42 Problems in Scientific Computing

Contents

1. Arithmetic

Our first group of problems are not science problems in themselves. But they do deal with broad issues that lurk in the background when solving computational science problems. Some of them are also fun to think about, and help practise programming.

These notes will freely mix mathematical notation with Python fragments. The meaning will always be clear from context. For example

$$\begin{aligned} x &= x + y \\ y &= x - y \\ x &= x - y \end{aligned} \tag{1.1}$$

is clearly not mathematics, but it makes sense as Python. Can you work out what it does?

1.1 Finite precision

Computers work internally in binary, but most of the time the binary is hidden from us. We can, however, see something of what is going on internally by examining arithmetic operations in Python.

Since integers are stored in binary,

$$pow(2, n) \tag{1.2}$$

is a trivial operation. At some point, Python will go to long integers. When this happens depends on how many bits (32 or 64) are being used for integers.

Python had no Y2K problem, because time is expressed in seconds from 1. January 1970 (UTC). As we can see from the following

$$\begin{aligned} &\textbf{from}\ time\ \textbf{import}\ ctime \\ &ctime(946681200) \end{aligned} \tag{1.3}$$

the millenium had no special significance for the machine. But something analogous to the Y2K bug may occur later in the future, when a 32-bit integer is no longer enough to store the time. If 64-bit integers are standard by then, Python will have no problems.

Floating-point numbers are stored in the form $\pm m \times 2^{\pm p}$. The components m and p are known as the mantissa and exponents respectively. Floating-point underflow is a situation of the type

$$1 + eps == 1 \tag{1.4}$$

where eps is small but nonzero.

Work out the following by experimenting with arithmetic operations.

(i) When is the deadline for changing to 64-bit integers?

(ii) How many bits are used to store the mantissa and exponent?

Write a short note explaining your conclusions.

1.2 Square and other roots

One often needs to solve equations of the form $f(x) = 0$. This is conveniently provided for us by *scipy.optimize.brentq* but it is interesting to implement a root-finder ourselves.

One elegant method, going back to Newton and his contemporary Joseph Raphson, is to iterate

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{1.5}$$

Use the Newton-Raphson method to implement

```
def nroot(x, n = 2):                          (1.6)
    # returns n-th root of x > 0
```

Try to make your function deliver the most accurate answer possible in floating point. That is, have it iterate to underflow.

1.3 Continued fractions and curlicues

In Figure 1.1 we see a curlicue pattern produced by connecting the partial sums of

$$\sum_{n=1}^{L} \exp(i\pi n^2/\mu) \tag{1.7}$$

The curls are encoded in the continued-fraction expansion of the parameter μ. Continued fractions are often written as a list of integers

$$[a_0, a_1, a_2, \ldots] \equiv a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \ldots}} \tag{1.8}$$

Replacing L by L/μ and removing the first term in the continued-fraction expansion of μ gives approximately a scaled and reflected version of the same curlicue.

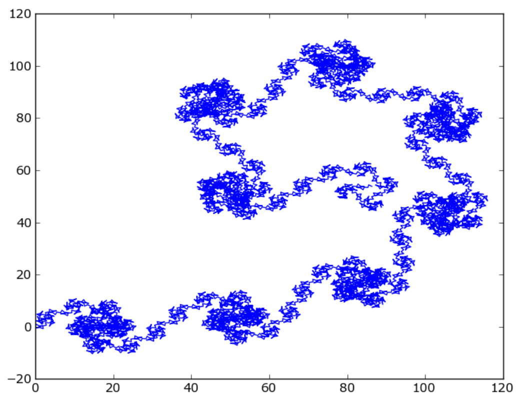

Figure 1.1: Curlicues from (1.7) with L going up to 10000 and $\mu = [2, 4, 8, 16, 32, \ldots]$ in the continued-fraction notation (1.8).

This example lends itself well to Python's functional-programming features. Note especially $\mathit{reduce}()$ which concisely does not only sums

$$\mathit{reduce}(\mathbf{lambda}\ s, n : s + n, \mathit{range}(101), 0) \tag{1.9}$$

and continued fractions

$$\begin{aligned} &p = [292, 1, 15, 7, 3] \\ &\mathit{reduce}(\mathbf{lambda}\ s, n : n + 1/s, p, 1.) \end{aligned} \tag{1.10}$$

but also partial sums.

$$\mathit{reduce}(\mathbf{lambda}\ s, n : s + [s[-1] + n], \mathit{range}(101), [0]) \tag{1.11}$$

Use $\mathit{pylab.plot}()$ to plot a curlicue as in Figure 1.1. Try to use $\mathit{reduce}()$ for the continued fractions as well as for the partial sums, avoiding loops.

1.4 Ada Lovelace's program

To be written

1.5 Machin's formula

Machin's formula is a very simple but surprisingly powerful formula for π, and readily yields hundreds or thousands of digits.

$$\frac{\pi}{4} = 4\arctan(1/5) - \arctan(1/239) \tag{1.12}$$

The Taylor expansion

$$\arctan x = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \ldots \tag{1.13}$$

means that for $x = \frac{1}{5}$ the series delivers better than one digit per term.

To derive Machin's formula we make use of the identity

$$\tan(a + b) = \frac{\tan a + \tan b}{1 - \tan a \tan b} \tag{1.14}$$

First define q by

$$q \equiv \tan(4\arctan(1/5) - \pi/4) \tag{1.15}$$

Using (1.14) we can evaluate $\tan(2\arctan\frac{1}{5}) = \frac{5}{12}$ and then $\tan(4\arctan\frac{1}{5}) = \frac{120}{119}$. We know $\arctan\frac{\pi}{4} = 1$. Substituting in (1.14) again gives $q = \frac{1}{239}$.

Implement Machin's formula and compute π to roundoff accuracy.

If you find that too easy, implement the sum as fractions, using integer arithmetic to keep track of numerators and denominators, and derive 1000 decimal digits. It helps to reduce all fractions to their lowest terms, by dividing the numerator and denominator in each case by their greatest common divisor. The compute the latter, the lazy way

```
**def** $gcd(a,b)$ **:**
    **if** $b == 0$ :
        **return** $a$                                        (1.16)
    $r = a\%b$
    **return** $gcd(b,r)$
```

is good enough for this problem. If the machine complains that you are overusing recursion, try this.

```
**import** $sys$                                              (1.17)
$sys.setrecursionlimit(10000)$
```

2. Sequences

2.6 Gamow's diamonds

In the years between the structure of DNA being discovered and the genetic code being correctly unravelled, there were several proposals for what the genetic code could be. They are mostly forgotten now, but one is still remembered, because though it was wrong, it was very ingenious.

George Gamow guessed (correctly!) that codons of three bases would map to 20 amino acids. Then he suggested that a sequence of bases would form a sort of cage for an amino acid. Different codons would give differently shaped cages, and each shape of cage would be just right for amino acid.

Consider a codon, say ACG. On the other strand there would be TGC. Now consider the inclined 'diamond' ACCG made from both strands. This is what Gamow suggested might be a cage for an amino acid. Now, the $4^3 = 64$ possible codons correspond to 64 possible diamonds. But some these are equivalent by symmetry. Tracing the ACCG diamond backwards, we have AGCC, and starting from the other side we have CCAG and CGAC.

Show that allowing for the symmetries leaves exactly 20 different Gamow diamonds.

2.7 Eight Queens

A classic problem in elementary combinatorics is to place eight queens on a chessboard such that no two are attacking each other. A more subtle problem is to find all the independent solution, meaning solutions not related by rotations and/or reflection.

The solutions can be enumerated in several ways. One convenient method is recursive backtracking. In this, we start with one queen per column, then move up the rows in odometer fashion. For example, if (say) queen 4 has moved to the end of rows 4, we backtrack to queen 3. On a 4×4 chessboard we easily find that the only possible solutions are $[2, 4, 1, 3]$ and $[3, 1, 4, 2]$ where each list element represents a column and its value the row.

The two above solutions are, however, not independent: they are related by reflection. Each solution also happens to be invariant under rotation by $90°$, so the two solutions are related by rotation as well. How to represent these operations?

On the complex plane, rotation and reflection are very easy. To take advantage of this, let us put the chessboard on the complex plane, centred at the origin. Assuming (arbitrarily) that that each square is two units on a side, the solution $[2, 4, 1, 3]$ can then be expressed as

$$[-3 - i,\ -1 + 3i,\ 1 - 3i,\ 3 + i\,] \tag{2.1}$$

while its complex conjugate represents $[3, 1, 4, 2]$.

In the complex representation, the order is arbitrary: we are free to sort. In fact, (2.1) is already sorted by increasing real part. To represent rotation and reflection, the operations we need to do are multiplication by $i, -1, -i$ and complex-conjugation, followed in each case by sorting. As is easy to verify, all these operations leave the complex list invariant.

Using the above strategy (or another algorithm, if you prefer) find all the independent solutions on an 8×8 chessboard.

2.8 Sequence alignment

The comparison of gene sequences (or protein sequences) in different species is a fundamental problem in bioinformatics. We will consider the simplest form of sequence comparison: pairwise dot plots of matching sub-sequences. Figure 2.1 shows an example comparing two strings.

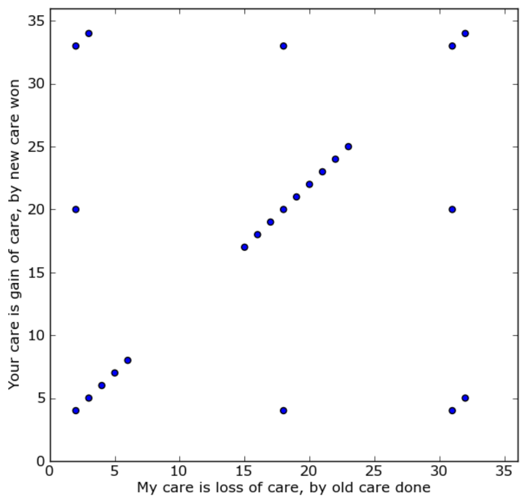

Figure 2.1: Alignment of two strings. A dot is plotted whenever a five-character region matches.

To apply the idea to genomic sequences, we need to get the data from one of several large databases. The following imports a protein sequence from `www.uniport.org` using the *Biopython* library.

```
from Bio import ExPASy, SeqIO
sid = raw_input("Sequence id?  ")
 try :
    handle = ExPASy.get_sprot_raw(sid)
    seq = SeqIO.read(handle, "swiss")                          (2.2)
    SeqIO.write(seq, sid + ".genbank", "genbank")
    print "Sequence length", len(seq)
 except Exception:
    print "Sequence not found"
```

This saves the data into a text file in the so-called Genbank format. One can use another *Biopython* function to extract the sequence, or simply read it directly from the file.

Biopython also has a nice function to draw phylogenetic trees. If you write a text file `tree.txt` like

```
(chimpanzee,(neanderthal,human))
```

in the so-called Newick format, the following code will draw the tree for you.

```
from Bio import Phylo
tree = Phylo.read("tree.txt","newick")
tree.rooted = True
Phylo.draw(tree)
```
(2.3)

Choose a gene found in many species and make some dot plots comparing the corresponding amino-acid sequence in different pairs of species. Try and infer the phylogenetic relationships of a few species from the dot plots, express it in Newick, and hence produce a phylogenetic tree.

3. Primes

3.9 The distribution of primes

A simple but still reasonably efficient way to generate primes is the ancient sieve of Eratosthenes: one removes multiples of 2, then multiples of 3, and so on, and this leaves the primes. A convenient way to implement the sieve is to keep two lists: one list $\{p_1, p_2, \ldots, p_k\}$ of all primes below some n, and a second list stepping through multiples of those primes. If n avoids getting on the second list, it is prime.

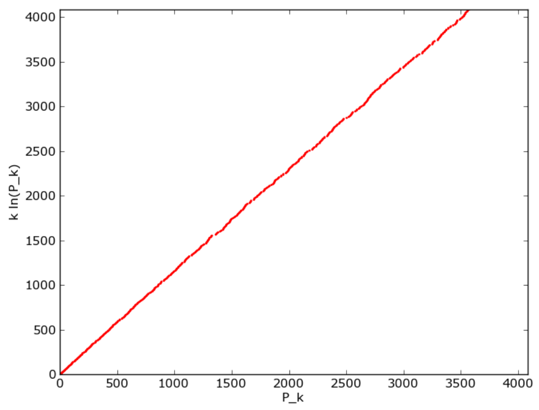

Figure 3.1: Distribution of primes.

Implement the sieve of Eratosthenes and plot p_k versus $k \ln(p_k)$, going further than Figure 3.1.

You can also try evaluating the product form of the Riemann zeta function for some values of z.

$$\zeta(z) = \prod_{p \text{ prime}} 1/(1 - p^{-z}) = \sum_{n=1}^{\infty} n^{-z} \tag{3.1}$$

For example $\zeta(2) = \pi^2/6$. The infinite product converges rather slowly.

3.10 Modular arithmetic and pseudo-primes

Some of the previous problems remind us that, powerful as computers are, computer arithmetic is always finite. It turns out that finite arithmetic has some interesting properties, and we will now explore a few of them. For this purpose, we will use Python's modular power function.

$$pow(a, n, N) \qquad \# \; a^n \pmod N \tag{3.2}$$

All arguments here must be integers, and n must be non-negative. The answer is always between 0 and N. In particular, for any integer m

$$pow(a + m * N, n, N) == pow(a, n, N) \tag{3.3}$$

First we need to introduce some finite-group theory. Finite groups have the following basic property.

Lagrange's theorem: for any subgroup S of a group G, $\mathcal{N}(S)$ divides $\mathcal{N}(G)$. (3.4)

We are using $\mathcal{N}$ to denote the order of a group.

To prove Lagrange's theorem, we consider cosets of S and note the following.

(i) Any $a \in G$ belongs to some coset.

(ii) For $s_1, s_2 \in S$ we have $as_1 = as_2 \Rightarrow s_2 = s_2$. In words, distinct $s \in S$ give distinct coset elements. Hence $\mathcal{N}(aS) = \mathcal{N}(S)$.

(iii) For $b \in G$ we have $as_1 = bs_2 \Rightarrow a(s_1 S) = b(s_2 S) \Rightarrow aS = bS$. That is to say, cosets are either equal or disjoint.

Combining these points proves the theorem.

Let us now definite an interesting finite group.

$$G_N \equiv \{a = 1, \ldots, N-1, \text{for all } a \text{ coprime with } N\} \tag{3.5}$$

To see that G_N is a group under multiplication, we reason as follows. Closure and identity obvious, the only non-trivial thing is to prove an inverse exists. To derive the inverse we use the so-called Bizout identity

$$ax + by = \gcd(a, b) \tag{3.6}$$

Given a, b we can find x, y to satisfy the Bizout identity using the extended Euclidean algorithm.

```
def $bizout(a,b)$ :
    if $b == 0$ :
        return $(1,0)$
    $q = a/b$
    $r = a - q * b$
    $x, y = bizout(b,r)$
    return $(y, x - q * y)$
```
(3.7)

As this function goes down the recursion, it replaces (a, b) by (b, r) where $r = a - qb$. This preserves common factors, and hence the recursion terminates at $\gcd(a, b)$. Coming out of the recursion, the function replaces (x, y) by $(y, x - qy)$. Since $bx + ry = ay + b(x - qy)$ coming out of the recursion preserves the Bizout identity.

Calling $bizout(a, N)[0]$ returns the inverse of a in G_N. Hence G_N is a group.

We now note that $\{1, a, a^2, \ldots, a^{k-1}\}$ is a subgroup of G_N for some k. Applying Lagrange's theorem, we have

# Theorem: for any $a \in G_N$
$pow(a, k, N) == 1$
# for some k that divides $\mathcal{N}(G_N)$. (3.8)

A corollary of (3.8) is Fermat's little theorem.

$pow(a, p-1, p) == 1$
# for any prime p and any a not a multiple of p. (3.9)

To prove this, we proceed as follows.

(i) For $a \in \{1, \dots, p-1\}$ we have $a \in G_p$.

(ii) For larger a, we recall the identity (3.3).

The converse of Fermat's little theorem is not true: there exist non-prime numbers q for which

$pow(a, q-1, q) == 1$ # for all a coprime with q (3.10)

Such q are known as Carmichael numbers, and they are rare (1729 is an example).

To find Carmichael numbers, one needs to have a list of primes, or some other test of primality. You may wish to use the sieve (Section 3.9).

Find the first few Carmichael numbers.

3.11 RSA encryption

Theorem (3.8) has more corollaries. Consider two primes p, q. Counting up elements, we have

$$\mathcal{N}(G_{pq}) = pq - 1 - (p-1) - (q-1) = (p-1)(q-1) \tag{3.11}$$

and hence the following.

# Corollary: For primes p, q and $a \in G_{pq}$
$pow(a, (p-1)(q-1), pq) == 1$ # Implicit $*$ signs (3.12)

A slight modification removes the restriction on a.

# Corollary: For primes p, q and any a
$pow(a, 1 + s(p-1)(q-1), pq) == a$ (3.13)

To prove this we must consider several cases.

(i) From identity (3.3) it is enough to consider $a \le pq$.

(ii) If $a = pq$ the identity is trivially true.

(iii) For $a \in G_{pq}$, validity follows from the previous corollary.

(iv) The remaining case is $a = mq$, $a \ne kp$. Then $a^{s(q-1)} \in G_p$. Applying Fermat's little theorem we have $a^{s(q-1)(p-1)} = 1 + np$. This implies $a^{1+s(p-1)(q-1)} = a + mqnp$ which is the desired result.

The well-known RSA public-key encryption method is based on the corollary (3.13). It is traditionally expressed in terms of two protagonists called Alice and Bob.

Bob selects two prime numbers p, q. These would typically have 200 or more decimal digits. Then he chooses two numbers $c, d \in G_{(p-1)(q-1)}$ that are mutually inverse. That is to say

$$cd = 1 + s(p-1)(q-1) \tag{3.14}$$

From (3.13) we can see that the following holds for arbitrary a.

$$\begin{aligned}&b = pow(a, c, N) \qquad \# \; N \equiv pq \\ &pow(b, d, N) == a\end{aligned} \tag{3.15}$$

Bob now announces N, c as his public key, and keeps d as his private key. (The individual primes p, q he also keeps secret.)

Now Alice can securely send Bob an integer a. Using the first line of (3.15) Alice encrypts her plaintext a to b. She then sends the cyphertext b over an open channel to Bob. Bob decrypts using the second line of (3.15). If a happens to be a multiple of p or q then $gcd(a, N)$ will reveal p and q, thus breaking the encryption — but for large numbers that is very unlikely.

The same basic scheme can also be used for digital signing. One signs (i.e., encrypts) using a private key, and the signature can be verified using the public key.

The encryption fails if someone (traditionally called Eve for eavesdropper) has enough computing power to factorize N. In fact, it is enough for Eve to solve a slightly simpler problem: given the cyphertext b, Eve needs to find r such that

$$pow(b, r, N) == 1 \tag{3.16}$$

From the first line of (3.15) it follows that

$$pow(a, r, N) == 1 \tag{3.17}$$

We have already assumed that a is coprime with p and q. In other words $a \in G_{pq}$. Hence $\{1, a, a^2, \ldots, a^{r-1}\}$ is a subgroup of G_{pq}. So by Lagrange's theorem, r must divide $(p-1)(q-1)$. Now $c \in G_{(p-1)(q-1)}$ is also coprime with $(p-1)(q-1)$. Hence c is coprime with r and $c \in G_r$.

Eve now chooses d' such that $cd' = 1 + mr$. This d' may or may not be equal to Bob's private key d, but it is good enough, because

$$pow(b, d', N) == pow(a, cd', N) == pow(a, 1 + mr, N) == a \tag{3.18}$$

where the last step follows from (3.17). Hence Eve recovers the plaintext a.

The computationally hard part of breaking RSA encryption is (3.16): finding the period of the plaintext. Quantum computers can in principle do that very efficiently (Shor's algorithm) but nobody knows yet if the technological difficulties can be overcome.

If the numbers are not too large, RSA encryption can be broken by brute force. Suppose Alice wishes to send a one-word text message to Bob. It is agreed that the word will be coded as an integer a, and extracted by the following function.

```
**def** $word(n)$ :
    $str =$ ""
    **while** $n > 0$ :
        $c = chr(n \% 32 + 96)$
        **if** $c <$ "a" **or** $c >$ "z" :                    (3.19)
            $c =$ " "
        $str \mathrel{+}= c$
        $n \mathrel{/}= 32$
    **return** $str$
```

For example $word(46204290)$ gives `"blabla"` and $word(969293283)$ gives `"cool  "`. This is just an open coding for convenience, not an encryption.

Meanwhile, Bob has announced his public key as $N = 1024384027, c = 910510237$. Alice encrypts her plaintext a using the public key and send the cyphertext $b = 100156265$ by an open channel. Bob uses his private key to recover a and hence the word.

Eve sees $b = 100156265$ on the open channel. She writes a program to solve (3.16) for r. With r in hand, she recovers a and finally the secret word.

What was the word Alice wanted to send?

4. Matrices

Linear algebra is an area where software libraries perform particularly well. This makes it worthwhile to think about how to reduce different kinds of physical problem to matrix systems. Many large and complicated problems can be rendered trivial in this way. We now explore some examples.

4.12 Circuit equations and the resistor-cube problem

Imagine twelve 1Ω resistors connected to make up a cube. What will be the resistance when a voltage is applied diagonally?

There are several clever ways of solving this problem, which you can find by searching online. But here we are not interested in clever ways that work *only* for this problem. We will do something much better: develop a general strategy for reducing electric circuits to a set of linear equations of the form

$$AX = B \tag{4.1}$$

which can be solved numerically, as follows.

$$\begin{array}{l} \textbf{from } \mathit{scipy.linalg} \textbf{ import } \mathit{solve} \\ X = \mathit{solve}(A, B) \end{array} \tag{4.2}$$

Before trying the resistor-cube, let us try a simpler problem, known as a Wheatstone bridge, shown in Figure 4.1.

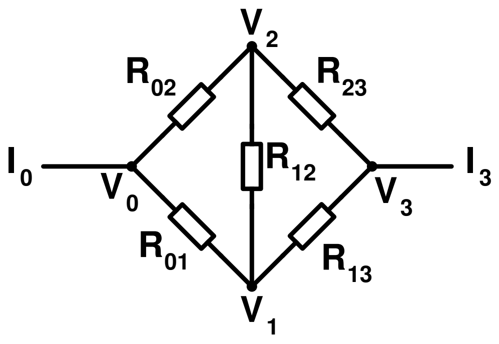

Figure 4.1: Circuit diagram of a Wheatstone Bridge.

We apply an external voltage, say

$$\begin{aligned} V_0 &= 0 \\ V_3 &= 1 \end{aligned} \tag{4.3}$$

and we want to know the currents I_0, I_3. Using Kirchhoff's current law we can describe each point as follows:

$$\begin{aligned}
&\frac{(V_0 - V_1)}{R_{01}} + \frac{(V_0 - V_2)}{R_{02}} + I_0 = 0 \\
&\frac{(V_1 - V_0)}{R_{01}} + \frac{(V_1 - V_3)}{R_{13}} + \frac{(V_1 - V_2)}{R_{12}} = 0 \\
&\frac{(V_2 - V_0)}{R_{02}} + \frac{(V_2 - V_3)}{R_{23}} + \frac{(V_2 - V_1)}{R_{12}} = 0 \\
&\frac{(V_3 - V_1)}{R_{13}} + \frac{(V_3 - V_2)}{R_{23}} + I_3 = 0
\end{aligned} \tag{4.4}$$

We now have six equations with six unknowns $V_0, V_1, V_2, V_3, I_0, I_3$. Of these, the two equations (4.3) are trivial. Moreover, we know on physical grounds that $I_3 = -I_0$. Hence we could reduce the system to three equations for the unknowns V_1, V_2, I_0. But it is rather convenient to leave the system as six equations. To see why, let us write the equations (4.3) and (4.4) in matrix form. For the case of all resistors being unity, the equations are as follows.

$$\begin{pmatrix} 2 & -1 & -1 & 0 & 1 & 0 \\ -1 & 3 & -1 & -1 & 0 & 0 \\ -1 & -1 & 3 & -1 & 0 & 0 \\ 0 & -1 & -1 & 2 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} V_0 \\ V_1 \\ V_2 \\ V_3 \\ I_0 \\ I_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{4.5}$$

The modification for arbitrary resistors is trivial. The 4×4 block on the upper left is what describes the network. It is a symmetric matrix with each row (and column) haveing zero sum. Such matrices are sometimes called Kirchhoff matrices.

Classically, the Wheatstone bridge was a device for measuring an unknown resistance. The resistor R_{12} would have been a galvanometer. Of the other four resistors, three were known (and one of them adjustable) while one would be unknown. The adjustable resistor was varied till there zero current through R_{12}. The situation then was $V_1 = V_2$. The circuit equations gave the unknown resistance.

Solve the equations (4.5) for the resistance of a Wheatstone bridge. Generalize to the resistor-cube problem. Then try out modifications where some of the resistors are replaced by inductances or capacitors.

4.13 Daylight

Figure 4.2 shows contours of daylight strength on a world map. In this example, we will generate such a map, with the help of rotation matrices.

For this we need four angular quantities.

(i) The latitude λ.

(ii) The inclination I of the Earth's axis.

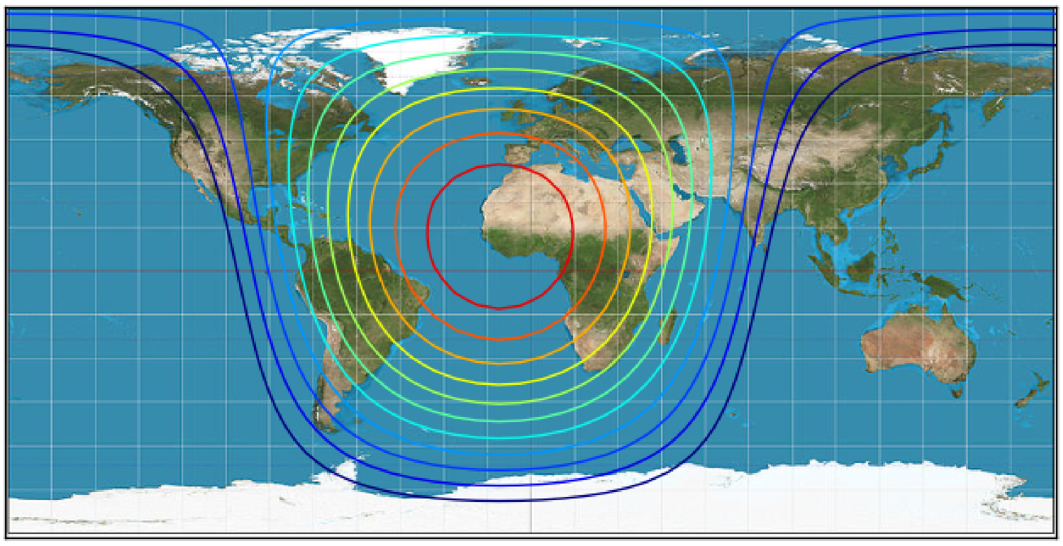

Figure 4.2: Daylight at noon CEST on 15 April. World map by 'Strebe', from Wikimedia commons. The projection is equirectangular (plate carrée).

(iii) An angular date δ, which is time in units such that 2π represents a year. The zero point of δ is noon UTC at the equinox.

(iv) The sidereal time σ. This is the longitude in the celestial coordinate system, and equals time in units such that 2π makes a day, plus the ordinary longitude plus δ.

With the above four angles in hand, the normal component of the Sun is given very simply in terms of rotation matrices.

$$\begin{pmatrix} 1 & 0 & 0 \end{pmatrix} R_y(\lambda)\, R_z(\sigma)\, R_x(I)\, R_z(-\delta) \tag{4.6}$$

We start with a unit vector along x. This we take to be the vertical direction at latitude and longitude of zero (close to Accra in Ghana) at equinox noon. Then we make consecutive rotations for the latitude λ, sidereal time, the Earth's inclination, and the angular date. The new x component is the daylight strength.

The rotation matrices as as follows. Note that angles must always be converted to radians before evaluating sin and cos.

$$R_x(\phi) \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{pmatrix} \qquad R_z(\phi) = \begin{pmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.7}$$

$$R_y(\phi) = \begin{pmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{pmatrix} \tag{4.8}$$

The rotations are composed from left to right. (If from right to left, all matrices would be transposed.)

We can get the current time in seconds from the system.

$$\begin{aligned}&\textbf{from } time \textbf{ import } time\\ &t = int(time()) - 1332331200\end{aligned} \tag{4.9}$$

By default the system measures time in seconds since midnight on January 1 1970, but (4.9) sets the zero of time to be noon UTC at the March equinox in 2012. From t we can easily calculate the date in units such that 2π makes a year.

Write a program to produce something like Figure 4.2, for the current time.

5. Polynomials

Integrating functions is one of the classical topics in numerical analysis. Nowadays, libraries such as *scipy.integrate* have excellent functions and in practice one rarely has to worry about the details of numerical integration. Nevertheless, it is interesting to explore the underlying ideas involved.

5.14 Newton-Cotes formulas

The simplest strategy in numerical integration, going back to Newton and his contemporaries if not even earlier, is to evaluate the integrand at equally-spaced points and fit polynomials through them.

To see how this works, let us approximate the integral

$$\int_{-1}^{1} f(x)\,dx \tag{5.1}$$

by the sum

$$c_{-1}f(-1) + c_0 f(0) + c_1 f(1) \tag{5.2}$$

with suitably chosen constants c_p. Now, if we make the formula symmetric (meaning set $c_{-1} = c_1$) the odd part of $f(x)$ will automatically integrate correctly to zero between. So we only need to worry about the even part. Requiring the formula to integrate $f(x) = 1$ and $f(x) = x^2$ correctly gives the following condition on the coefficients.

$$\begin{pmatrix} \frac{1}{2} & 1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \end{pmatrix} = \begin{pmatrix} 1 \\ \frac{1}{3} \end{pmatrix} \tag{5.3}$$

The solution is

$$c_0 = \frac{4}{3} \qquad c_1 = \frac{1}{3} \tag{5.4}$$

and is called Simpson's rule (sometimes Kepler's rule). As we required, it integrates up to x^3 correctly. But if $f(x) = x^4$ the answer is wrong: $\frac{2}{3}$ instead of $\frac{2}{5}$.

In practice, one would stack a bunch of Simpson's-rule blocks to do an integration, more small blocks for more accuracy.

To improve on Simpson's rule, let us consider

$$\int_{-2}^{2} f(x)\,dx \tag{5.5}$$

and the approximate sum

$$c_{-2}f(-2) + c_{-1}f(-1) + c_0 f(0) + c_1 f(1) + c_2 f(2) \tag{5.6}$$

The condition that 1, x^2 and x^4 are integrated correctly gives

$$\begin{pmatrix} \frac{1}{2} & 1 & 1 \\ 0 & 1 & 2^2 \\ 0 & 1 & 2^4 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 2 \\ \frac{1}{3}2^3 \\ \frac{1}{5}2^5 \end{pmatrix} \tag{5.7}$$

The solution is

$$c_0 = \frac{8}{15} \qquad c_1 = \frac{64}{45} \qquad c_2 = \frac{14}{45} \tag{5.8}$$

and is sometimes known as Boole's rule.

In this way, we can go to arbitrarily high order.

In Simpson's rule and its higher-order cousins, there is always an evaluation at the endpoints of the interval. This is undesirable if $f(x)$ has a singularity at an endpoint. A way to get around this problem is to use a so-called open formula, which does not evaluate the function at the endpoints. Consider approximating

$$\int_{-3}^{3} f(x)\,dx \tag{5.9}$$

by the sum

$$c_{-2}f(-2) + c_0 f(0) + c_2 f(2) \tag{5.10}$$

The condition of integrating 1 and x^2 correctly gives

$$\begin{pmatrix} \frac{1}{2} & 1 \\ 0 & 2^2 \end{pmatrix} \begin{pmatrix} c_0 \\ c_2 \end{pmatrix} = \begin{pmatrix} 3 \\ \frac{1}{3}3^3 \end{pmatrix} \tag{5.11}$$

whose solution is

$$c_0 = \frac{3}{2} \qquad c_2 = \frac{9}{4} \tag{5.12}$$

This is an open-type analog of Simpson's rule.

Generalize the above open-type formula to find the coefficients in

$$c_{-4}f(-4) + c_{-2}f(-2) + c_0 f(0) + c_2 f(2) + c_4 f(4) \tag{5.13}$$

such that

$$\int_{-5}^{5} f(x)\,dx \tag{5.14}$$

is integrated correctly up to x^5. For the matrix equation involved, the *solve*() function from *scipy.linalg* is convenient.

Try and express the coefficients as fractions, as in (5.8). You can derive a fraction by first expanding the floating-point values of the coefficients as continued fractions, and truncating when the terms become extremely large as a result of roundoff error.

5.15 Arbitrary integration intervals

Implement the five-point formula (5.13) as a function.

```
def fivepoint(f, a, b, B = 1):                                  (5.15)
    # Compute $\int_a^b f(x)\,dx$
    # using $B$ blocks of the 5-point integrator.
```

Use the function to numerically compute

$$\int_{-\infty}^{\infty} e^{-x^2}\, dx \tag{5.16}$$

To do this, you will need to change variable to make the interval finite. (The exact answer is $\sqrt{\pi}$.)

5.16 Legendre polynomials

A Legendre polynomial $P_n(x)$ is a polynomial of degree n (the first few are shown in Figure 5.1) such that

$$\int_{-1}^{1} P_m(x)\, P_n(x)\, dx = \frac{1}{n+\frac{1}{2}}\, \delta_{mn} \tag{5.17}$$

Legendre polynomials have many nice properties and appear in many physical problems.

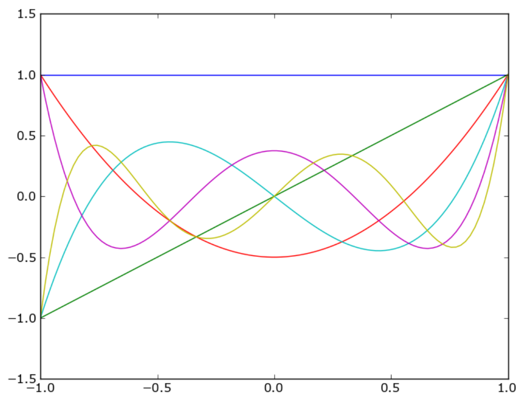

Figure 5.1: Legendre polynomials from P_0 to P_5.

The orthogonality relation (5.17) is usually considered as a derived property rather than a definition. However, it is interesting to adopt (5.17) as the definition and derive the coefficients using numerical integration.

A remark about evaluating polynomials. While it is not wrong to evaluate a polynomial with something like

```
**def** $evalp(c, x)$ :
    $sum = 0$
    **for** $k$ **in** $range(len(c) + 1)$ :                (5.18)
        $sum = c[k] * pow(x, k)$
    **return** $sum$
```

it is considered amateurish. It is better to avoid explicit powers by expressing polynomials as nested sums and products, for example

$$1 + 2x + 3x^2 + 4x^3 = 1 + x(2 + x(3 + x(4))) \tag{5.19}$$

One way to avoid calling $pow()$ would be as follows.

```
def $evalp(c, x)$:
    $n = len(c) - 1$
    $sum = c[-1]$
    for $k$ in $range(n, 0, -1)$:
        $sum = x * sum + c[k-1]$
    return $sum$
```
(5.20)

Compute the coefficients of the first few $P_n(x)$ using your implementation of (5.15) and Gram-Schmidt orthonormalization, and plot the polynomials. It is convenient to first assume the rhs of (5.17) as simply δ_{mn} and apply the factor of $(n+\frac{1}{2})^{-\frac{1}{2}}$ later. Large B gives more accuracy. But the first few (P_0, P_1, P_2) are correct to roundoff precision even with $B = 1$. Why is that?

5.17 Chebyshev polynomials and Gauss-Chebyshev quadrature

Chebyshev polynomials $T_n(x)$ can be defined by the orthogonality relation

$$\int_{-1}^{1} \frac{T_m(x)\,T_n(x)}{\sqrt{1-x^2}}\,dx = \frac{\pi}{2}\delta_{mn} \tag{5.21}$$

except for $m = n = 0$ when the rhs is π instead. The first few are shown in Figure fig:chebyshev. Note an interesting difference from the Legendre polynomials: the Chebyshev polynomials always turn over at ± 1. This property makes Chebyshev polynomials very important for approximating functions.

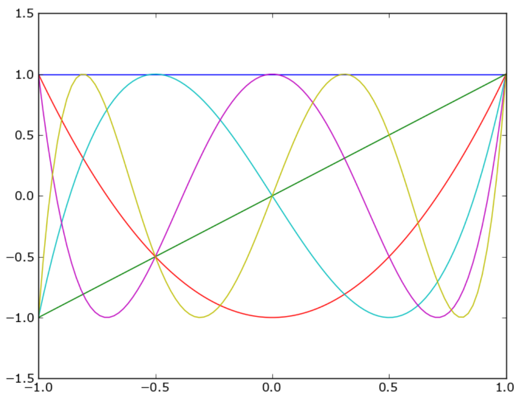

Figure 5.2: Chebyshev polynomials from T_0 to T_5.

Compute and plot the first few Chebyshev polynomials using (5.15) and Gram-Schmidt orthonormalization as before.

The Newton-Cotes formulas do not do very well at deriving the T_n, because of the $(1-x^2)^{-1/2}$ singularity. The cure is to transform the singularity away:

$$\int_{-1}^{1} \frac{f(x)}{\sqrt{1-x^2}}\,dx = \int_{0}^{\pi} f(\cos\theta)\,d\theta \tag{5.22}$$

In fact, we can do even better. We can see from (5.21) that Chebyshev polynomials are simply cosines in disguise.

$$T_n(\cos\theta) = \cos(n\theta) \tag{5.23}$$

Now, cosines satisfy a very useful identity

$$\sum_{k=0}^{M-1} \cos((k+\tfrac{1}{2})m\pi/M) = 0 \qquad \text{for } m = 1,\dots,2M-1 \tag{5.24}$$

which is a discrete orthogonality relation. This is the basis of the Discrete Cosine Transform (DCT) used, for example, for `jpeg` encoding. It is not hard to show from (5.24) that

$$\frac{\pi}{M}\sum_{k=0}^{M-1} f\left(\cos((k+\tfrac{1}{2})\pi/M)\right) = \int_0^\pi f(\cos\theta)\,d\theta \tag{5.25}$$

exactly, provided the Fourier series for $f(\cos\theta)$ has only terms lower than $\cos(2M\theta)$. Formula (5.25) is thus an integration formula that integrates polynomials of degree $2M-1$, using only M evaluations. It is known as Gauss-Chebyshev quadrature (the general idea is known as Gauss quadrature).

Compute and plot the first few Chebyshev polynomials again, this time using Gauss-Chebyshev quadrature and Gram-Schmidt orthonormalization. It will turn out that if $n < M$ (M being the number of evaluations) T_n is given perfectly, but beyond that the results are garbage. Why is that?

6. Fourier transforms

6.18 Fourier transform

A Fourier transform is defined as

$$\begin{aligned} F(k) &= \int_{-\infty}^{\infty} \exp(ikx)\, f(x)\, dx \\ f(x) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} \exp(-ikx)\, F(k)\, dk \end{aligned} \tag{6.1}$$

It has many interesting properties. Three of these will be particularly useful to us.

(1) The Fourier transform of $f'(k)$ is $-ikF(k)$. This means that, by going into Fourier space, we can replace differentiation by multiplication.

(2) The Fourier transform of a Gaussian is also Gaussian.

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\tfrac{1}{2}x^2/\sigma^2\right) \iff F(k) = \exp\left(-\tfrac{1}{2}k^2\sigma^2\right) \tag{6.2}$$

(3) The convolution theorem

$$\int_{-\infty}^{\infty} f(y)\, g(x-y)\, dy = \frac{1}{2\pi} \int_{-\infty}^{\infty} \exp(-ikx)\, F(k)\, G(k)\, dx \tag{6.3}$$

The special case $f = g$ and $x = 0$ is known as Parseval's relation.

A discrete Fourier transform is defined as follows. An array f_x where $x \in \{0, \ldots, N-1\}$ is transformed into another array F_k of the same size

$$F_k = \sum_{x=0}^{N-1} \exp\left(\frac{2\pi ikx}{N}\right) f_x \tag{6.4}$$

Note that in (6.4) x and k are array indices, whereas in (6.1) they are arguments of functions. They are related by

$$\Delta x\, \Delta k = \frac{2\pi}{N} \tag{6.5}$$

We can choose any two of Δx, Δk and N.

The sum (6.4) is only a first-order approximation to the first integral in (6.1) and one could do much better. However, the definition (6.4) has two important properties to recommend it. First, it is exactly unitary, in the sense that reversing the sign of k gives an exact inverse (apart from a factor of N). Second, it can be computed very efficiently. As it stands, (6.4) would appear to need $O(N^2)$ time to compute. But in fact it can be computed in $O(N \ln N)$ time. To see why, let us split the sum into two sums.

$$\begin{aligned} F_k &= \sum_{x=0}^{N/2-1} \exp\left(\frac{2\pi ik(2x)}{N}\right) f_{2x} + \sum_{x=0}^{N/2-1} \exp\left(\frac{2\pi ik(2x+1)}{N}\right) f_{2x+1} \\ &= \sum_{x=0}^{N/2-1} \exp\left(\frac{2\pi ikx}{N/2}\right) f_{2x} + \exp\left(\frac{2\pi ki}{N}\right) \sum_{x=0}^{N/2-1} \exp\left(\frac{2\pi ikx}{N/2}\right) f_{2x+1} \end{aligned} \tag{6.6}$$

We have replaced the original N-fold sum with two sums of half the size. Now consider what happens if we replace k with $k + N/2$. The the sums in the second line of 6.6 remain the same, but the prefactor $\exp(2\pi ki/N)$ changes to $\exp(-\pi i)\exp(2\pi ki/N)$ or $-\exp(2\pi ki/N)$. Introducing the notation

$$\omega_N^{kx} \equiv \exp\left(\frac{2\pi ikx}{N}\right) \tag{6.7}$$

gives

$$\begin{aligned} F_k &= \sum_{x=0}^{N/2-1} \omega_{N/2}^{kx} f_{2x} + \omega_N^k \sum_{x=0}^{N/2-1} \omega_{N/2}^{kx} f_{2x+1} \\ F_{k+N/2} &= \sum_{x=0}^{N/2-1} \omega_{N/2}^{kx} f_{2x} - \omega_N^k \sum_{x=0}^{N/2-1} \omega_{N/2}^{kx} f_{2x+1} \end{aligned} \tag{6.8}$$

If N is a power of 2, the sums can be split recursively. Analogous decompositions are possible[1] for any factors of N. But if N is prime, no decomposition is possible.

Fast Fourier transform (FFT) refers to an $O(N \ln N)$ implementation of (6.4). It is conveniently available as *fft* and *ifft* in *scipy*.

```
from numpy import pi, arange, concatenate
from scipy import fft
N = pow(2,6)
L = 8
dx = 2. * L/N
x = (arange(N) - N/2) * dx
k = 2 * pi/(N * dx) * (arange(N) - N/2)
f = 1/(1 + x * x)
F = fft(f)
F = concatenate((F[N/2 : N], F[0 : N/2]))
```
(6.9)

We can plot the results as follows.

```
from pylab import subplot, plot, show
subplot(212)
plot(x, f)
subplot(211)
plot(k, F.real, color = "blue")
plot(k, F.imag, color = "magenta")
show()
```
(6.10)

6.19 Fourier series

A Fourier series is in a sense intermediate between the discrete and continuous transforms.

$$f(x) = \sum_{n=-\infty}^{\infty} c_n e^{-inx} \qquad c_n = \frac{1}{2\pi}\int_0^{2\pi} f(x)\, e^{inx}\, dx \tag{6.11}$$

[1] Gauss remarked on this circa 1805, but nobody seems to have realized what he was talking about until the 1960s.

Separate real and imaginary parts, write $c_n = \frac{1}{2}(a_n + ib_n)$ and $c_{-n} = \frac{1}{2}(a_n - ib_n)$.

$$
\begin{aligned}
f(x) &= \tfrac{1}{2}a_0 + \sum_{n=1}^{\infty} (a_n \cos nx + b_n \sin nx) \\
a_n &= \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos nx\,dx \qquad b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin nx\,dx
\end{aligned}
\tag{6.12}
$$

As an example, let us consider a square wave

$$
f(x) = \begin{cases} 1 & \text{if } x > 0 \\ -1 & \text{if } x < 0 \end{cases} \tag{6.13}
$$

in the domain $[-\pi, \pi]$ and periodic outside. This gives

$$
f(x) = \frac{4}{\pi}\sum_{n\ \text{odd}} \frac{\sin nx}{n} \tag{6.14}
$$

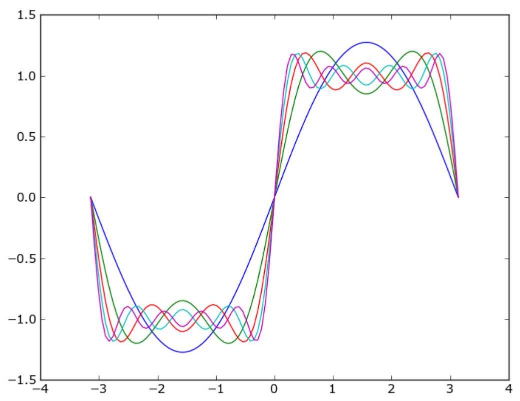

Figure 6.1: Fourier series for a square wave.

Figure 6.1 shows the square wave and its approximations by its Fourier series. Several things are noticeable:

(i) even a square wave, which looks very unlike sines and cosines, can be approximated by them, to any desired accuracy;

(ii) though we only considered the domain $[-\pi, \pi]$ the Fourier series automatically extends the domain to all real x by generating a periodic answer;

(iii) at discontinuities the Fourier series gives the mean value;

(iv) close to discontinuities the Fourier series overshoots.

Properties (i) to (iii) apply whenever $f(x)$ has a finite number of discontinuties and extrema (the Dirichlet conditions). The curious effect (iv) is called Gibbs' phenomenon—and adding more terms does not reduce the overshoot, it just moves the overshoot closer to the discontinuity.

Putting $x = -\frac{\pi}{2}$ in (6.14) gives an unexpected identity:

$$1 - \tfrac{1}{3} + \tfrac{1}{5} - \tfrac{1}{7} + \ldots = \frac{\pi}{4} \,. \tag{6.15}$$

Parseval relation

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} |f(x)|^2 \, dx = \sum_{n=-\infty}^{\infty} |c_n|^2 \tag{6.16}$$

or

$$\frac{1}{\pi} \int_{-\pi}^{\pi} f^2(x) \, dx = \tfrac{1}{2} a_0^2 + \sum_{n=1}^{\infty} (a_n^2 + b_n^2) \tag{6.17}$$

Using this we can derive

$$1 + \tfrac{1}{3^2} + \tfrac{1}{5^2} + \tfrac{1}{7^2} + \ldots = \frac{\pi^2}{8} \tag{6.18}$$

Find a Fourier series for $f(x) = x^2$ and use it to derive another series for π^2 and a series for π^4.

6.20 Hilbert spaces

One of the earliest

$$\begin{aligned} &\textbf{from } \mathit{numpy.linalg} \textbf{ import } \mathit{eig} \\ &\mathit{vals},\ \mathit{Vecs} = \mathit{eig}(M) \end{aligned} \tag{6.19}$$

$$\begin{aligned} &\mathit{sa} = \mathit{vals.argsort}() \\ &\mathit{vals} = \mathit{vals}[\mathit{sa}] \\ &\mathit{Vecs} = \mathit{Vecs}[\,:, \mathit{sa}] \end{aligned} \tag{6.20}$$

Consider

$$T_{lm} = 2\delta_{lm} - \delta_{l,m+1} - \delta_{l,m-1} \tag{6.21}$$

We know the eigenvectors

$$E_{kl} = \frac{e^{ikl\omega}}{\sqrt{N}} \qquad \omega = \frac{2\pi}{N} \tag{6.22}$$

where N is the number of dimensions. Clearly

$$\sum_l E_{kl} E_{lm}^* = \delta_{kn} \tag{6.23}$$

It is straightforward to verify that

$$\sum_{lm} E_{kl} T_{lm} E_{mn}^* = 4 \sin^2\left(\tfrac{1}{2} k\omega\right) \, \delta_{kn} \tag{6.24}$$

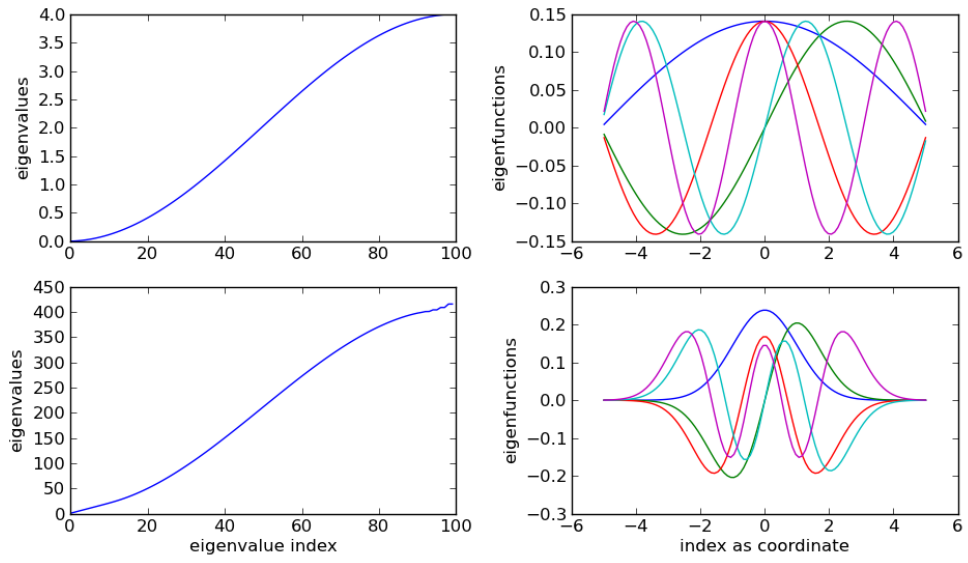

Figure 6.2: Eigenvalue spectrum (left) and the first few eigenfunctions (right) of the matrices T (upper) and H (lower).

For small k, the eigenvalues are $\lambda_k \approx k^2\omega^2$.

6.21 Schroedinger equation

In this program we solve the 1D Schrödinger equation

$$i\,\frac{\partial\psi}{\partial t} = -\tfrac{1}{2}\,\frac{\partial^2\psi}{\partial x^2} + V(x)\,\psi \tag{6.25}$$

Stationary solutions satisfy

$$-\tfrac{1}{2}\,\frac{\partial^2\psi}{\partial x^2} + V(x)\,\psi = E\psi \tag{6.26}$$

Fourier transforms are a powerful way of solving this type of partial differential equation. If we introduce the Fourier transform

$$\tilde{\psi}(k) = \int_{-\infty}^{\infty} e^{ikx}\psi(x)\,dx \tag{6.27}$$

equation (6.25) turns into

$$i\,\frac{\partial\tilde{\psi}}{\partial t} = \tfrac{1}{2}k^2\tilde{\psi} + \int_{-\infty}^{\infty} e^{ikx}\,V\psi\,dx \tag{6.28}$$

and now the first term on the right is trivial whereas the last term is an operator.

For

$$i\,\frac{d\psi}{dt} = H\psi \tag{6.29}$$

time evolution can be expressed formally as

$$\exp(-it\,H) \tag{6.30}$$

Given a Hamiltonian

$$H = A + B \tag{6.31}$$

where A and B individually are integrable, the Baker-Campbell-Hausdorff identity

$$\begin{gathered}\exp\left(\tfrac{1}{2}\tau i\,A\right)\,\exp(\tau i\,B)\,\exp\left(\tfrac{1}{2}\tau i\,A\right) = \exp\left(\tau i H_1\right)\\ H_1 = H + \tau^2\Big(\tfrac{1}{12}[[A,B],B] - \tfrac{1}{24}[A,[A,B]]\Big) + O(\tau^4)\end{gathered} \tag{6.32}$$

where the square brackets denote commutators.

Implement a *Tkinter* animation of time evolution under the Schrödinger equation. Choose some non-trivial potential, such as the double well $V(x) = \frac{1}{2}|x-1|^2$, which illustrates quantum tunnelling.

7. Random walks and Markov chains

7.22 Quasi-random numbers

Random numbers on a computer are sometimes called pseudo-random numbers, to distinguish them from truly random numbers that would arise from a quantum process. Quasi-random numbers are different again.

A nice example of quasi-random numbers is a Halton sequence. To generate such a sequence, we write the integers from 1 to N in base 2 or some other prime base, and move the digits in reverse order to the right of the radix point. This naturally yields a number in $(0, 1)$. In more than one dimension, we simply use a different prime base for each dimension. Figure 7.1 illustrates.

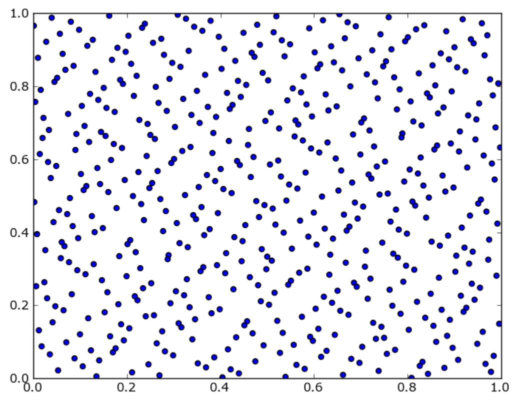

Figure 7.1: The first 512 numbers of a two-dimensional Halton sequence, using base 2 for x and base 3 for y.

Compute and plot a Halton sequence in two dimensions, and contrast it with random numbers.

7.23 Random Walks and the Central Limit Theorem

In Figure 7.2 we the see distribution of endpoints of a random walk. The walk consists of $N =$ steps of unit length but with random orientation. The endpoint has the probability distribution

$$p(x, y) = \frac{1}{N\pi} e^{-(x^2+y^2)/N} \tag{7.1}$$

that is, a Gaussian distribution. The probability distribution for the distance is

$$p(r) = \frac{2r}{N} e^{-r^2/N} \tag{7.2}$$

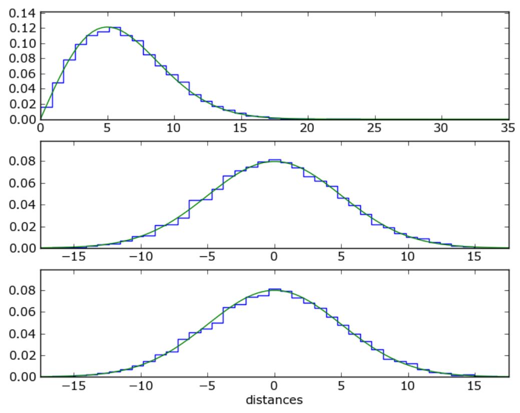

Figure 7.2: Endpoints of 10^4 realizations of a 50-step random walk, shown the Gaussian approximation. Upper panel shows the distance, middle and lower panels are the x and y distances.

or a Maxwellian distribution.

Random walks are an illustration of the central limit theorem in probability theory. This theorem asserts that mean of N trials of an arbitrary probability distribution tends to a Gaussian distribution, with the mean remaining the same and the variance being reduced by a factor of N. In particular, for the above random walk, it predicts (7.1), since the variance of a single step is $\frac{1}{2}$.

To prove the central limit theorem, we note first that when we have two independent trials from some $p(x)$, the probability distribution of the sum is a convolution.

$$p(\text{sum of two trials} = x) = \int p(y)\, p(x-y)\, dy \tag{7.3}$$

Hence from the convolution theorem (6.3) we have

$$p(\text{sum of } N \text{ trials} = x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \exp(-ikx)\, P^N(k)\, dx \tag{7.4}$$

where $P(k)$ (known as the characteristic function) is the Fourier transform of the probability distribution $p(x)$. We can write

$$P(k) = \left\langle e^{ikx} \right\rangle = 1 + ik \langle x \rangle - \frac{k^2}{2!} \left\langle x^2 \right\rangle + \ldots \tag{7.5}$$

Assuming $\langle x \rangle$ and $\langle x^2 \rangle$ exist, it is enough to consider the case of $\langle x \rangle = 0$ and $\langle x^2 \rangle = 1$. Then

$$P(k) = 1 - \tfrac{1}{2}k^2 + k^2\theta(k) \tag{7.6}$$

where the remainder term $\theta(k) \to 0$ as $k \to 0$. For large N

$$P(k) \to e^{-Nk^2/2} \tag{7.7}$$

and using the Fourier-transform identity (6.2) we have

$$p(\text{sum of } N \text{ trials} = x) = \left(\frac{N}{2\pi}\right)^{\frac{1}{2}} \exp\left(-\frac{x^2}{2N}\right) \tag{7.8}$$

It is important to note that the central limit theorem requires that the variance exists. So-called heavy-tailed distributions like the Lorentzian

$$p(x) = \frac{1}{\pi(1+x^2)} \qquad P(k) = e^{-|k|} \tag{7.9}$$

do not have a finite variance and the central limit theorem does not apply.

Choose a different type of step and generate a figure, analogous to Figure 7.2, verifying the Gaussian and Maxwellian properties.

7.24 Arnold's five-minute problem

The mathematician Vladimir I. Arnold (1937–2010) is famous for his contributions to dynamical systems, catastrophe theory, and several other areas of mathematics. His efforts to reform the teaching of mathematics are also rather well known. He once wrote:

> A student who takes much more than five minutes to calculate the mean of $\sin^{100} x$ with 10% accuracy has no mastery of mathematics, even if he has studied non-standard analysis, universal algebra, supermanifolds, or embedding theorems.

Show your mastery of mathematics by solving this problem. Since sines and cosines are periodic, it is enough to estimate

$$\int_{-\pi}^{\pi} \cos^N x \, dx \tag{7.10}$$

assuming $N \gg 1$. No programming is required, but the proof of the central limit theorem suggests one way (not the only way) of estimating the integral.

7.25 Turing and ESP

To be written

7.26 Hypothesis Testing

A variant of the standard random walk is a walk that is constrained to return to its starting point. It helps introduce the topic of hypothesis testing in statistics.

Imagine a party, where among the other offerings there are two trays of identical chocolates, initially an equal number of pieces on each tray. As the chocolates gradually disappear, you are discreetly watching, while entertaining the people around you with a puppet. Each time a chocolate is taken from one of the trays, you make the puppet take a step forward; and whenever a chocolate is taken from the other tray, you make the puppet take a step back. When both trays are empty, the puppet will be back at its original position. The maximum distance the puppet gets to, whether forward or back, is known as the Kolmogorov-Smirnov statistic (or KS statistic).

More formally, the KS statistic is the maximum difference between two cumulative probability distributions. Think of the distribution of the time that chocolates spend on a tray before being eaten. The corresponding cumulative distribution gives the number of chocolates remaining on the tray at any time.

The KS statistic has the following very important property. Provided the two trays have the same probability distributions (the chocolates are equivalent, and people do not compare the trays), the KS statistic does not care what that distribution actually is. The people at the party could be the sort who eat the chocolates very slowly, or complete gluttons who devour them instantly. It could be that the chocolates disappear very quickly at first, but then people become shy and eat more slowly. The distribution of the KS statistic (meaning here a distribution over parties, since the statistic has only one value per party) will be the same. Think of the puppet randomly taking a forward or backward step. This property makes the KS statistic a good test for when the probability distributions are not the same. What matters is whether the KS statistic has a typical or extreme value. We can ask questions of the type “Is dark chocolate more (or less) popular than milk chocolate?” and answer them from even a single party.

Figure 7.3 shows the distribution of the KS statistic: the histogram is a sort of Monte-Carlo over many parties, while the curve is an asymptotic formula derived in different ways by Kolmogorov and Smirnov.

$$\begin{aligned} p_{\mathrm{KS}}(s) &= 2\sum_{k=0}^{\infty}(-1)^k\, e^{-2(k+1)^2\mu^2 s^2} \\ \mu &\simeq \nu + 0.12 + 0.11/\nu \qquad \nu = \sqrt{DM/(D+M)} \end{aligned} \tag{7.11}$$

The initial number of chocolates in the two trays is allowed to be different: D and M. We allow for this by making the forward steps longer (or shorter) than the backward steps, such that the puppet still ends up at the origin when all the chocolates are gone.

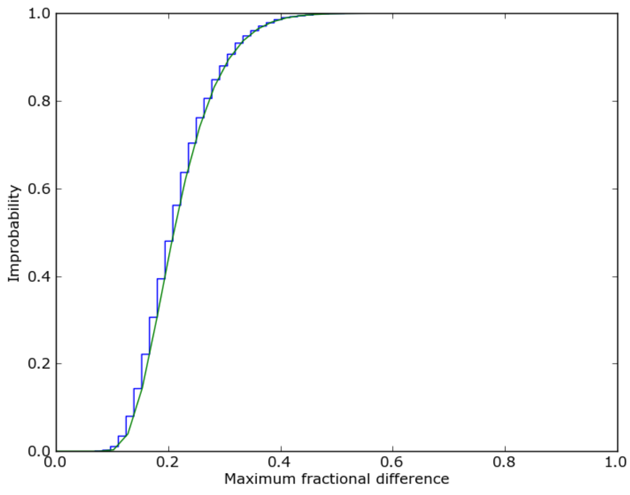

Figure 7.3: The Kolmogorov-Smirnov statistic and its improbability for the case of $N_1 = 24, N_2 = 36$. The histogram is from a simulation and the curve is $1 - p_{\mathrm{KS}}(s)$ from the formula (7.11)

From Figure 7.3 we can read off the following. If we put down two dozen dark chocolates and three dozen milk chocolates, and then find that the dark chocolates are all gone when half the milk chocolates are still left, then we would infer that dark chocolate is significantly more popular; but if only a quarter of the milk chocolates were left, there would be no such inference.

Write a program to simulate the distribution of the KS statistic and verify against the formula (7.11). The invent your own statistic, based on the puppet's walk, and compute its distribution.

7.27 Vermeer's window

The paintings of Johannes Vermeer are considered some of the supreme representations of light and shadows in art. Some of his paintings also illustrate striking examples of perspective, and it thought that Vermeer may have used some form of pinhole camera.[2] In this section, we choose a picture and try to reconstruct the location and orientation of the pinhole camera.

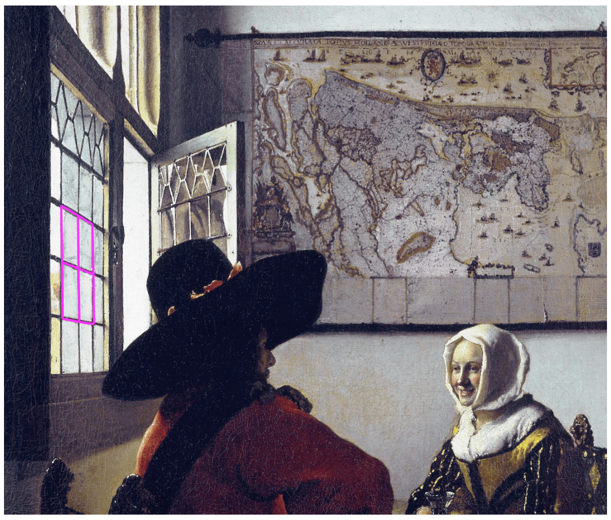

Figure 7.4: Detail from *De soldaat en het lachende meisje* (painted by Vermeer circa 1660) with grid overlaid.

Figure 7.4 shows an example[3] of a painting with strong perspective. The perspective is evident in several aspects of the picture, but in the window it is especially clear. The window seems to consist of square glass panes. Let us assume the glass panes are identical squares. The magenta grid overlaid on the picture mimics the perspective.

Consider the following steps.

(1) We start by selecting an image file and measuring the pixel coordinates of nine points on the image, on which we want to overlay the grid.

[2] `http://www.essentialvermeer.com/camera_obscura/co_one.html`

[3] See also `http://www.googleartproject.com/museums/frick/officer-and-laughing-girl-6`

(2) Next, we initialize a square grid.

$$\begin{array}{ccc} (-1,1) & (0,1) & (1,1) \\ (-1,0) & (0,0) & (1,0) \\ (-1,-1) & (0,-1) & (1,-1) \end{array} \tag{7.12}$$

Choosing 1 for the grid size amounts to taking the size of the glass panes as the unit of length.

(3) Next, rotate this grid around a vertical axis by 90°. Quaternions are convenient for this, but not essential. If you like, you can adjust the rotation axis and rotation angle (with additional parameters) but rotation about a vertical axis by 90° turns out to be a very good approximation.

(4) Then, translate the rotated grid in the room. This part needs three parameters, and is basically the placement of the pinhole camera with respect to the window.

(5) Finally, put the grid on the image using the perspective formula (10.14). The observer distance is a fourth parameter, and represents the distance between pinhole and the picture in our units of length (the glass-pane size), times the number pixels in a length unit.

Work out the four parameters described above, using a least-squares fit to the measured pixels. The *leastsq* function from *scipy.optimize* will do the numerical hard work for you.

7.28 Markov chain Monte-Carlo

A general type of random walk would be to walk from point to point according to some probability distribution:

$$p(x \to y) \qquad \textstyle\sum_y p(x \to y) = 1 \tag{7.13}$$

being the probability of stepping from x to y. If such a random walk satisfies the so-called micro-reversibility condition

$$f(x)\, p(x \to y) = f(y)\, p(y \to x) \tag{7.14}$$

for some function $f(x)$, then the random walk tends to sample $f(x)$.

To show the above property, let us consider a sequence function $f_n(x)$ related by

$$f_{n+1}(x) = \textstyle\sum_y f_n(y)\, p(y \to x) \tag{7.15}$$

If $f_n(x) = f(x)$ we can apply the microreversibility condition (7.14) to (7.15). Then it follows from (7.13) that $f_{n+1}(x) = f_n(x)$. In other words, if $f_n(x)$ has converged to $f(x)$, further iterates will stay there.

To see why $f_n(x)$ tends to approach $f(x)$, we take

$$|f(x) - f_{n+1}(x)| = \left| f(x) - \textstyle\sum_y f_n(y)\, p(y \to x) \right| \tag{7.16}$$

Using (7.13) and (7.14) we can replace

$$f(x) = \sum_y f(x)\, p(x \to y) = \sum_y f(y)\, p(y \to x) \tag{7.17}$$

This gives

$$|f(x) - f_{n+1}(x)| = \left| \textstyle\sum_y \left(f(y) - f_n(y) \right) p(y \to x) \right| \tag{7.18}$$

and then, the triangle inequality tells us that

$$|f(x) - f_{n+1}(x)| \le \textstyle\sum_y |f(y) - f_n(y)|\; p(y \to x) \tag{7.19}$$

Summing now over x, we get

$$\sum_x |f(x) - f_{n+1}(x)| \le \sum_y |f(y) - f_n(y)| \tag{7.20}$$

which implies convergence.

There are many techniques for sampling based on random walks with the micro-reversibility property (7.14). These are known as Markov chain Monte-Carlo (or simply MCMC) methods. The best known of these is the Metropolis algorithm, which proceeds as follows. From a point x_n, propose a random trial step Δx and then set

$$x_{n+1} = \begin{cases} x_n + \Delta x & \text{if } r < f(x_n + \Delta x)/f(x_n) \\ x_n & \text{otherwise} \end{cases} \tag{7.21}$$

A simple but interesting application of the Metropolis algorithm is the Bayesian lighthouse problem. Imagine lighthouse out at sea, near a straight length of shoreline. Let the x axis represent the shoreline, and let the lighthouse be at (a, b). The lighthouse has a rotating beacon which sends out flashes in random directions ϕ. These flashes are observed along the shoreline.

The above scenario generates a sequence of random numbers $\{x_i\}$ distributed according to

$$x = a + b \tan\phi \tag{7.22}$$

where ϕ is uniform random. For definiteness we assume $b > 0$ and $x \in [-1, 1]$. Rearranging (7.22) we have

$$\phi = \arctan \frac{x - a}{b} \tag{7.23}$$

and because ϕ is uniform random, it follows that the distribution of x is

$$p(x) = \frac{d\phi}{dx} \tag{7.24}$$

Here $p(x)$ is really a probability density along x. The condition that $x \in [-1, 1]$ gives the normalization

$$p(x|x \in [-1, 1]) = \frac{p(x)}{\int_{-1}^{1} p(x)} \tag{7.25}$$

which simplifies to

$$p(x|x \in [-1,1]) = \frac{d\phi/dx}{\phi|_{x=1} - \phi|_{x=-1}} \tag{7.26}$$

The joint probability of a data set $\{x_1, \ldots, x_N\}$ is

$$p(\{x_1, \ldots, x_N\}) = \prod_{i=1}^{N} p(x_i) \tag{7.27}$$

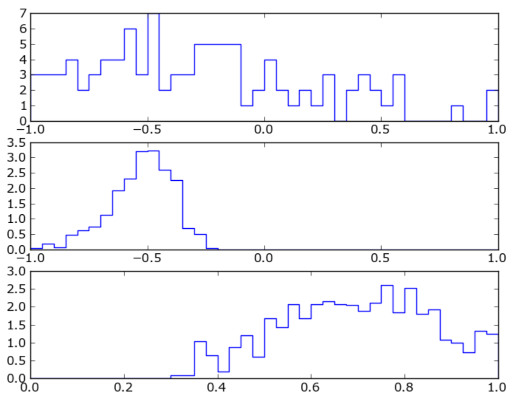

Figure 7.5: Lighthouse with 100 flashes with $a = -0.5, b = 0.7$. The top panel is a histogram of the x values. The middle and lower panels are the inferred probability distributions of a and b respectively.

Choose some values for a, b and then generate a simulated data set $\{x_1, \ldots, x_N\}$. Then use the Metropolis algorithm to infer the values of a, b with uncertainties. On the way, you will need to derive an expression for the probability (7.26) in terms of a and b.

8. Some dynamical systems

8.29 Euler and Runge-Kutta integrators

An integral can also be thought of as a differential equation.

Ordinary differential equations have the general form

$$\frac{dy}{dx} = f(y,x) \tag{8.1}$$

where y can be a vector, but x is a scalar variable. If the right hand side depends only on x, in other words, $f = f(x)$, we have an integral. Since differential equations appear in just about any area of science, and *scipy* offers tools to solve them numerically. Consider the following example.

```
from scipy.integrate import odeint
from numpy import exp
def f(y, x) :
    return exp(-x * x/4)
res = odeint(f, 0, range(12))
```
(8.2)

This computes

$$\int_0^X e^{-x^2/4}\, dx \tag{8.3}$$

for different values of the upper limit, so we can see convergence to the answer for $X \to \infty$.

Before using the library methods, we will try to understand how they work, by implementing some basic algorithms for solving ordinary differential equations numerically.

The simplest such method is based on truncating the Taylor series after one term:

$$y(x+h) = y(x) + hf(y(x), x) + O(h^2) \tag{8.4}$$

since $y' = f(y,x)$. The algorithm is

$$y_{n+1} = y_n + h\, f(y_n, x_n) \tag{8.5}$$

where y_n means $y(x)$ and y_{n+1} means $y(x+h)$. The error term will be $\frac{1}{2}h^2 f'(y_n)$. Over a given interval in x the error will be $O(h)$. This method is very crude and only useful for the simplest problems.[4]

One can improve on (8.5) by using the value of $f(y,x)$ at some other point to get an estimate of $f'(y,x)$. This is the strategy behind the Runge-Kutta integrators. For notational simplicity we now assume $f(y,x)$ has no explicit dependence on x. This

[4] It is called Euler's method, but it is hard to believe Euler could not invent anything better.

is to say we look at problems like $dy/dx = 2y$ (whose solution is $y = e^{2x}$), but not $dy/dx = 2xy$ (which has a solution $y = e^{x^2}$). In the later case, we could include x as one of the components of y, so there is no loss of generality in the assumption.

The simplest Runge-Kutta integrator is

$$y_{n+1} = y_n + hf\big(y_n + \tfrac{1}{2}hf(y_n)\big) \tag{8.6}$$

and by Taylor expanding (and substituting 8.1) we can see that

$$y_{n+1} = y_n + hf(y_n) + \tfrac{1}{2}h^2 f'(y_n) + O(h^3) \tag{8.7}$$

Over a given interval, the error is $O(h^2)$ and the method is second order. Many high-order Runge-Kutta formulas are in wide use. The best known is fourth-order Runge-Kutta

$$\begin{aligned} f_1 &= f(y_n) \\ f_2 &= f(y_n + \tfrac{1}{2}hf_1) \\ f_3 &= f(y_n + \tfrac{1}{2}hf_2) \\ f_4 &= f(y_n + hf_3) \\ y_{n+1} &= y_n + \tfrac{1}{6}h(f_1 + 2f_2 + 2f_3 + f_4) \end{aligned} \tag{8.8}$$

Chebyshev polynomials satisfy the differential equation

$$(1 - x^2)T_n'' - xT_n' + n^2 T_n = 0 \tag{8.9}$$

with the initial conditions

$$T_n(0) = \cos(n\pi/2) \qquad T_n'(0) = n\sin(n\pi/2) \tag{8.10}$$

This differential equation can be used as an alternative definition of the Chebyshev polynomials.

Implement one or more of the above numerical methods.

8.30 The Lotka-Volterra equations

A simple example of ordinary differential equations requiring numerical solution is the Lotka-Volterra system

$$\begin{aligned} \frac{dx}{dt} &= \alpha x - xy \\ \frac{dy}{dt} &= xy - y \end{aligned} \tag{8.11}$$

with α being a constant parameter. (Note that the independent variable is t and not x.) These equations are a simple model for the population dynamics of two species: a predator y and its prey x. In isolation, x would grow exponentially as $e^{\alpha t}$ and y would die out as e^{-t}. But when they are present together, individuals from x and y encounter each other at a rate proportional to xy, and y grows by feeding on x.

The Lotka-Volterra equations are usually written with four parameters, for example as

$$\begin{aligned} \frac{1}{x}\frac{dx}{dt} &= A - By \\ \frac{1}{y}\frac{dy}{dt} &= Cx - D \end{aligned} \tag{8.12}$$

But three of the parameters can be eliminated by rescaling

$$t \leftarrow Dt \qquad x \leftarrow (C/D)x \qquad y \leftarrow (B/D)y \tag{8.13}$$

Basically, we have chosen units so as to make the problem simpler.

There is another interesting way of writing the Lotka-Volterra equations. Defining

$$p = \ln x \quad q = \ln y \qquad H(p,q) = e^p + e^q - p - \alpha q \tag{8.14}$$

it is easy to verify that

$$\dot{p} = -\frac{\partial H}{\partial q} \qquad \dot{q} = \frac{\partial H}{\partial p} \tag{8.15}$$

are equivalent to the original equations (8.11). The equations (8.15) are known as Hamilton's equation while the function $H(p,q)$ is known as the Hamiltonian. This may seem a contrived way of writing, but it reveals interesting properties about the system. In particular, we have

$$\dot{H} = \frac{\partial H}{\partial p}\dot{p} + \frac{\partial H}{\partial q}\dot{q} + \frac{\partial H}{\partial t} = \frac{\partial H}{\partial t} \tag{8.16}$$

Hence H is a constant. In other words $x + y - \ln(xy^\alpha)$ is a constant.

Solve the Lotka-Volterra system.

8.31 The Lorenz equations

Our last example is not a Hamiltonian system, but we include it because it is the archetypical chaotic dynamical system. It consists of three nonlinear differential equations

$$\begin{aligned} \dot{X} &= \sigma(Y - X) \\ \dot{Y} &= rX - Y - XZ \\ \dot{Z} &= XY - bZ \end{aligned} \tag{8.17}$$

with (σ, r, b) being constant parameters. The equations originally appeared as a simple model of fluid flow[5] but were later found to describe other systems as well.

In the original context, X, Y, Z describe a simplified atmosphere model: X represents the amount of convection, Y the temperature difference between the rising and falling

[5] You can easily find Lorenz's paper from 1963 by searching for the title: *Deterministic Nonperiodic Flow*. As well as being a scientific classic, it is very readable.

currents, and Z is how much the temperature gradient differs from linear. The parameter σ is a scaled viscosity, r a scaled temperature, and b a geometric factor. The values

$$\sigma = 10 \qquad r = 28 \qquad b = 8/3 \tag{8.18}$$

produce especially interesting behaviour.

One can infer some properties of the solution by considering the linearized equations

$$\begin{pmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{pmatrix} = \begin{pmatrix} -\sigma & \sigma & 0 \\ r - Z & -1 & -X \\ Y & X & -b \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{8.19}$$

It follows that the divergence of a small volume (X, Y, Z) of phase space is

$$\frac{\partial \dot{X}}{\partial X} + \frac{\partial \dot{Y}}{\partial Y} + \frac{\partial \dot{Z}}{\partial Z} = -(\sigma + b + 1) \tag{8.20}$$

Because the divergence is always negative, the system is said to have an attractor. At the time, when a system had an attractor, people expected it would go to a fixed point or a periodic orbit. There is an obvious fixed point at the origin, but if $r > 0$ it is not a stable fixed point. For $r > 1$ there are two additional fixed points $X = Y = \pm\sqrt{b(r-1)}, Z = r - 1$, but if $\sigma > b + 1$ these also become unstable for large-enough r. For values such as (8.18) the system has a chaotic or *strange attractor*.

Write a program to show the Lorenz strange attractor in screensaver style. You can make a *Tkinter Canvas* fill the screen as follows. (A top bar may remain, depending on the operating system.)

```
screen = Tk()
wd, ht = screen.winfo_screenwidth(), screen.winfo_screenheight()
screen.geometry("%dx%d+0+0"%(wd, ht))
canv = Canvas(screen, height = ht, width = wd, background = "black")    (8.21)
canv.pack()
update()      # Draw a frame
screen.mainloop()
```

8.32 Chandrasekhar's limit

$$\begin{aligned} \frac{dM(r)}{dr} &= 4\pi r^2 \rho(r) \\ \frac{dP(r)}{dr} &= -\frac{M(r)\rho(r)}{r^2} \end{aligned} \tag{8.22}$$

$$n_e = \frac{1}{3\pi^2}\left(\frac{p_F}{\hbar}\right)^3 \qquad \rho = \mu n_e \tag{8.23}$$

$$P = \frac{1}{3\pi^2}\int_0^{p_F}\left(\frac{p_F}{\hbar}\right)^3 v(p)\,dp \qquad v = \frac{pc}{\sqrt{p^2 + m_e^2 c^2}} \tag{8.24}$$

This gives

$$\frac{dP}{dr} = n_e v(p_F)\,\frac{dp_F}{dr} \tag{8.25}$$

$$\begin{aligned}\frac{dM}{dr} &= 4\pi r^2 \mu n_e(p_F)\\ \frac{dp_F}{dr} &= -\frac{GM\mu}{r^2 v(p_F)}\end{aligned} \tag{8.26}$$

$$\begin{aligned} r &= \frac{\mathcal{L}\,\mathcal{M}^2}{m_e\mu} \times x\\ M &= \frac{\mathcal{M}^3}{\mu^2} \times Y\\ p_F &= m_e c \times Z\end{aligned} \tag{8.27}$$

$$\mathcal{M} = \left(\frac{\hbar c}{G}\right)^{1/2} \qquad \mathcal{L} = \left(\frac{\hbar^3}{cG}\right)^{1/2} \tag{8.28}$$

$$\begin{aligned}\frac{dY}{dx} &= \frac{4}{3\pi}\,x^2\,Z^3\\ \frac{dZ}{dx} &= -\frac{Y}{x^2}\,\left(1 + 1/Z^2\right)^{1/2}\end{aligned} \tag{8.29}$$

8.33 Vampires

To be written.

8.34 Pattern formation

To be written.

9. Hamiltonians

9.35 Driven pendulum

Many physical systems have their dynamics described by a Hamiltonian. In this and the next few sections we will explore some contrasting examples.

Consider the Hamiltonian

$$H(p,q,t) = \tfrac{1}{2}p^2 - \cos q - \epsilon \cos(q - \omega t) \tag{9.1}$$

The first two terms are well known: they describe a pendulum. The last term is not so familiar. In fact, it corresponds to putting the pendulum on a vertical wheel and spinning the wheel at a constant rate.

To derive the physical meaning of (9.1) let us consider a pendulum of length l in a gravitational field g. This is mounted on a wheel of radius b, spinning with angular speed ω. The coordinates of the pendulum bob, measured from the centre of the wheel, are as follows.

$$\begin{aligned} x &= b \sin \omega t + l \sin q \\ y &= -b \cos \omega t - l \cos q \end{aligned} \tag{9.2}$$

From the equation

$$\dot{q} = \frac{\partial H}{\partial p} = p \tag{9.3}$$

it is clear that p is the angular velocity of the pendulum. We need to show that

$$\dot{p} = -\frac{\partial H}{\partial q} = -\sin q - \epsilon \sin(q - \omega t) \tag{9.4}$$

is the angular acceleration.

There are two forces on the pendulum bob, from gravity and a fictitious force from the wheel. The acceleration is

$$(\ddot{x},\ddot{y}) = (0,-g) - (\ddot{x},\ddot{y})_{\text{wheel}} = (0,-g) - b\omega^2(-\sin \omega t, \cos \omega t) \tag{9.5}$$

The radial part of this acceleration is balanced by the pendulum tension, the tangential part equals $l\dot{p}$. Since the tangential direction with respect to the pendulum is $(\cos q, \sin q)$ we have

$$l\dot{p} = -gl - b\omega^2 \sin(q - \omega t) \tag{9.6}$$

If we choose

$$\frac{g}{l} = 1 \qquad \epsilon = \frac{b}{l}\omega^2 \tag{9.7}$$

we recover the desired form (9.4).

An alternative strategy is to proceed using the usual machinery of Lagrangian and Hamiltonian mechanics. The Lagrangian is

$$\begin{aligned} L(q,\dot{q},t) &= \tfrac{1}{2}\dot{x}^2 + \tfrac{1}{2}\dot{y}^2 - gy \\ &= \tfrac{1}{2}\omega^2 b^2 + \tfrac{1}{2}l^2\dot{q}^2 + gl\cos q + lb\omega\dot{q}\cos(q-\omega t) + gb\cos t \end{aligned} \tag{9.8}$$

The first term in the second line is constant and has no effect. The last term is of the form dS/dt and hence can also be dropped from a Lagrangian. Dividing by l^2 and substituting from (9.7) gives

$$L = \tfrac{1}{2}\dot{q}^2 + \cos q + (\epsilon/\omega)\,\dot{q}\cos(q-\omega t) \tag{9.9}$$

Now we apply

$$L \to L - \frac{d}{dt}\sin(q-\omega t) \tag{9.10}$$

which results in modifying the last term, leaving us with

$$L = \tfrac{1}{2}\dot{q}^2 + \cos q + \epsilon\cos(q-\omega t) \tag{9.11}$$

The Hamiltonian (9.1) then follows.

Another way of describing the system is to artificially go to higher dimensions

$$H(\mathbf{p},\mathbf{q}) = \tfrac{1}{2}p_1^2 + \omega p_2 - \cos q_1 - \epsilon\cos(q_1 - q_2) \tag{9.12}$$

Then Hamilton's equations

$$\dot{\mathbf{p}} = -\frac{\partial H}{\partial \mathbf{q}} \qquad \dot{\mathbf{q}} = \frac{\partial H}{\partial \mathbf{p}} \tag{9.13}$$

give q_2 equals ωt plus a constant, along with the original equations. There is nothing physically new here, but formally H has no functional dependence on t. Hence, by the argument following equation (8.16), $H(\mathbf{p},\mathbf{q})$ is a constant of motion.

The equations of motion can be integrated by any of the general purpose method. But there is a particularly simple integration method that exploits the special form of the Hamiltonian, namely that

$$H(\mathbf{p},\mathbf{q}) = H_A(\mathbf{p}) + H_B(\mathbf{q}) \tag{9.14}$$

Both H_A and H_B are exactly soluble in isolation. An integrator for the whole system proceeds as (i) evolve under H_A for half a time step, (ii) evolve under H_B for a full time step, (iii) (i) evolve under H_A for half a time step again.

$$\begin{aligned} &q1, q2 = (q1 + p1 * dt/2, q2 + omega * dt/2) \\ &s1, s2 = (sin(q1), eps * sin(q1 - q2)) \\ &p1, p2 = (p1 - (s1 + s2) * dt, p2 + s2 * dt) \\ &q1, q2 = (q1 + p1 * dt/2, q2 + omega * dt/2) \end{aligned} \tag{9.15}$$

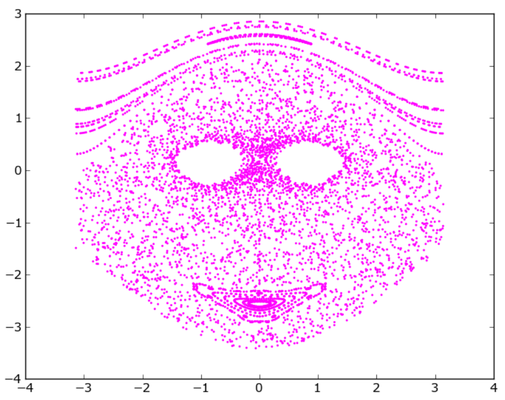

Figure 9.1: Surface of section (q,p) at $\omega t = n\pi$ for the Hamiltonian (9.1) with $\omega = -2, \epsilon = 0.3$.

This is known as leapfrog.

Another interesting quantity in this type of system is a surface of section. Figure 9.1 shows an example. This was plotted by integrating Hamilton's equations from $p = p_{\text{ini}}, q = 0, t = 0$ (for many different p_{ini}) whenever $\omega t = 2n\pi$.

Integrate the dynamical equations for a driven pendulum and try to find some interesting sorts of behaviour, such as resonances or chaos.

9.36 Cyclotrons

Particle accelerators use combinations of time-varying electric and magnetic fields to accelerate charged particle to high energies. In this section we will study a Hamiltonian system which is a simple model for a particle accelerator.

The Hamiltonian we will study is

$$H = \sqrt{1 + (p_x + y)^2 + (p_y - x)^2} - \alpha y \cos \omega t \tag{9.16}$$

With respect to Hamilton's equations (9.13) for motion in two dimensions, we identify x, y with $\mathbf{q}$ and p_x, p_y with $\mathbf{p}$.

The Hamiltonian (9.16) represents a relativistic charged particle in a constant magnetic field along B_z and an alternating electric field along y. Let us see why.

In special relativity, the Hamiltonian for a particle of mass m is

$$\sqrt{(mc^2)^2 + p^2c^2} \tag{9.17}$$

Differentiating to get Hamilton's equation for $\dot{\mathbf{q}}$ (which we write as $\mathbf{v}$), and then rearranging, we get

$$\mathbf{p} = \frac{m\mathbf{v}}{\sqrt{1 - v^2/c^2}} \tag{9.18}$$

which is the usual expression for relativistic momentum. The non-relativistic limit is $p \ll mc$, in which case the Hamiltonian reduces to

$$mc^2 + \frac{p^2}{2m} \tag{9.19}$$

giving the Newtonian equations of motion.

If the particle has charge e and is in an electric field $\mathbf{E}$ and magnetic field $\mathbf{B}$, both of which may depend on position and on time time, the Hamiltonian is modified in two ways. For the electric field, one simply adds the electric potential energy $e\Phi$, where $\mathbf{E} = -\nabla\Phi$. For the magnetic field, we need the vector potential $\mathbf{A}$ where $\mathbf{B} = \nabla \times \mathbf{A}$. Then $\mathbf{p}$ in the Hamiltonian is replaced by $\mathbf{p} - e\mathbf{A}$.

For an alternating electric field along y, we have

$$\Phi = -E_0 y \cos\omega t \quad \to \quad \mathbf{E} = (0, E_0 \cos\omega t, 0) \tag{9.20}$$

For a constant magnetic field B_z, we have

$$\mathbf{A} = B_z\left(-\tfrac{1}{2}y, \tfrac{1}{2}x, 0\right) \quad \to \quad \mathbf{B} = (0, 0, B_z) \tag{9.21}$$

The full Hamiltonian is therefore

$$H = \sqrt{(mc^2)^2 + c^2\left(p_x + \tfrac{1}{2}eB_z y\right)^2 + c^2\left(p_y - \tfrac{1}{2}eB_z x\right)^2} - eE_0 y \cos\omega t \tag{9.22}$$

We have three constant parameters with dimensions as follows.

$$\begin{array}{ll} mc^2 & ML^2T^{-2} \\ \frac{1}{2}eB_z & MT^{-1} \\ eE_0 & MLT^{-2} \end{array} \tag{9.23}$$

If we choose our units of mass, length and time, respectively as

$$m \qquad \frac{2mc}{eB_z} \qquad \frac{2m}{eB_z} \tag{9.24}$$

we can set the first two constants to unity. Our implied unit of force is $\frac{1}{2}ceB_z$ and since the last constant is a force, we can introduce

$$\alpha = \frac{2E_0}{cB_z} \tag{9.25}$$

and bring the Hamiltonian into the form (9.16).

In the non-relativistic limit, we have

$$H = mc^2 + \frac{\left(p_x + \frac{1}{2}eB_z y\right)^2 + \left(p_y - \frac{1}{2}eB_z x\right)^2}{2m} - eE_0 y \cos\omega t \tag{9.26}$$

We can now drop the constant mc^2 term and make a still more drastic choice of units of mass, length, and time

$$m \qquad \left(\frac{2m}{eB_z}\right)^2 \frac{eE_0}{m} \qquad \frac{2m}{eB_z} \tag{9.27}$$

we get

$$H = \tfrac{1}{2}(p_x + y)^2 + \tfrac{1}{2}(p_y - x)^2 - y\cos\omega t \tag{9.28}$$

The non-relativistic Hamiltonian (9.28) is in fact exactly soluble. Writing down Hamilton's equations and eliminating p_x, p_y gives

$$\ddot{x} = 2\dot{y} \qquad \ddot{y} = -2\dot{x} + \cos\omega t \tag{9.29}$$

We can recognize the forces due to a constant magnetic field along z and an alternating electric field along y. The solution is

$$x(t) = a\sin(2t + k) + \frac{2\sin(\omega t)}{\omega(4-\omega^2)} \qquad y(t) = \tfrac{1}{2}\frac{d}{dt}x(t) \tag{9.30}$$

where a and k are integration constants. (Two further integration constants can be added to x and y.) Two limiting cases are particularly interesting.

- For $\omega =\rightarrow 0$ the particle moved on a cycloid (if starting at rest, in general a trochoid). The curious feature is, that on average, it moved sideways to the electric field.
- For $\omega =\rightarrow 2$ the spirals outwards. This is the principle of a cyclotron.

Compute and plot a few particle trajectories in (9.28) and (9.16). There are some interesting differences. First, for the non-relativistic case, changing α simply rescales the trajectory, but in relativity α is not a simply scaling. Second, for $\omega = 2$, the particle can be accelerated to arbitrarily high speeds, but in relativity that does not happen.

9.37 Three-body problem

Another interpretation of the magnetic force (9.29) is the Coriolis force in a frame rotating with unit angular velocity. If we now add $\frac{1}{2}(x^2+y^2)$ to the Hamiltonian (9.28), that amounts to adding a centrifugal force. The resulting Hamiltonian

$$H = \tfrac{1}{2}p_x^2 + \tfrac{1}{2}p_y^2 + yp_x - xp_y \tag{9.31}$$

corresponds to a particle in a rotating frame. If we now add a potential $V(x,y)$

$$H = \tfrac{1}{2}p_x^2 + \tfrac{1}{2}p_y^2 + yp_x - xp_y + V(x,y) \tag{9.32}$$

we have the Hamiltonian of a particle in a rotating potential. An important example has

$$V = -\frac{1-\mu}{\sqrt{(x+\mu)^2 + y^2}} - \frac{\mu}{\sqrt{(x-1+\mu)^2 + y^2}} \tag{9.33}$$

The potential represents two gravitating bodies, with masses in the ratio $(\mu, 1-\mu)$ in a circular orbit (the origin is chosen as the centre of mass). The Hamiltonian (9.32) describes a test particle in the combined gravitational field.

The Hamiltonian (9.32) can also be derived more formally, using the machinery of Hamiltonian dynamics. To do this, we recall that Hamilton's equations derive from a variational principle

$$\delta \int \mathbf{p} \cdot d\mathbf{q} - H(\mathbf{p}, \mathbf{q}, t)\, dt = 0 \tag{9.34}$$

with $\mathbf{q}, t$ fixed at the ends. Consider a new set of variables $\mathbf{P}, \mathbf{Q}$ and let the Hamiltonian in terms of these be $H(\mathbf{P}, \mathbf{Q}, t)$. [Note that $H(\mathbf{p}, \mathbf{q}, t)$ and $H(\mathbf{P}, \mathbf{Q}, t)$ will have different functional forms. It simplifies notation if we use the same symbol H for both functions, letting the argument list specify which function we mean.] In these variables, we can write down a variational principle

$$\delta \int \mathbf{P} \cdot d\mathbf{Q} - H(\mathbf{P}, \mathbf{Q}, t)\, dt = 0 \tag{9.35}$$

The condition for both variational equations to represent the same dynamics is that their integrands differ by an exact differential. Let us write this exact differential as $d(\mathbf{p} \cdot \mathbf{q}) - dS(\mathbf{p}, \mathbf{Q}, t)$.

$$\mathbf{p} \cdot d\mathbf{q} - H(\mathbf{p}, \mathbf{q}, t)\, dt = \mathbf{P} \cdot d\mathbf{Q} - H(\mathbf{P}, \mathbf{Q}, t)\, dt + d(\mathbf{q} \cdot \mathbf{p}) - dS(\mathbf{p}, \mathbf{Q}, t) \tag{9.36}$$

where $S(\mathbf{p}, \mathbf{Q}, t)$ is an arbitrary function. Expanding this out and comparing coefficients we have

$$\mathbf{q} = \frac{\partial S}{\partial \mathbf{p}} \qquad \mathbf{P} = \frac{\partial S}{\partial \mathbf{Q}} \qquad H(\mathbf{p}, \mathbf{q}, t) = H(\mathbf{P}, \mathbf{Q}, t) + \frac{\partial S}{\partial t} \tag{9.37}$$

which gives $(\mathbf{p}, \mathbf{q}) \to (\mathbf{P}, \mathbf{Q})$ implicitly. This is known as a canonical transformation and $S(\mathbf{p}, \mathbf{Q}, t)$ is called the generating function.

Now let us identify $\mathbf{P}, \mathbf{Q}$ with some cartesian variables X, Y, P_X, P_Y. In particular, let us assume that

$$H = \tfrac{1}{2}(P_X^2 + P_Y^2) + V(X, Y, t) \tag{9.38}$$

generates the Newtonian equations of motion in a time-dependent potential. Let $\mathbf{p}, \mathbf{q}$ with some other variables p_x, p_y, x, y and let us choose

$$S = \begin{pmatrix} p_x \\ p_y \end{pmatrix} \mathbf{M} \begin{pmatrix} X \\ Y \end{pmatrix} \tag{9.39}$$

where

$$\mathbf{M} \equiv \begin{pmatrix} \cos t & -\sin t \\ \sin t & \cos t \end{pmatrix} \tag{9.40}$$

The transformation generated is

$$\begin{pmatrix} x \\ y \end{pmatrix} = \mathbf{M} \begin{pmatrix} X \\ Y \end{pmatrix} \qquad \begin{pmatrix} p_x \\ p_y \end{pmatrix} = \mathbf{M} \begin{pmatrix} P_x \\ P_y \end{pmatrix} \tag{9.41}$$

from which it is clear that x, y are rotating with respect to X, Y at unit angular velocity. The Hamiltonians are related by

$$H(p_x, p_y, x, y) = H(P_X, P_Y, X, Y) + \begin{pmatrix} p_x & p_y \end{pmatrix} \frac{\partial \mathbf{M}}{\partial t} \mathbf{M}^{-1} \begin{pmatrix} x \\ y \end{pmatrix} \qquad (9.42)$$

to the Hamiltonian. The last expression simplifies to $yp_x - xp_y$.

Numerically integrate the Hamilton's equations for (9.32) and plot the resulting orbits for some values of μ and the initial conditions. For example, $\mu = 0.2$ with initial $(x, y, p_x, p_y) = (0, 0, 0, 1)$ gives an orbit around the larger mass. Try to find some orbits around the smaller mass, and some chaotic orbits.

9.38 Near a black hole

The trajectory of a particle near a black hole is given by the Hamiltonian.

$$2H = (1 - 2/r)\, p_r^2 + \frac{p_\phi^2}{r^2} - (1 - 2/r)^{-1} \qquad (9.43)$$

Distances are in units of GM/c^2, the so-called gravitational radius of the back hole.

We can transform to a more convenient form for integration as follows. Referring back to the canonical transformation (9.37), let us identify $(\mathbf{p}, \mathbf{q})$ with (p_x, p_y, x, y) and identify $(\mathbf{P}, \mathbf{Q})$ with (r, ϕ, p_r, p_ϕ). Choosing

$$S = (r \cos \phi)\, p_x + (r \sin \phi)\, p_y \qquad (9.44)$$

gives

$$x = r \cos \phi\,, \qquad y = r \sin \phi\,, \qquad (9.45)$$

as desired for polar coordinates. The momentum components become

$$p_r = \frac{xp_x + yp_y}{r}\,, \qquad p_\phi = xp_y - yp_x\,. \qquad (9.46)$$

and transforms the Hamiltonian to

$$H = \tfrac{1}{2} p_x^2 + \tfrac{1}{2} p_y^2 - \tfrac{1}{2} (1 - 2/r)^{-1} - \frac{(xp_x + yp_y)^2}{r^3} \qquad (9.47)$$

Note that now r is to be considered a function of x, y.

Numerically integrate Hamilton's equations for the Hamiltonian (9.47) and plot some orbits, starting from a few different initial conditions. For example, if $(x, y, p_x, p_y) = (25, 0, 0, 0.2)$ initially, the result is very similar to a Newtonian circular orbit.

9.39 Vortices

This example concerns a curious analogue of a gravitational N-body problem, arising from fluid mechanics.

Let $\mathbf{v}(\mathbf{r})$ be the velocity field of a fluid. If the fluid is incompressible, $\nabla \cdot \mathbf{v} = 0$ and we can introduce the so called *stream function,* a vector field ψ such that

$$\mathbf{v} = \nabla \times \psi, \qquad \nabla \cdot \psi = 0 \tag{9.48}$$

The stream function satisfies

$$\nabla^2 \psi = -\omega \tag{9.49}$$

where $\omega \equiv \nabla \times \mathbf{v}$ and is called the *vorticity*. In the case of flow without viscosity ("dry water") vorticity is conserved, and the flow satisfies

$$\frac{d\omega}{dt} = \omega \cdot \nabla \mathbf{v} \tag{9.50}$$

There is an analogy with magnetic fields: $\mathbf{v}$ is like a magnetic field, ψ is like the vector potential, and ω is like current.

For two-dimensional flow, some interesting things can happen. First, suppose ω is zero except at some discrete *vortex* points, $z_1, \ldots, z_N$. Then the stream function will have the solution

$$\psi(z) = -\sum_n \ln|z - z_n| \tag{9.51}$$

There is an arbitrary coefficient with each vortex term, but we take all the vorticities equal here. Also, ω is naturally normal to the flow and hence

$$\frac{d\omega}{dt} = 0 \tag{9.52}$$

which is to say, ω moves with the fluid. Hence the vortices move with $\mathbf{v}$, or

$$\dot{z}_m = \nabla \times \psi(z_m) \tag{9.53}$$

Now, if we identify z_n with (q_n, p_n), we can write Hamilton's equations as

$$\dot{z}_m = \nabla_m \times H(z_n, \bar{z}_n) \tag{9.54}$$

Hence we have an effective Hamiltonian for the N-vortex problem:

$$H = -\sum_{m \neq n} \ln|z_m - z_n| \tag{9.55}$$

We can thus interpret the (x, y) of each vortex as a phase-space location (q, p) and evolve these under the Hamiltonian. The vortices behave like interacting particles, even though they have no kinetic energy and it's really the stream function that makes them move. There are three integrals of motion

$$H \qquad \sum_n z_n \qquad \sum_n |z_n|^2 \tag{9.56}$$

which are like energy, momentum, and angular momentum. As a result, the three-vortex problem is integrable, for four or more we will find chaos.

10. Perspective

10.40 Interactive graphics: triangle-interior test

Given four complex numbers a, b, c and p, does p lie inside the triangle joining a, b, c? This problem, versions of which are pervasive in computer visualization, is a good place to introduce interactive graphics.

There are several libraries for doing interactive graphics with Python. The simplest one is *Tkinter*. A simple program to respond to mouse clicks and mouse dragging looks like this.

```
from Tkinter import Tk, Canvas, ALL
# code for clicked() and moved()
root = Tk()
canv = Canvas(root, width = 300, height = 300)
canv.bind("<Button-1>", clicked)
canv.bind("<B1-Motion>", moved)
canv.pack()
root.mainloop()
```
(10.1)

The above code binds mouse events to functions which must be defined. The following example lets the user draw a little pendulum on the canvas with the mouse.

```
def clicked(event):
    global xo, yo
    xo = event.x
    yo = event.y
def moved(event):
    canv.delete(ALL)
    x = event.x
    y = event.y
    canv.create_line(xo, yo, x, y)
    canv.create_oval(x - 4, y - 4, x + 4, y + 4)
```
(10.2)

As we see, *Tkinter.canv* has a coordinate system defined. The origin is that the upper left and the units are pixels.

The *Tkinter* library also allows animation. Here is some code to make a ball bounce around in a window.

```
from Tkinter import Tk, Canvas, ALL
# Initialize ht, wd, rad, x, y, vx, vy
def update():
    global x, y, vx, vy
    canv.delete(ALL)
    d = canv.create_oval(x − rad, y − rad, x + rad, y + rad, fill = "blue")
    canv.update()
    # Reverse vx,vy if necessary
    x += vx
    y += vy
    canv.after(40, update)        # wait 40 msec
window = Tk()
canv = Canvas(window, height = ht, width = wd)
canv.pack()
update()
window.mainloop()
```
(10.3)

We will not attempt to combine user-interaction and animation.

Let us now return to the triangle-interior problem at the start of this section. There are several ways of solving the problem, and here is one. Let us take a straight line from the midpoint $(b+c)/2$ to p and keep going. If p is inside the triangle, we will intersect one of the sides a-to-b or a-to-c. If p is outside the triangle, we will need to go backwards to intersect, or we will intersect outside the triangle.

Writing down expressions for the lines, it is not difficult to see that the condition for intersecting the side a-to$b$ is $\gamma > 0$ and $0 < \beta < 1$ in

$$p + \gamma(p - (b+c)/2) = a + \beta(b - a) \tag{10.4}$$

For the side a-to-c an analogous pair of conditions applies. If either pair of conditions holds, p is in the triangle, otherwise it is outside.

Write a GUI implementing the following on a *Tkinter canv*.

(i) A triangle is drawn, which can be modified by dragging the mouse to shift a chosen corner.

(ii) Clicking anywhere on the canvas causes a dot to appear at that point. The dot should be green if inside the triangle and red if outside.

10.41 Flight paths on the globe

There are numerous ways of projecting the globe to two dimensions. The Hammer-Aitoff projection is a particularly important one, because it preserves areas with comparatively little shape-distortion. The transformation is

$$x = \frac{2\sqrt{2}\sin\theta\sin\frac{1}{2}\phi}{\sqrt{1+\sin\theta\cos\frac{1}{2}\phi}} \qquad y = \frac{2\cos\theta}{\sqrt{1+\sin\theta\cos\frac{1}{2}\phi}} \tag{10.5}$$

and the inverse is defined with the help of an auxiliary variable u as

$$\begin{aligned} u &= \sqrt{1 - x^2/16 - y^2/4} \\ \theta &= \arccos(uy) \\ \phi &= 2\arctan(ux/(4u^2 - 2)) \end{aligned} \tag{10.6}$$

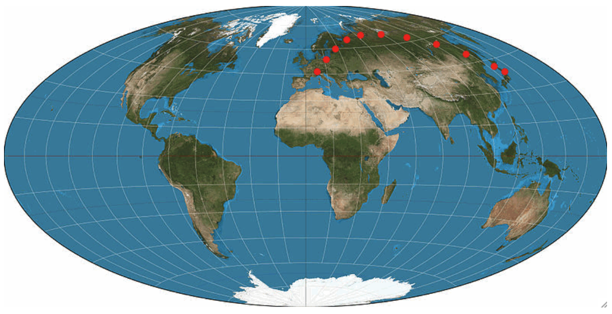

Figure 10.1: Zurich and Sapporo are at similar latitudes, but the shortest route between them goes far to the North. Consecutive dots are 1000 km apart, except for the last pair, which are closer.

Figure 10.1 shows a world map in Hammer-Aitoff projection, and as we can see, the shortest between two points can take unexpected shapes. We want to find the shortest route between two points. An elegant way of doing so involves going to four dimensions.

Complex numbers have a non-commmutative but useful generalization to four dimensions, known as quaternions. There are many ways of representing quaternions, and one is to use Pauli matrices.

$$\sigma_1 = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad \sigma_2 = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \qquad \sigma_3 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{10.7}$$

The Pauli matrices behave as generalizations of $\sqrt{-1}$. A general quaternion has the form

$$\mathbf{Q} = \alpha + i(x\sigma_1 + y\sigma_2 + z\sigma_3) \tag{10.8}$$

where the components α, x, y, z are all real. The norm is defined as

$$|\mathbf{Q}|^2 = \alpha^2 + x^2 + y^2 + z^2 \tag{10.9}$$

Once the matrices $\sigma_1, \sigma_2, \sigma_2$ have been initialized, we can forget the details in (10.7). Everything can be done with matrix operations. In particular, the various components can be extracted by

$$\alpha = \tfrac{1}{2}\,\mathrm{Tr}(\mathbf{Q}) \qquad x = \mathrm{Tr}(\sigma_1\mathbf{Q})/(2i) \tag{10.10}$$

and similarly for y and z. Quaternions can be multiplied by matrix multiplication, and the norm of a product is the product of the norms. We can also express $\mathbf{Q}$ in a kind of polar form

$$r(\cos\theta + i\sin\theta\,\mathbf{n}) \tag{10.11}$$

where r is real and $\mathbf{n}$ is a traceless unit quaternion. This form lets us take arbitrary powers.

Clearly, any three-dimensional vector can be represented by a traceless quaternion. If we have two such quaternions $\mathbf{P}$ and $\mathbf{Q}$, the product $\mathbf{PQ}$ encodes both the dot and cross products: the dot product is $-\frac{1}{2}\,\mathrm{Tr}(\mathbf{PQ})$, and the traceless part of $\mathbf{PQ}$ represents the cross product. So everything in three-dimensional vector algebra can be done with quaternions, often more concisely.

Quaternions also have two other properties, which make them very important in visualization.

The first of these is spherical interpolation, known as slerp. Let $\mathbf{P}$ and $\mathbf{Q}$ be two traceless unit quaternions (that is, points on a unit sphere). Now consider

$$\mathbf{P}\left(\mathbf{P}^{-1}\mathbf{Q}\right)^{s} = \mathbf{P}\left(-\mathbf{PQ}\right)^{s} \tag{10.12}$$

with s varying from 0 to 1. This interpolates from $\mathbf{P}$ to $\mathbf{Q}$ while remaining on a sphere.

The second property is that rotations become very simple. Let $\mathbf{P}$ be a traceless quaternion. Then

$$(\cos\theta + i\sin\theta\,\mathbf{n})\,\mathbf{Q}\,(\cos\theta - i\sin\theta\,\mathbf{n}) \tag{10.13}$$

rotates $\mathbf{Q}$ by an angle 2θ around the axis defined by $\mathbf{n}$.

10.42 Dancing polyhedron

In this example we show a cube or other polyhedron in perspective, spinning about an arbitrary axis.

The basic formula for perspective is very simple. We put the observer at the origin and the screen at distance d in the z direction. The screen coordinates (X, Y) of a spatial point (x, y) are then

$$X = xd/z \qquad Y = yd/z \tag{10.14}$$

If we want X, Y in pixels, then d must be in pixels.

Write an animation showing a spinning wire-frame cube or tetrahedron in perspective. The perspective is visually more effective if you draw covered lines (that is, lines with a face in front of them) differently. To test if an edge of a regular polyhedron is covered, it is enough to do the test for the midpoint of the edge. Try and reduce the problem of testing whether any given point is covered to the triangle-interior test from section 10.40.