement of the rectangle in the computational domain. We look for convergence of the numerical solution as $h \to 0$, $N_x, N_y \to \infty$ and the rectangle is placed far from the boundaries of the computationally domain.

Discretizing the governing equations, we now write

$$\psi_{i,j} = \psi(x_i, y_j), \quad \omega_{i,j} = \omega(x_i, y_j).$$

To solve the coupled Poisson equations given by (17.1) and (17.2), we make use of the SOR method, previously described in §7.1. The notation we use here is for the Jacobi method, but faster convergence is likely to be achieved using red-back Gauss-Seidel. The Poisson equation for the stream function, given by (17.1), becomes

$$\psi_{i,j}^{n+1} = (1 - r_\psi)\psi_{i,j}^n + \frac{r_\psi}{4}\left(\psi_{i+1,j}^n + \psi_{i-1,j}^n + \psi_{i,j+1}^n + \psi_{i,j-1}^n + h^2\omega_{i,j}^n\right). \tag{17.5}$$

The Poisson equation for the vorticity, given by (17.2), requires use of the centered finite difference approximation for the derivatives that appear on the right-hand-side. For $x = x_i$, $y = y_i$, these approximations are given by

$$\frac{\partial\psi}{\partial x} \approx \frac{1}{2h}\left(\psi_{i+1,j} - \psi_{i-1,j}\right), \qquad \frac{\partial\psi}{\partial y} \approx \frac{1}{2h}\left(\psi_{i,j+1} - \psi_{i,j-1}\right),$$
$$\frac{\partial\omega}{\partial x} \approx \frac{1}{2h}\left(\omega_{i+1,j} - \omega_{i-1,j}\right), \qquad \frac{\partial\omega}{\partial y} \approx \frac{1}{2h}\left(\omega_{i,j+1} - \omega_{i,j-1}\right).$$

We then write for (17.2),

$$\omega_{i,j}^{n+1} = (1 - r_\omega)\omega_{i,j}^n + \frac{r_\omega}{4}\left(\omega_{i+1,j}^n + \omega_{i-1,j}^n + \omega_{i,j+1}^n + \omega_{i,j-1}^n + \frac{\mathrm{Re}}{4}f_{i,j}^n\right), \tag{17.6}$$

where

$$f_{ij}^n = \left(\psi_{i+1,j}^n - \psi_{i-1,j}^n\right)\left(\omega_{i,j+1}^n - \omega_{i,j-1}^n\right) - \left(\psi_{i,j+1}^n - \psi_{i,j-1}^n\right)\left(\omega_{i+1,j}^n - \omega_{i-1,j}^n\right). \tag{17.7}$$

Now, the right-hand-side of (17.6) contains a nonlinear term given by (17.7). This nonlinearity can result in the iterations becoming unstable.

The iterations can be stabilized as follows. First, the relaxation parameters, r_ψ and r_ω, should be less than or equal to unity, and unstable iterations can often be made stable by decreasing r_ω. One needs to numerically experiment to obtain the best trade-off between computationally stability and speed. Second, to determine the solution with Reynolds number Re, the iteration should be initialized using the steady solution for a slightly smaller Reynolds number. Initial conditions for the first solution with Re slightly larger than zero should be chosen so that this first iteration is stable.

The path of convergence can be tracked during the iterations. We define

$$\varepsilon_\psi^{n+1} = \max_{i,j} \left| \psi_{i,j}^{n+1} - \psi_{i,j}^{n} \right|,$$
$$\varepsilon_\omega^{n+1} = \max_{i,j} \left| \omega_{i,j}^{n+1} - \omega_{i,j}^{n} \right|.$$

The iterations are to be stopped when the values of ε_ψ^{n+1} and ε_ω^{n+1} are less than some pre-defined error tolerance, say 10^{-8}.

17.1.2 Boundary conditions

Boundary conditions on $\psi_{i,j}$ and $\omega_{i,j}$ must be prescribed at $i = 0$ (inflow), $i = N_x$ (outflow), $j = 0$ (midline), and $j = N_y$ (top of computational domain). Also, boundary conditions must be prescribed on the surface of the obstacle; that is, on the front surface: $i = m$, $0 \le j \le J$; the back surface: $i = m + I$, $0 \le j \le J$; and the top surface: $m \le i \le m + I$, $j = J$. Inside of the obstacle, $m < i < m + I$, $0 < j < J$, no solution is sought.

For the inflow and top-of-domain boundary conditions, we may assume that the flow field satisfies dimensionless free-stream conditions; that is,

$$u = 1, \quad v = 0.$$

The vorticity may be taken to be zero, and the stream function satisfies

$$\frac{\partial \psi}{\partial y} = 1, \quad \frac{\partial \psi}{\partial x} = 0.$$

Integrating the first of these equations, we obtain

$$\psi = y + f(x);$$

and from the second equation we obtain $f(x) = c$, where c is a constant. Without loss of generality, we may choose $c = 0$. Therefore, for the inflow and top-of-domain boundary conditions, we have

$$\psi = y, \quad \omega = 0. \tag{17.8}$$

At the top of the domain, notice that $y = N_y h$ is a constant.

For the outflow boundary conditions, we have two possible choices. We could assume free-stream conditions if we place the outflow boundary sufficiently far away from the obstacle. However, one would expect that the disturbance to the flow field downstream of the obstacle might be substantially greater than that upstream of the obstacle. Perhaps better outflow boundary conditions may be zero normal derivatives of the flow field; that is

$$\frac{\partial \psi}{\partial x} = 0, \quad \frac{\partial \omega}{\partial x} = 0.$$

For the midline boundary conditions, we will assume a symmetric flow field so that the flow pattern will look the same when rotated about the x-axis. The symmetry conditions are therefore given by

$$u(x,-y) = u(x,y), \quad v(x,-y) = -v(x,y).$$

The vorticity exhibits the symmetry given by

$$\begin{aligned}\omega(x,-y) &= \frac{\partial v(x,-y)}{\partial x} - \frac{\partial u(x,-y)}{\partial(-y)} \\ &= -\frac{\partial v(x,y)}{\partial x} + \frac{\partial u(x,y)}{\partial(y)} \\ &= -\omega(x,y).\end{aligned}$$

On the midline ($y = 0$), then, $\omega(x,0) = 0$. Also, $v(x,0) = 0$. With $v = -\partial\psi/\partial x$, we must have $\psi(x,0)$ independent of x, or $\psi(x,0)$ equal to a constant. Matching the midline boundary condition to our chosen inflow condition determines $\psi(x,0) = 0$.

Our boundary conditions are discretized using (17.4). The outflow condition of zero normal derivative could be approximated either to first- or second-order. Since it is an approximate boundary condition, first-order is probably sufficient and we use $\psi_{N_x,j} = \psi_{N_x-1,j}$ and $\omega_{N_x,j} = \omega_{N_x-1,j}$.

Putting all these results together, the boundary conditions on the borders of the computational domain are given by

$\psi_{i,0} = 0,$	$\omega_{i,0} = 0,$	midline;
$\psi_{0,j} = jh,$	$\omega_{0,j} = 0,$	inflow;
$\psi_{N_x,j} = \psi_{N_x-1,j},$	$\omega_{N_x,j} = \omega_{N_x-1,j},$	outflow;
$\psi_{i,N_y} = N_y h,$	$\omega_{i,N_y} = 0,$	top-of-domain.

Boundary conditions on the obstacle can be derived from the no-penetration and no-slip conditions. From the no-penetration condition, $u = 0$ on the sides and $v = 0$ on the top. Therefore, on the sides, $\partial\psi/\partial y = 0$, and since the side boundaries are parallel to the y-axis, ψ must be constant. On the top, $\partial\psi/\partial x = 0$, and since the top is parallel to the x-axis, ψ must be constant. Matching the constant to the value of ψ on the midline, we obtain $\psi = 0$ along the boundary of the obstacle.

From the no-slip condition, $v = 0$ on the sides and $u = 0$ on the top. Therefore, $\partial\psi/\partial x = 0$ on the sides and $\partial\psi/\partial y = 0$ on the top. To interpret the no-slip boundary conditions in terms of boundary conditions on the vorticity, we make use of (17.1); that is,

$$\omega = -\left(\frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial y^2}\right). \tag{17.9}$$

First consider the sides of the obstacle. Since ψ is independent of y we have $\partial^2\psi/\partial y^2 = 0$, and (17.9) becomes

$$\omega = -\frac{\partial^2\psi}{\partial x^2}. \tag{17.10}$$

We now Taylor series expand $\psi(x_m - h, y_j)$ about (x_m, y_j), corresponding to the front face of the rectangular obstacle. We have to order h^2:

$$\psi_{m-1,j} = \psi_{m,j} - h\left.\frac{\partial\psi}{\partial x}\right|_{m,j} + \frac{1}{2}h^2\left.\frac{\partial^2\psi}{\partial x^2}\right|_{m,j}.$$

The first term in the Taylor series expansion is zero because of the no-penetration condition, the second term is zero because of the no-slip condition, and the third term can be rewritten using (17.10). Solving for the vorticity, we have

$$\omega_{m,j} = -\frac{2}{h^2}\psi_{m-1,j}.$$

Similar considerations applied to the back face of the rectangular obstacle yields

$$\omega_{m+I,j} = -\frac{2}{h^2}\psi_{m+I+1,j}.$$

On the top of the obstacle, $y = Jh$ is fixed, and since ψ is independent of x, we have $\partial^2\psi/\partial x^2 = 0$. Therefore,

$$\omega = -\frac{\partial^2\psi}{\partial y^2}.$$

We now Taylor series expand $\psi(x_i, y_{J+1})$ about (x_i, y_J). To order h^2,

$$\psi_{i,J+1} = \psi_{i,J} + h\left.\frac{\partial\psi}{\partial y}\right|_{i,J} + \frac{1}{2}h^2\left.\frac{\partial^2\psi}{\partial y^2}\right|_{i,J}.$$

Again, the first and second terms in the Taylor series expansion are zero, and making use of (17.10), we obtain

$$\omega_{i,J} = -\frac{2}{h^2}\psi_{i,J+1}.$$

We summarize the boundary conditions on the obstacle:

$$\begin{aligned}
&\text{front face:} \quad \psi_{m,j} = 0, && \omega_{m,j} = -\frac{2}{h^2}\psi_{m-1,j}, && 0 \le j \le J;\\
&\text{back face:} \quad \psi_{m+I,j} = 0, && \omega_{m+I,j} = -\frac{2}{h^2}\psi_{m+I+1,j}, && 0 \le j \le J;\\
&\text{top face:} \quad \psi_{i,J} = 0, && \omega_{i,J} = -\frac{2}{h^2}\psi_{i,J+1}, && m \le i \le m+I.
\end{aligned}$$

17.2 Flow past a circle

We now consider flow past a circular obstacle of radius R, with free-stream velocity $\mathbf{u} = U\hat{\mathbf{x}}$. Here, we nondimensionalize the governing equations using U and R. We will define the Reynolds number—the only dimensionless parameter of this problem—by

$$\mathrm{Re} = \frac{2UR}{\nu}. \tag{17.11}$$

The extra factor of 2 bases the definition of the Reynolds number on the diameter of the circle rather than the radius, which allows a better comparison to computations of flow past a square ($a = 1$), where the Reynolds number was based on the side length.

The dimensionless governing equations in vector form, can be written as

$$\nabla^2\psi = -\omega \tag{17.12a}$$

$$\nabla^2\omega = \frac{\mathrm{Re}}{2}\mathbf{u}\cdot\boldsymbol{\nabla}\omega, \tag{17.12b}$$

where the extra factor of one-half arises from nondimensionalizing the equation using the radius of the obstacle R, but defining the Reynolds number in terms of the diameter $2R$.

17.2.1 Log-polar coordinates

Although the free-stream velocity is best expressed in Cartesian coordinates, the boundaries of the circular obstacle are more simply expressed in polar coordinates, with origin at the center of the circle. Polar coordinates are defined in the usual way by

$$x = r\cos\theta, \quad y = r\sin\theta,$$

with the cartesian unit vectors given in terms of the polar unit vectors by

$$\hat{\mathbf{x}} = \cos\theta\hat{\mathbf{r}} - \sin\theta\hat{\boldsymbol{\theta}}, \quad \hat{\boldsymbol{y}} = \sin\theta\hat{\mathbf{r}} + \cos\theta\hat{\boldsymbol{\theta}}.$$

The polar unit vectors are functions of position, and their derivatives are given by

$$\frac{\partial\hat{\mathbf{r}}}{\partial r} = 0, \quad \frac{\partial\hat{\mathbf{r}}}{\partial\theta} = \hat{\boldsymbol{\theta}}, \quad \frac{\partial\hat{\boldsymbol{\theta}}}{\partial r} = 0, \quad \frac{\partial\hat{\boldsymbol{\theta}}}{\partial\theta} = -\hat{\mathbf{r}}.$$

The del differential operator in polar coordinates is given by

$$\boldsymbol{\nabla} = \hat{\mathbf{r}}\frac{\partial}{\partial r} + \hat{\boldsymbol{\theta}}\frac{1}{r}\frac{\partial}{\partial\theta},$$

and the two-dimensional Laplacian is given by

$$\nabla^2 = \frac{1}{r^2}\left(\left(r\frac{\partial}{\partial r}\right)\left(r\frac{\partial}{\partial r}\right) + \frac{\partial^2}{\partial\theta^2}\right). \tag{17.13}$$

The velocity field is written in polar coordinates as

$$\mathbf{u} = u_r\hat{\mathbf{r}} + u_\theta\hat{\boldsymbol{\theta}}.$$

The free-stream velocity in polar coordinates is found to be

$$\begin{aligned}\mathbf{u} &= U\hat{\mathbf{x}} \\ &= U\left(\cos\theta\hat{\mathbf{r}} - \sin\theta\hat{\boldsymbol{\theta}}\right),\end{aligned} \tag{17.14}$$

from which can be read off the components in polar coordinates. The continuity equation $\boldsymbol{\nabla}\cdot\mathbf{u} = 0$ in polar coordinates is given by

$$\frac{1}{r}\frac{\partial}{\partial r}\left(ru_r\right) + \frac{1}{r}\frac{\partial u_\theta}{\partial\theta} = 0,$$

so that the stream function can be defined by

$$ru_r = \frac{\partial\psi}{\partial\theta}, \quad u_\theta = -\frac{\partial\psi}{\partial r}. \tag{17.15}$$

The vorticity, here in cylindrical coordinates, is given by

$$\begin{aligned}\boldsymbol{\omega} &= \boldsymbol{\nabla}\times\mathbf{u} \\ &= \hat{\mathbf{z}}\left(\frac{1}{r}\frac{\partial}{\partial r}\left(ru_\theta\right) - \frac{1}{r}\frac{\partial u_r}{\partial\theta}\right),\end{aligned}$$

so that the z-component of the vorticity for a two-dimensional flow is given by

$$\omega = \frac{1}{r}\frac{\partial}{\partial r}\left(ru_\theta\right) - \frac{1}{r}\frac{\partial u_r}{\partial\theta}. \tag{17.16}$$

Furthermore,

$$\begin{aligned}\mathbf{u}\cdot\boldsymbol{\nabla} &= \left(u_r\hat{\mathbf{r}}+u_\theta\hat{\boldsymbol{\theta}}\right)\cdot\left(\hat{\mathbf{r}}\frac{\partial}{\partial r}+\hat{\boldsymbol{\theta}}\frac{1}{r}\frac{\partial}{\partial\theta}\right)\\ &= u_r\frac{\partial}{\partial r}+\frac{u_\theta}{r}\frac{\partial}{\partial\theta}\\ &= \frac{1}{r}\frac{\partial\psi}{\partial\theta}\frac{\partial}{\partial r}-\frac{1}{r}\frac{\partial\psi}{\partial r}\frac{\partial}{\partial\theta}.\end{aligned} \tag{17.17}$$

The governing equations given by (17.12), then, with the Laplacian given by (17.13), and the convection term given by (17.17), are

$$\nabla^2\psi = -\omega, \tag{17.18}$$

$$\nabla^2\omega = \frac{\mathrm{Re}}{2}\left(\frac{1}{r}\frac{\partial\psi}{\partial\theta}\frac{\partial\omega}{\partial r}-\frac{1}{r}\frac{\partial\psi}{\partial r}\frac{\partial\omega}{\partial\theta}\right). \tag{17.19}$$

The recurring factor $r\partial/\partial r$ in the polar coordinate Laplacian, (17.13), is awkward to discretize and we look for a change of variables $r = r(\xi)$, where

$$r\frac{\partial}{\partial r} = \frac{\partial}{\partial\xi}.$$

Now,

$$\frac{\partial}{\partial\xi} = \frac{dr}{d\xi}\frac{\partial}{\partial r},$$

so that we require

$$\frac{dr}{d\xi} = r. \tag{17.20}$$

This simple differential equation can be solved if we take as our boundary condition $\xi = 0$ when $r = 1$, corresponding to points lying on the boundary of the circular obstacle. The solution of (17.20) is therefore given by

$$r = e^{\xi}.$$

The Laplacian in the so-called log polar coordinates then becomes

$$\begin{aligned}\nabla^2 &= \frac{1}{r^2}\left(\left(r\frac{\partial}{\partial r}\right)\left(r\frac{\partial}{\partial r}\right)+\frac{\partial^2}{\partial\theta^2}\right)\\ &= e^{-2\xi}\left(\frac{\partial^2}{\partial\xi^2}+\frac{\partial^2}{\partial\theta^2}\right).\end{aligned}$$

Also, transforming the right-hand-side of (17.19), we have

$$\begin{aligned}\frac{1}{r}\frac{\partial\psi}{\partial\theta}\frac{\partial\omega}{\partial r}-\frac{1}{r}\frac{\partial\psi}{\partial r}\frac{\partial\omega}{\partial\theta} &= \frac{1}{r^2}\left(r\frac{\partial\omega}{\partial r}\frac{\partial\psi}{\partial\theta}-r\frac{\partial\psi}{\partial r}\frac{\partial\omega}{\partial\theta}\right)\\ &= e^{-2\xi}\left(\frac{\partial\psi}{\partial\theta}\frac{\partial\omega}{\partial\xi}-\frac{\partial\psi}{\partial\xi}\frac{\partial\omega}{\partial\theta}\right).\end{aligned}$$

The governing equations for $\psi = \psi(\xi,\theta)$ and $\omega = \omega(\xi,\theta)$ in log-polar coordinates can therefore be written as

$$-\left(\frac{\partial^2}{\partial\xi^2}+\frac{\partial^2}{\partial\theta^2}\right)\psi = e^{2\xi}\omega, \tag{17.21a}$$

$$-\left(\frac{\partial^2}{\partial\xi^2}+\frac{\partial^2}{\partial\theta^2}\right)\omega = \frac{\mathrm{Re}}{2}\left(\frac{\partial\psi}{\partial\xi}\frac{\partial\omega}{\partial\theta}-\frac{\partial\psi}{\partial\theta}\frac{\partial\omega}{\partial\xi}\right). \tag{17.21b}$$

17.2.2 Finite difference approximation

A finite difference approximation to the governing equations proceeds on a grid in (ξ, θ) space. The grid is defined for $0 \leq \xi \leq \xi_m$ and $0 \leq \theta \leq \pi$, so that the computational domain forms a rectangle without holes. The sides of the rectangle correspond to the boundary of the circular obstacle ($\xi = 0$), the free stream ($\xi = \xi_m$), the midline behind the obstacle ($\theta = 0$), and the midline in front of the obstacle ($\theta = \pi$) .

We discretize the equations using square grid cells, and write

$$\xi_i = ih, \qquad i = 0, 1, \ldots, n; \tag{17.22a}$$

$$\theta_j = jh, \qquad j = 0, 1, \ldots, m, \tag{17.22b}$$

where n and m are the number of grid cells spanning the ξ- and θ-directions, and h is the side length of a grid cell. Because $0 \leq \theta \leq \pi$, the grid spacing must satisfy

$$h = \frac{\pi}{m},$$

and the maximum value of ξ is given by

$$\xi_{\max} = \frac{n\pi}{m}.$$

The radius of the computational domain is therefore given by $e^{\xi_{\max}}$, which is to be compared to the obstacle radius of unity. The choice $n = m$ would yield $e^{\xi_{\max}} \approx 23$, and the choice $n = 2m$ would yield $e^{\xi_{\max}} \approx 535$. To perform an accurate computation, it is likely that both the value of m (and n) and the value of $\xi_{\max}$ will need to increase with Reynolds number.

Again we make use of the SOR method, using the notation for the Jacobi method, although faster convergence is likely to be achieved using red-black Gauss-Seidel. The finite difference approximation to (17.21) thus becomes

$$\psi_{i,j}^{n+1} = (1 - r_\psi)\psi_{i,j}^n + \frac{r_\psi}{4}\left(\psi_{i+1,j}^n + \psi_{i-1,j}^n + \psi_{i,j+1}^n + \psi_{i,j-1}^n + h^2 e^{2\xi_i}\omega_{i,j}^n\right), \tag{17.23}$$

and

$$\omega_{i,j}^{n+1} = (1 - r_\omega)\omega_{i,j}^n + \frac{r_\omega}{4}\left(\omega_{i+1,j}^n + \omega_{i-1,j}^n + \omega_{i,j+1}^n + \omega_{i,j-1}^n + \frac{\mathrm{Re}}{8} f_{i,j}^n\right), \tag{17.24}$$

where

$$f_{ij}^n = \left(\psi_{i+1,j}^n - \psi_{i-1,j}^n\right)\left(\omega_{i,j+1}^n - \omega_{i,j-1}^n\right) - \left(\psi_{i,j+1}^n - \psi_{i,j-1}^n\right)\left(\omega_{i+1,j}^n - \omega_{i-1,j}^n\right). \tag{17.25}$$

17.2.3 Boundary conditions

Boundary conditions must be prescribed on the sides of the rectangular computational domain. The boundary conditions on the two sides corresponding to the midline of the physical domain, $\theta = 0$ and $\theta = \pi$, satisfy $\psi = 0$ and $\omega = 0$. The boundary condition on the side corresponding to the circular obstacle, $\xi = 0$, is again determined from the no-penentration and no-slip conditions, and are given by $\psi = 0$ and $\partial\psi/\partial\xi = 0$. And the free-stream boundary condition may be applied at $\xi = \xi_{\max}$.

We first consider the free-stream boundary condition. The dimensionless free-stream velocity field in polar coordinates can be found from (17.14),

$$\mathbf{u} = \cos\theta\hat{\mathbf{r}} - \sin\theta\hat{\boldsymbol{\theta}}.$$

The stream function, therefore, satisfies the free-stream conditions

$$\frac{\partial \psi}{\partial \theta} = r\cos\theta, \quad \frac{\partial \psi}{\partial r} = \sin\theta,$$

and by inspection, the solution that also satisfies $\psi = 0$ when $\theta = 0, \pi$ is given by

$$\psi(r,\theta) = r\sin\theta.$$

In log-polar coordinates, we therefore have the free-stream boundary condition

$$\psi(\xi_{\max},\theta) = e^{\xi_{\max}}\sin\theta.$$

One has two options for the vorticity in the free stream. One could take the vorticity in the free stream to be zero, so that

$$\omega(\xi_{\max},\theta) = 0.$$

A second, more gentle option is to take the derivative of the vorticity to be zero, so that

$$\frac{\partial \omega}{\partial \xi}(\xi_{\max},\theta) = 0.$$

This second option seems to have somewhat better stability properties for the flow field far downstream of the obstacle. Ideally, the computed values of interest should be independent of which of these boundary conditions is chosen, and finding flow-field solutions using both of these boundary conditions provides a good measure of accuracy.

The remaining missing boundary condition is for the vorticity on the obstacle. Again, we need to convert the two boundary conditions on the stream function, $\psi = 0$ and $\partial\psi/\partial\xi = 0$ to a boundary condition on ψ and ω. From (17.21), we have

$$\omega = -e^{-2\xi}\left(\frac{\partial^2\psi}{\partial\xi^2} + \frac{\partial^2\psi}{\partial\theta^2}\right),$$

and since on the circle $\psi = 0$, independent of θ, and $\xi = 0$, we have

$$\omega = -\frac{\partial^2\psi}{\partial\xi^2}. \tag{17.26}$$

A Taylor series expansion one grid point away from the circular obstacle yields

$$\psi_{1,j} = \psi_{0,j} + h\left.\frac{\partial\psi}{\partial\xi}\right|_{(0,j)} + \frac{1}{2}h^2\left.\frac{\partial^2\psi}{\partial\xi^2}\right|_{(0,j)}.$$

Now, both $\psi = 0$ and $\partial\psi/\partial\xi = 0$ at the grid point $(0,j)$. Using the equation for the vorticity on the circle, (17.26), results in

$$\omega_{0,j} = -\frac{2}{h^2}\psi_{1,j}.$$

The boundary conditions are summarized below:

$$\begin{aligned}
\xi = 0,\ 0 \le \theta \le \pi: &\quad \psi_{0,j} = 0, \quad \omega_{0,j} = -2\psi_{1,j}/h^2;\\
\xi = \xi_{\max},\ 0 \le \theta \le \pi: &\quad \psi_{n,j} = e^{\xi_{\max}}\sin jh, \quad \omega_{n,j} = 0 \text{ or } \omega_{n,j} = \omega_{n-1,j};\\
0 \le \xi \le \xi_{\max},\ \theta = 0: &\quad \psi_{i,0} = 0, \quad \omega_{i,0} = 0;\\
0 \le \xi \le \xi_{\max},\ \theta = \pi: &\quad \psi_{i,m} = 0, \quad \omega_{i,m} = 0.
\end{aligned} \tag{17.27}$$

17.2.4 Solution using Newton's method

We consider here a much more efficient method to find the steady fluid flow solution. Unfortunately, this method is also more difficult to program. Recall from §7.2 that Newton's method can be used to solve a system of nonlinear equations. Newton's method as a root-finding routine has the strong advantage of being very fast when it converges, but the disadvantage of not always converging. Here, the problem of convergence can be overcome by solving for larger Re using as an initial guess the solution for slightly smaller Re, with *slightly* to be defined by trial and error.

Recall that Newton's method can solve a system of nonlinear equations of the form

$$F(\psi,\omega)=0, \quad G(\psi,\omega)=0. \tag{17.28}$$

Newton's method is implemented by writing

$$\psi^{(k+1)}=\psi^{(k)}+\Delta\psi, \quad \omega^{(k+1)}=\omega^{(k)}+\Delta\omega, \tag{17.29}$$

and the iteration scheme is derived by linearizing (17.28) in $\Delta\psi$ and $\Delta\omega$ to obtain

$$\mathrm{J}\begin{pmatrix}\Delta\psi\\ \Delta\omega\end{pmatrix}=-\begin{pmatrix}F\\ G\end{pmatrix}, \tag{17.30}$$

where J is the Jacobian matrix of the functions F and G. All functions are evaluated at $\psi^{(k)}$ and $\omega^{(k)}$.

Here, we should view ψ and ω as a large number of unknowns and F and G a correspondingly large number of equations, where the total number of equations must necessarily equal the total number of unknowns.

If we rewrite our governing equations into the form given by (17.28), we have

$$-\left(\frac{\partial^2}{\partial\xi^2}+\frac{\partial^2}{\partial\theta^2}\right)\psi-e^{2\xi}\omega=0, \tag{17.31a}$$

$$-\left(\frac{\partial^2}{\partial\xi^2}+\frac{\partial^2}{\partial\theta^2}\right)\omega-\frac{\mathrm{Re}}{2}\left(\frac{\partial\psi}{\partial\xi}\frac{\partial\omega}{\partial\theta}-\frac{\partial\psi}{\partial\theta}\frac{\partial\omega}{\partial\xi}\right)=0. \tag{17.31b}$$

With n and m grid cells in the ξ- and θ-directions, the partial differential equations of (17.31) represent $2(n-1)(m-1)$ coupled nonlinear equations for $\psi_{i,j}$ and $\omega_{i,j}$ on the internal grid points. We will also include the boundary values in the solution vector that will add an additional two unknowns and two equations for each boundary point, bringing the total number of equations (and unknowns) to $2(n+1)(m+1)$.

The form of the Jacobian matrix may be determined by linearizing (17.31) in $\Delta\psi$ and $\Delta\omega$. Using (17.29), we have

$$-\left(\frac{\partial^2}{\partial\xi^2}+\frac{\partial^2}{\partial\theta^2}\right)(\psi^{(k)}+\Delta\psi)-e^{2\xi}(\omega^{(k)}+\Delta\omega)=0,$$

and

$$-\left(\frac{\partial^2}{\partial\xi^2}+\frac{\partial^2}{\partial\theta^2}\right)(\omega^{(k)}+\Delta\omega)$$
$$-\frac{\mathrm{Re}}{2}\left(\frac{\partial(\psi^{(k)}+\Delta\psi)}{\partial\xi}\frac{\partial(\omega^{(k)}+\Delta\omega)}{\partial\theta}-\frac{\partial(\psi^{(k)}+\Delta\psi)}{\partial\theta}\frac{\partial(\omega^{(k)}+\Delta\omega)}{\partial\xi}\right)=0.$$

Liinearization in $\Delta\psi$ and $\Delta\omega$ then results in

$$-\left(\frac{\partial^2}{\partial\xi^2}+\frac{\partial^2}{\partial\theta^2}\right)\Delta\psi-e^{2\xi}\Delta\omega=-\left[-\left(\frac{\partial^2}{\partial\xi^2}+\frac{\partial^2}{\partial\theta^2}\right)\psi^{(k)}-e^{2\xi}\omega^{(k)}\right], \tag{17.32}$$

and

$$-\left(\frac{\partial^2}{\partial\xi^2}+\frac{\partial^2}{\partial\theta^2}\right)\Delta\omega-\frac{\mathrm{Re}}{2}\left(\frac{\partial\omega^{(k)}}{\partial\theta}\frac{\partial\Delta\psi}{\partial\xi}-\frac{\partial\omega^{(k)}}{\partial\xi}\frac{\partial\Delta\psi}{\partial\theta}+\frac{\partial\psi^{(k)}}{\partial\xi}\frac{\partial\Delta\omega}{\partial\theta}-\frac{\partial\psi^{(k)}}{\partial\theta}\frac{\partial\Delta\omega}{\partial\xi}\right)$$
$$=-\left[-\left(\frac{\partial^2}{\partial\xi^2}+\frac{\partial^2}{\partial\theta^2}\right)\omega^{(k)}-\frac{\mathrm{Re}}{2}\left(\frac{\partial\psi^{(k)}}{\partial\xi}\frac{\partial\omega^{(k)}}{\partial\theta}-\frac{\partial\psi^{(k)}}{\partial\theta}\frac{\partial\omega^{(k)}}{\partial\xi}\right)\right], \quad (17.33)$$

where the first equation was already linear in the Δ variables, but the second equation was originally quadratic, and the quadratic terms have now been dropped. The equations given by (17.32) and (17.33) can be observed to be in the form of the Newton's iteration equations given by (17.30).

Numerically, both $\Delta\psi$ and $\Delta\omega$ will be vectors formed by a natural ordering of the grid points, as detailed in §6.2. These two vectors will then be stacked into a single vector as shown in (17.30). To write the Jacobian matrix, we employ the shorthand notation $\partial_\xi^2=\partial^2/\partial\xi^2$, $\psi_\xi=\partial\psi/\partial\xi$, and so on. The Jacobian matrix can then be written symbolically as

$$\mathrm{J} \quad = \quad \begin{pmatrix} -\left(\partial_\xi^2+\partial_\theta^2\right) & -e^{2\xi}\mathrm{I} \\ 0 & -\left(\partial_\xi^2+\partial_\theta^2\right) \end{pmatrix} - \frac{\mathrm{Re}}{2}\begin{pmatrix} 0 & 0 \\ \omega_\theta\partial_\xi-\omega_\xi\partial_\theta & \psi_\xi\partial_\theta-\psi_\theta\partial_\xi \end{pmatrix}, \quad (17.34)$$

where I is the identity matrix and the derivatives of ψ and ω in the second matrix are all evaluated at the kth iteration.

The Jacobian matrix as written is valid for the grid points interior to the boundary, where each row of J corresponds to an equation for either ψ or ω at a specific interior grid point. The Laplacian-like operator is represented by a Laplacian matrix, and the derivative operators are represented by derivative matrices. The terms $e^{2\xi}$, $\partial\omega/\partial\theta$, and the other derivative terms are to be evaluated at the grid point corresponding to the row in which they are found.

To incorporate boundary conditions, we extend the vectors $\Delta\psi$ and $\Delta\omega$ to also include the points on the boundary as they occur in the natural ordering of the grid points. To the Jacobian matrix and the right-hand-side of (17.30) are then added the appropriate equations for the boundary conditions in the rows corresponding to the boundary points. By explicitly including the boundary points in the solution vector, the second-order accurate Laplacian matrix and derivative matrices present in J can handle the grid points lying directly next to the boundaries without special treatment.

The relevant boundary conditions to be implemented are the boundary conditions on $\Delta\psi$ and $\Delta\omega$. The boundary conditions on the fields ψ and ω themselves have already been given by (17.27). The grid points with fixed boundary conditions on ψ and ω that do not change with iterations will have a one on the diagonal in the Jacobian matrix corresponding to that grid point, and a zero on the right-hand-side. In other words, ψ and ω will not change on iteration of Newton's method, and their initial values need to be chosen to satisfy the appropriate boundary conditions.

The two boundary conditions which change on iteration, namely

$$\omega_{0,j}=-2\psi_{1,j}/h^2, \quad \omega_{n,j}=\omega_{n-1,j},$$

must be implemented in the Newton's method iteration as

$$\Delta\omega_{0,j}=-2\Delta\psi_{1,j}/h^2, \quad \Delta\omega_{n,j}=\Delta\omega_{n-1,j},$$

and these equations occur in the rows corresponding to the grid points $(0,j)$ and (n,j), with $j=0$ to m. Again, the initial conditions for the iteration must satisfy the correct boundary conditions.

The MATLAB implementation of (17.30) using (17.34), requires both the construction of the $(n+1)(m+1) \times (n+1)(m+1)$ matrix that includes both the Jacobian matrix and the boundary conditions, and the construction of the corresponding right-hand-side of the equation. For the Laplacian matrix, one can make use of the function `sp_laplace_new.m`; and one also needs to construct the derivative matrices $\partial_{\tilde{\xi}}$ and ∂_θ. Both of these matrices are banded, with a band of positive ones above the main diagonal and a band of negative ones below the main diagonal. For $\partial_{\tilde{\xi}}$, the bands are directly above and below the diagonal. For ∂_θ, the bands are a distance $n+1$ away from the diagonal, corresponding to $n+1$ grid points in each row. Both the Laplacian and derivative matrices are to be constructed for $(n+1) \times (m+1)$ grid points and placed into a 2×2 matrix using the MATLAB function `kron.m`, which generates a block matrix by implementing the so-called Kronecker product. Rows corresponding to the boundary points are then to be replaced by the equations for the boundary conditions.

The MATLAB code needs to be written efficiently, using sparse matrices. A profiling of this code should show that most of the computational time is spent solving (17.30) (with boundary conditions added) for $\Delta\psi$ and $\Delta\omega$ using the MATLAB backslash operator. With 4GB RAM and a notebook computer bought circa 2013, and with the resolution 512×256 and using the $Re = 150$ result as the initial field, I can solve for $Re = 200$ in seven iterations to an accuracy of 10^{-12}. The total run time was about 48 sec with about 42 sec spent on the single line containing `J\b`.

17.3 Visualization of the flow fields

Obtaining correct contour plots of the stream function and vorticity can be a challenge and in this section I will provide some guidance. The basic MATLAB functions required are `meshgrid.m`, `pol2cart.m`, `contour.m`, and `clabel.m`. More fancy functions such as `contourf.m` and `imagesc` may also be used, though I will not discuss these here.

Suppose the values of the stream function are known on a grid in two dimensional Cartesian coordinates. A contour plot draws curves following specified (or default) constant values of the stream function in the x-y plane. Viewing the curves on which the stream function is constant gives a clear visualization of the fluid flow.

To make the best use of the function `contour.m`, one specifies the x-y grid on which the values of the stream function are known. The stream function variable `psi`, say, is given as an n-by-m matrix. We will examine a simple example to understand how to organize the data.

Let us assume that the stream function is known at all the values of x and y on the two-dimensional grid specified by `x=[0,1,2]` and `y=[0,1]`. To properly label the axis of the contour plot, we use the function `meshgrid`, and write `[X,Y]=meshgrid(x,y)`. The values assigned to `X` and `Y` are the following 2-by-3 matrices:

$$X = \begin{bmatrix} 0 & 1 & 2 \\ 0 & 1 & 2 \end{bmatrix}, \quad Y = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}.$$

The variable `psi` must have the same dimensions as the variables `X` and `Y`. Suppose `psi` is given by

$$\text{psi} = \begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix}.$$

Then the data must be organized so that psi $= a$ at $(x,y) = (0,0)$, psi $= b$ at $(x,y) = (1,0)$, psi $= d$ at $(x,y) = (0,1)$, etc. Notice that the values of `psi` across a row (`psi(i,j), j=1:3`) correspond to different x locations, and the values of `psi` down a column ((`psi(i,j), i=1:2`)) correspond to different y locations. Although this is visually intuitive since x corresponds to horizontal variation and y to vertical variation, it is algebraically counterintuitive: the first index of `psi` corresponds to the y variation and the second index corresponds to the x variation. If one

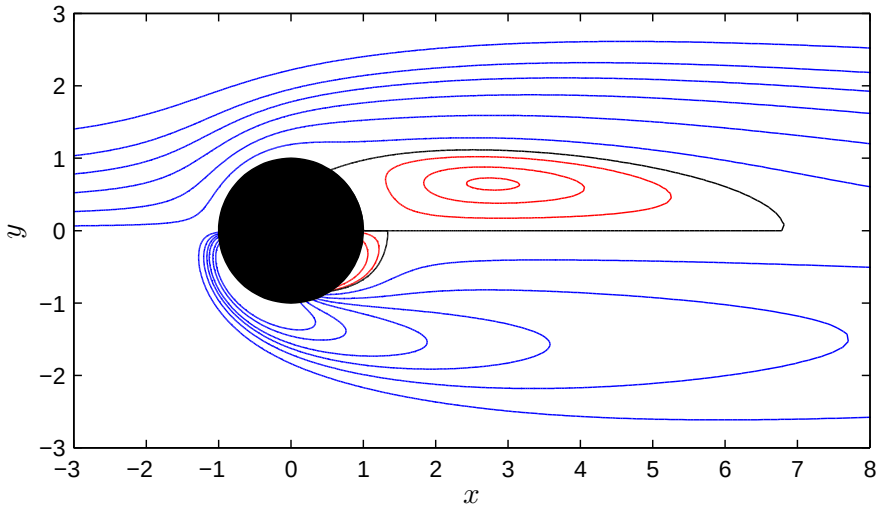


Figure 17.1: *Contour plots of the stream function ($y > 0$) and vorticity ($y < 0$) for $Re = 50$. Negative contours are in red, the zero contour in black, and positive contours in blue. The contour levels for the stream function are -0.05, -0.04, -0.02, 0, 0.05, 0.2, 0.4, 0.6, 0.8, 1.1 and those for the vorticity are -0.2, -0.05, 0, 0.25, 0.5, 0.75, 1, 1.5, 2.*

uses the notation psi(x, y) during computation, then to plot one needs to take the transpose of the matrix `psi`.

Now the computation of the flow field around a circle is done using log-polar coordinates (ξ, θ). To construct a contour plot, the solution fields need to be transformed to cartesian coordinates. The MATLAB function `pol2cart.m` provides a simple solution. One defines the variables `theta` and `xi` that defines the mesh in log-polar coordinates, and then first transforms to standard polar coordinates with `r=exp(xi)`. The polar coordinate grid is then constructed from `[THETA, XI]=meshgrid(theta,xi)`, and the cartesian grid is constructed from `[X, Y]=pol2cart(THETA,XI)`. The fields can then be plotted directly using the cartesian grid, even though this grid is not uniform. That is, a simple contour plot can be made with the command `contour(X,Y,psi)`. More sophisticated calls to `contour.m` specify the precise contour lines to be plotted, and their labelling using `clabel.m`.

A nice way to plot both the stream function and the vorticity fields on a single graph is to plot the stream function contours for $y > 0$ and the vorticity contours for $y < 0$, making use of the symmetry of the fields around the x-axis. In way of illustration, a plot of the stream function and vorticity contours for $Re = 50$ is shown in Fig. 17.1.