Chapter 17

Ordinary Differential Equations I

17.1 Newton's law of cooling

Newton's law of cooling says that the rate of change of temperature of a body is proportional to the temperature difference between the body and its surroundings. Let $T(t)$ denote the temperature of the body as a function of time t. Newton's law of cooling is

$$\frac{dT(t)}{dt} = -k[T(t) - T_a] \,, \tag{17.1}$$

where T_a is the ambient temperature, the temperature of the surroundings. The left–hand side of this equation is the time rate of change of the body's temperature. The right hand side is the temperature difference, $T(t) - T_a$, multiplied by a proportionality constant $-k$ with $k > 0$. If $T(t) > T_a$, then $dT(t)/dt$ is negative and $T(t)$ decreases in time. If $T(t) < T_a$, then $dT(t)/dt$ is positive and $T(t)$ increases in time. The particular value for k depends on the size, shape and composition of the body, as well as the nature of the surroundings.

Equation (17.1) is an ordinary differential equation (ODE) for the unknown function $T(t)$. Most differential equations cannot be solved analytically, but Newton's law of cooling is an exception. The exact, analytical solution is found by separation of variables:

$$\frac{dT}{T - T_a} = -k\, dt \,. \tag{17.2}$$

Integrating both sides, we have

$$\ln(T - T_a) = -k\, t + c \tag{17.3}$$

where c is an integration constant. Solving for T gives

$$T(t) = T_a + e^c e^{-kt} \,. \tag{17.4}$$

Let $T_0 = T(0)$ denote the temperature of the body at the initial time $t = 0$. Then Eq. (17.4) evaluated at $t = 0$ implies $e^c = T_0 - T_a$. This yields

$$T(t) = T_a + (T_0 - T_a)e^{-kt} \tag{17.5}$$

for the temperature of the body as a function of time.

The "half–life" t_H for the cooling (or heating) process is the time it takes the body to reach a temperature that is half–way between the initial and ambient temperatures. That is, when $t = t_H$, the temperature is $T = (T_0 + T_a)/2$. Inserting these values into the solution (17.5), we find

$$k = \frac{1}{t_H} \ln 2 \ , \tag{17.6}$$

where ln is the natural logarithm.

Exercise 17.1

Plot a graph of $T(t)$ versus t using reasonable values for the initial and ambient temperatures T_0 and T_a, and the half–life t_H.

17.2 Euler's method

Let's solve the differential equation (17.1) numerically. Our goal will be to find the temperature of the body at discrete times t_0, t_1, t_2, *etc.* up to some final time t_N. Let the index i range from 0 through N, so the discrete times are t_i. The time difference from one discrete time to the next is $\Delta t = t_{i+1} - t_i$, or $\Delta t = (t_N - t_0)/N$. See Fig. 17.1.

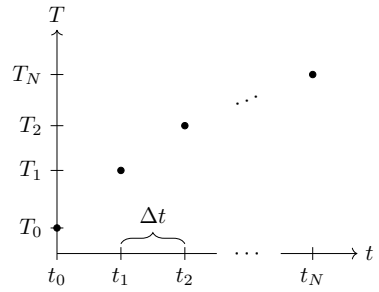

Fig. 17.1: Values of temperature $T_i = T(t_i)$ at discrete times t_i. The figure shows N timesteps, each of length Δt.

At each of the times t_i the rate of change of temperature should equal $-k[T(t_i) - T_a]$. Recall from Ch. 16 on numerical differentiation that the derivative at time t_i can be approximated by

$$\dot{T}(t_i) \approx \frac{T(t_i + \Delta t) - T(t_i)}{\Delta t} \ , \tag{17.7}$$

where the "dot" denotes d/dt. This is the two point forward difference formula (16.4) with some changes of notation: T in place of f, t_i in place of x_0, Δt in place of h and "dot" in place of "prime". We can use this approximation to replace the derivative in Newton's law of cooling, with the result

$$\frac{T(t_i + \Delta t) - T(t_i)}{\Delta t} = -k[T(t_i) - T_a] \ . \tag{17.8}$$

Now simplify the notation by letting $T_i = T(t_i)$. Since $t_{i+1} = t_i + \Delta t$, then $T_{i+1} = T(t_{i+1}) = T(t_i + \Delta t)$ and Eq. (17.8) becomes

$$T_{i+1} = T_i - k(T_i - T_a)\Delta t \ . \tag{17.9}$$

The temperature at each discrete time t_i is obtained by iterating this equation. Each iteration is called a *timestep*. Given an initial temperature T_0, we use Eq. (17.9) with $i = 0$ to obtain T_1. This is the first timestep. Now that we know T_1, we use Eq. (17.9) with $i = 1$ to obtain T_2. This is the second timestep. We continue in this fashion for N timesteps.

This step–by–step procedure for solving the differential equation (17.1) is called *Euler's method*. Here is an example code using $T_a = 180°\text{C}$, $T_0 = 20°\text{C}$, $t_H = 1000\,\text{s}$, and a final time of $3600\,\text{s}$:

```
import numpy as np

tH = 1000.0        # half--life in seconds (s)
Ta = 180.0         # ambient temperature in Celcius (C)
Tinitial = 20.0    # initial temperature (C)
tinitial = 0.0     # initial time (s)
tfinal = 3600.0    # final time (s)
N = 20             # number of timesteps

Deltat = (tfinal - tinitial)/N  # compute Delta t
k = np.log(2)/tH                # compute k

# Create arrays to hold times and temperatures
t = np.linspace(tinitial,tfinal,N+1)
T = np.zeros(len(t))

# Solve using Euler's method
```

```
T[0] = Tinitial          # initial temperature
for i in range(N):       # loop for N timesteps
    T[i+1] = T[i] - k*(T[i] - Ta)*Deltat

print(T)                 # print temperature array
```

Make sure you understand how this code works. Why does `N+1` appear in the `linspace` command if there are only `N` timesteps?

Exercise 17.2

Use Euler's method to solve the differential equation (17.1) for Newton's law of cooling. Use the parameters listed above. Experiment with different values of N and compare the numerical answers for the final temperature to the analytical result $T(3600) = 166.80°\text{C}$. The numerical answers should improve as the number of timesteps is increased. Make a plot of temperature versus time showing both the numerical data and the analytical result for $T(t)$.

17.3 Truncation error

Euler's method is subject to truncation errors because it uses Eq. (17.9) iteratively to approximate the solution of the differential equation (17.1). The truncation errors come from the use of the two–point forward difference formula (17.7) in place of the time derivative $\dot{T}$. The two–point forward difference formula is a truncated version of the exact relation

$$\dot{T}(t_i) = \frac{T_{i+1} - T_i}{\Delta t} - \frac{1}{2}\ddot{T}(t_i)\Delta t + \cdots \ . \tag{17.10}$$

This is just Eq. (16.3) with, once again, some changes in notation. It is obtained by expanding $T(t_i + \Delta t)$ in a Taylor series about $T(t_i)$.

Let's rederive the iterative formula (17.9), this time keeping the extra terms from Eq. (17.10). Replace the time derivative in Newton's law of cooling with the right–hand side of Eq. (17.10), then rearrange:

$$T_{i+1} = T_i - k(T_i - T_a)\Delta t + \frac{1}{2}\ddot{T}(t_i)\Delta t^2 + \cdots \ . \tag{17.11}$$

The $\cdots$ terms are proportional to Δt^3 and higher powers of Δt.

This result shows that the error for a single timestep of the Euler method is

$$\text{error for one timestep} = \frac{1}{2}\ddot{T}(t_i)\Delta t^2 + \cdots \ . \tag{17.12}$$

Assuming Δt is small we can drop the $\cdots$ terms. Then the total error for N timesteps is

$$\text{Euler's method error} \approx \sum_{i=0}^{N-1} \frac{1}{2}\ddot{T}(t_i)\Delta t^2 \ . \tag{17.13}$$

Apart from a factor of $\Delta t/2$, we can recognize this expression as the left–endpoint rule approximation to the integral of $\ddot{T}$. That is,

$$\text{Euler's method error} \approx \frac{\Delta t}{2}\int_{t_0}^{t_N} \ddot{T}\, dt \ . \tag{17.14}$$

The integral equals $\dot{T}(t_N) - \dot{T}(t_0)$. This fact is not particularly useful because (in general) we don't know the values of $\dot{T}(t_N)$ or $\dot{T}(t_0)$. Nevertheless, these results show that the error for Euler's method is proportional to Δt. In turn, Δt is proportional to N^{-1}, where N is the number of timesteps. Therefore the error in the final value of T is

$$\text{Euler's method error} \propto N^{-1} \ . \tag{17.15}$$

You can perform a convergence test of your Euler's method code by finding the error in the final value of temperature as a function of resolution N.

Exercise 17.3a

Use Euler's method to solve Newton's law of cooling (17.1) with the parameter values of Sec. 17.2. Compute the error in the temperature at the final time $t = 3600\,\text{s}$ for various values of N, and plot $\log|\text{error}|$ versus $\log(N)$. Find the slope from the plot and verify that the error is proportional to N^{-1}.

Clearly, there is nothing special about 3600 s. We could choose a final time of, say, 1800 s and the conclusion would be the same: The error at any time is proportional to N^{-1}.

We can analyze the error at 1800 s even if the final time used in the simulation is 3600 s.

Exercise 17.3b

Repeat the analysis of the previous exercise. Keep the final run time at 3600 s, but compute the error at 1800 s. Plot $\log|\text{error}|$ versus $\log(N)$. Should you use the total number of timesteps N needed to reach 3600 s, or half that number? Show that it doesn't

matter. Either way, you should find that the error is proportional to N^{-1}.

17.4 Error estimation

Previously we assumed that the ambient temperature T_a in Newton's law of cooling is a constant. We can modify the problem by letting the ambient temperature vary in time. For example, let

$$T_a(t) = a + b \arctan(t/c) \ , \tag{17.16}$$

where a, b and c are constants and arctan is the inverse tangent function. Newton's law of cooling becomes

$$\dot{T}(t) = -k[T(t) - a - b \arctan(t/c)] \ . \tag{17.17}$$

This differential equation can't be solved analytically, but we can still solve it numerically. Using Euler's method the differential equation becomes

$$T_{i+1} = T_i - k[T_i - a - b \arctan(t_i/c)]\Delta t \ . \tag{17.18}$$

To be concrete, let the constants have values $a = 20°\text{C}$, $b = 100°\text{C}$, and $c = 1000\,\text{s}$. Also choose $T_0 = 60°\text{C}$ for the initial temperature and $k = 0.002/\text{s}$. What is the body's temperature at time $t = 3000\,\text{s}$?

Figure 17.2 shows a plot of the temperatures T_i obtained from Euler's method with $N = 10$. The plot also shows the ambient temperature $T_a(t)$ as a function of time.

Let's take a close look at the temperature at the final time, $t_N = 3000\,\text{s}$. With Euler's method and $N = 10$ timesteps, we find that the final temperature is $T(3000) = 138.384084°\text{C}$. How accurate is this result? If we only run our code with $N = 10$ timesteps, there is no way to tell how accurate the answer is. Since the truncation errors are proportional to N^{-1}, we can improve the result by increasing N. If we run the code with $N = 20$, the numerical answer changes from $138.384084°\text{C}$ to $137.850497°\text{C}$. This tells us that the correct answer is around $137°\text{C}$ or $138°\text{C}$. To be sure, we should run the code at even higher resolution.

Table 17.1 shows the final temperatures for increasing values of N. By $N = 80$ we see that the correct answer to three significant figures is $137°\text{C}$ and to four significant figures the answer is approximately $137.4°\text{C}$. To be confident in the answer to four significant figures, we need to increase the resolution even further. By $N = 1280$, it is apparent that the final

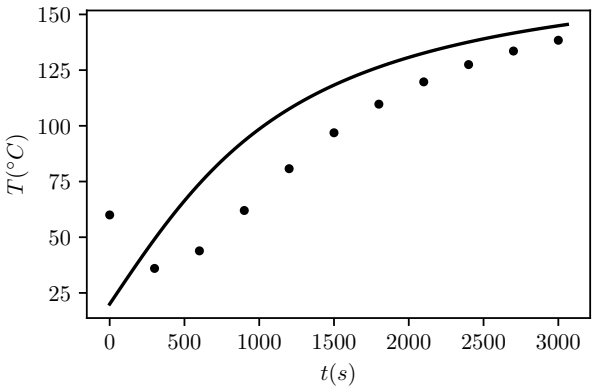

Fig. 17.2: Newton's law of cooling for a body with initial temperature $T_0 = 60°\mathrm{C}$ and ambient temperature $T_a(t) = 20°\mathrm{C} + 100°\mathrm{C}\arctan(t/1000\,\mathrm{s})$. The dots are obtained from Euler's method with $N = 10$ timesteps. The solid curve is the ambient temperature.

N	final temp (°C)
10	138.384084
20	137.850497
40	137.588677
80	137.461121
160	137.398355
320	137.367243
640	137.351756
1280	137.344030
2560	137.340172

Table 17.1: Temperatures at the final time $t_N = 3000\,\mathrm{s}$ for a body with initial temperature $T_0 = 60°\mathrm{C}$, in an environment with ambient termperature given by Eq. (17.16).

temperature is 137.3°C. By $N = 2560$, the answer has settled down to 137.34°C, giving us five significant figures.

Remember: You can't tell how accurate your numerical answers are

until you run your code at multiple resolutions.

Exercise 17.4a

Use Euler's method to solve for the temperature of a body when the ambient temperature varies sinusoidally:

$$T_a(t) = a + b\sin(t/c) \ .$$

Choose $a = 100°\text{C}$, $b = 40°\text{C}$, and $c = 1000\,\text{s}$. Also let $T_0 = 140°\text{C}$ and $k = 0.001$. Find the temperature of the body at time $16000\,\text{s}$, accurate to 4 significant figures.

Exercise 17.4b

As in the previous exercise, let the ambient temperature vary sinusoidally with $a = 100°\text{C}$, $b = 40°\text{C}$, $c = 1000\,\text{s}$, and $T_0 = 140°\text{C}$. Experiment with different values of k. Plot a graph showing both T and T_a as functions of time. As time passes, the body temperature T should settle down and execute regular oscillations. How does k affect the oscillation amplitude for T? How does k affect the phase difference between T and T_a? Interpret these results physically.

17.5 Three–point convergence test

Continue to focus on the final temperatures listed in Table 17.1. Let $T_{(N)}$ denote the temperature at the final time obtained from Euler's method with N timesteps. That is, $T_{(10)} = 138.384084°\text{C}$, $T_{(20)} = 137.850497°\text{C}$, *etc.* The differences between successive temperature values are

$$\begin{aligned}
T_{(10)} - T_{(20)} &= 0.533587°\text{C} \ , \\
T_{(20)} - T_{(40)} &= 0.261820°\text{C} \ , \\
T_{(40)} - T_{(80)} &= 0.127556°\text{C} \ , \\
T_{(80)} - T_{(160)} &= 0.062766°\text{C} \ , \\
T_{(160)} - T_{(320)} &= 0.031112°\text{C} \ , \\
T_{(320)} - T_{(640)} &= 0.015487°\text{C} \ , \\
T_{(640)} - T_{(1280)} &= 0.007726°\text{C} \ , \\
T_{(1280)} - T_{(2560)} &= 0.003858°\text{C} \ .
\end{aligned}$$

Do you see the pattern? These numbers differ by a factor of about 2 from one level to the next. For example, for the first two levels,

$$\frac{T_{(10)} - T_{(20)}}{T_{(20)} - T_{(40)}} = 2.03799 \ , \tag{17.19}$$

and for the last two levels,

$$\frac{T_{(640)} - T_{(1280)}}{T_{(1280)} - T_{(2560)}} = 2.003 \ . \tag{17.20}$$

This is not a coincidence, rather, a consequence of the fact that the errors are approximately proportional to N^{-1}.

Let's show why this should be the case. Express the final temperature as

$$T_{(N)} = T_{(exact)} + C/N \ , \tag{17.21}$$

where $T_{(exact)}$ is the (unknown) exact answer and C is a proportionality constant. This relation holds, approximately, for any number of timesteps. In particular, for $2N$ and $4N$ timesteps we have

$$T_{(2N)} = T_{(exact)} + C/(2N) \ , \tag{17.22a}$$

$$T_{(4N)} = T_{(exact)} + C/(4N) \ . \tag{17.22b}$$

These last three equations combine to yield

$$\frac{T_{(N)} - T_{(2N)}}{T_{(2N)} - T_{(4N)}} = 2 \ . \tag{17.23}$$

This suggests a simple test for any Euler's method code, called a *three–point convergence test.* Compute the expression (17.23) using three successive values for the number of timesteps. The result should be close to 2 in the limit as the three resolutions are increased. If it isn't, there is likely an error in the code.[1]

Exercise 17.5

Apply the three–point convergence test to your code from Exercise 17.4a.

[1] For some systems, such as the driven damped pendulum of Ch. 19, a convergence test might fail due to the growth of machine roundoff errors.

17.6 Richardson extrapolation

Since we know how the errors change as N increases, we can use this information to predict the temperature for values of N beyond 2560. Of course, what we really want is the value of T as $N \to \infty$. In this limit $T_{(N)}$ equals the exact value, $T_{(exact)}$. To find $T_{(exact)}$, we can use the relations

$$T_{(low)} = T_{(exact)} + C/(N_{low}) \ , \tag{17.24a}$$

$$T_{(high)} = T_{(exact)} + C/(N_{high}) \ , \tag{17.24b}$$

for two values of N, which we call N_{low} and N_{high}. Eliminating C and solving for the exact temperature gives

$$T_{(exact)} = 2T_{(high)} - T_{(low)} \ , \tag{17.25}$$

assuming $N_{high} = 2N_{low}$.

This result is known as *Richardson extrapolation*. In practice, we usually apply Richardson extrapolation to the two highest resolutions available. For our Newton's cooling problem with ambient temperature (17.16),

$$T_{(exact)} = 2T_{(2560)} - T_{(1280)} = 137.336313^\circ\text{C} \ . \tag{17.26}$$

Of course, this isn't really the *exact* answer; keep in mind that Eqs. (17.24) are approximations that ignore terms proportional to higher powers of $1/N$. Nevertheless, the result from Richardson extrapolation should be more accurate than any of the individual results in Table 17.1.

Just how accurate is the result (17.26)? As usual, it's not possible to judge the accuracy without further information. Let's do a numerical experiment. Consider a code that uses Euler's method to solve Newton's cooling with ambient temperature (17.16) using two values for the timestep, N_{low} and N_{high}. The code then uses Richardson extrapolation to find the final temperature. A few results are shown here:

N_{low}	N_{high}	$T(^\circ\text{C})$
320	640	137.336269
640	1280	137.336304
1280	2560	137.336313

Evidently the final temperature is $T = 137.3363^\circ$C to 7 significant figures. To 8 significant figures the answer is probably 137.33631°C or 137.33632°C, but we need further tests to be sure.

Richardson extrapolation can be applied in many contexts, including numerical integration and differentiation. In general, our numerical schemes yield results with errors proportional to some power of $1/N$, where N is a

measure of the resolution. For ODEs, N is the number of timesteps; for numerical integration, N is the number of subintervals; for numerical differentiation, $N \propto 1/h$ where h is the separation between stencil points. For a numerical method of "order n," the leading term in the error is proportional to $1/N^n$. In this case we can compute the numerical approximations of some quantity Q using resolutions $N_{(low)}$ and $N_{(high)}$. The low and high Q values will be related to the "exact" answer by

$$Q_{(low)} = Q_{(exact)} + C/(N_{low})^n \ , \tag{17.27a}$$

$$Q_{(high)} = Q_{(exact)} + C/(N_{high})^n \ . \tag{17.27b}$$

Assuming $N_{high} = 2N_{low}$, these equations imply

$$Q_{(exact)} = \frac{2^n Q_{(high)} - Q_{(low)}}{2^n - 1} \ . \tag{17.28}$$

For example, Simpson's rule is an order $n = 4$ scheme for numerical integration, with errors proportional to N^{-4}. You can use Simpson's rule to approximate an integral at resolutions N_{low} and $N_{high} = 2N_{low}$, then use Richardson extrapolation (17.28) with $n = 4$ to obtain an improved answer.

Exercise 17.6a

Consider once again Newton's law of cooling with the sinusoidally varying ambient temperature of Eq. (17.16). Use the parameter values $a = 100°\text{C}$, $b = 30°\text{C}$, $c = 400\,\text{s}$, $T_0 = 10°\text{C}$ and $k = 0.005$. Solve this problem using Euler's method at two resolutions, N_{low} and $N_{high} = 2N_{low}$, and apply Richardson extrapolation. Determine the body temperature at time $8000\,\text{s}$ with an accuraracy of 6 significant figures.

Exercise 17.6b

Use Simpson's rule and Richardson extrapolation to approximate the integral

$$I = \int_0^{10} \ln(x + \cos(x))\, dx$$

to 8 significant figures.

Exercise 17.6c

Consider the function

$$f(x) = e^{\sin(x+x^2)} \ .$$

What is $f'(0)$? Use the centered 3–point stencil to approximate $f'(0)$ at high resolution $h = 0.1$, and at low resolution $h = 0.2$. Verify that the centered 3–point difference formula for $f'(x)$ is an order $n = 2$ approximation by showing that the error decreases by a factor of about 4 between the low and high resolution results. Use Richardson extrapolation to find an improved answer for $f'(0)$.

17.7 Practical issues with error estimation

We worry a lot about errors in numerical calculations. You might ask: Can I just use a really high resolution (large N) and simply assume the answer is "accurate enough?" This strategy might be fine if you want a rough answer to a simple problem. But remember, there are always machine roundoff errors. If your resolution is too high, roundoff errors can spoil the results. There are also practical limitations on resolution. Realistic research problems can be very complicated, and can stretch the limits of human patience and computer memory. You might not be able to reach a really high resolution because it simply takes too long for the calculation to finish, or because it overloads the available memory on your computer.

You might also reason: "I want an answer that is accurate to, say, 3 significant figures. I will instruct Python to print the answer to 3 significant figures, then increase the resolution until the numbers stop changing." Is this a valid strategy? The short answer is maybe. This strategy sometimes works, but not always. Consider an example problem with exact answer $Q_{(exact)} = 1.4443$. If the numerical scheme has errors of order N^{-1}, then the numerical answers at resolution N have the form

$$Q_{(N)} = 1.4443 + C/N \tag{17.29}$$

for some constant C. Let's assume $C = 0.0321$. The results for Q with increasing N are shown in this table:

N	1	2	4	8	16	32	64	128
Q	1.48	1.46	1.45	1.45	1.45	1.45	1.44	1.44

If we were to follow the proposed strategy, we would see that the answer is unchanged from $N = 4$ to $N = 8$ and declare the answer to be 1.45. This is wrong; the correct answer to 3 significant figures, 1.44, does not appear until the resolution reaches $N = 64$. This strategy can be especially misleading if N is increased by a small amount from one numerical test to the next.

Exercise 17.7

Use Simpson's rule to approximate the integral

$$I = \int_0^\pi \sin(x)\, dx \ ,$$

using $N = 10^9$ timesteps. How long does it take for your computer to complete the calculation? (Use the `time()` function.) The exact answer is $I = 2$. How does the error at $N = 10^9$ compare to the error at $N = 10^3$?