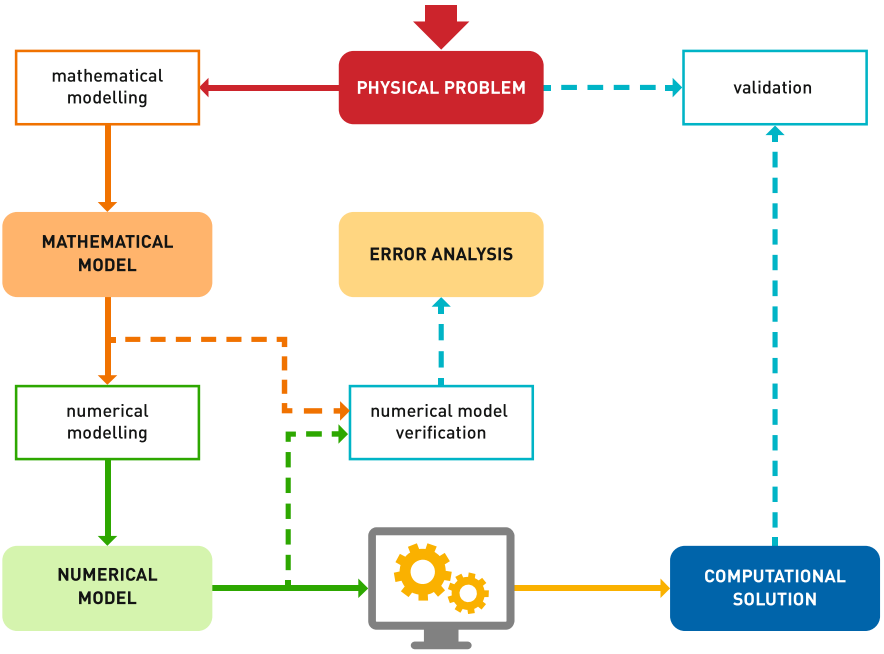
A Warm-Up to Models

1



Boxes with white background represent processes, while coloured boxes represent either the input or the output of those processes.

INGREDIENTS	Definition of function of one variable, average velocity.
WHAT WE LEARN	Mathematical model, numerical model, errors, validation, verification.
PREREQUISITES	None.

A. Quarteroni, P. Gervasio, *A Primer on Mathematical Modelling*, UNITEXT 121,
https://doi.org/10.1007/978-3-030-44541-6_1

1.1 Mathematical Modelling

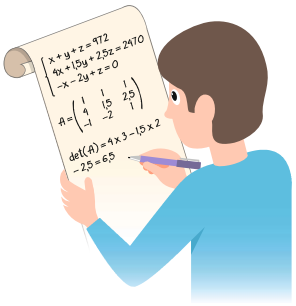
The various aspects of the real world, their interaction and their dynamics can very often be described by mathematical formulas, functions and equations, in other words by *mathematical models*. The purpose of *mathematical modelling* is to translate certain problems arising from the real world into mathematical problems, and provide possible solutions.

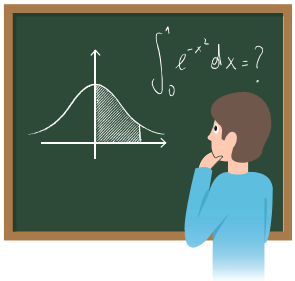
Rather than a subject, it is a process that starts from a real phenomenon or problem of practical interest (from now on called *physical problem*), and aims to:

1. detect the aspects that are most relevant and meaningful;
2. translate them into a mathematical model;
3. compute the solution of the model;
4. verify its soundness, by comparing the solution thus found to the observed (and quantified) phenomenon.

In the study of complex problems very often computing the solution of the model by hand, on paper, is difficult, if not impossible. This difficulty can be attributed to two distinct factors.

Complexity. In certain situations, even if we know the abstract formula for the solution, using it would involve an enormous amount of calculations and take too much time, sometimes even years or decades (as we shall see in Chap. 2, Sect. 2.2).





Unawareness of the solution. In other cases (the vast majority) we cannot even characterise the solution of the mathematical model, meaning we are not able to write a formula to solve the problem explicitly (see Chap. 2, Sect. 2.1).

Numerical modelling and *scientific computing* will help us overcome these obstacles.

1.2 Numerical Modelling and Scientific Computing

Numerical models are examples of formulas and equations, just like mathematical models. What distinguishes them is that the solution to the former, called the *numerical solution*, can be calculated by *numerical methods*, i.e. methods which can be readily translated into *algorithms* and then *programs* executable by a *computer*.

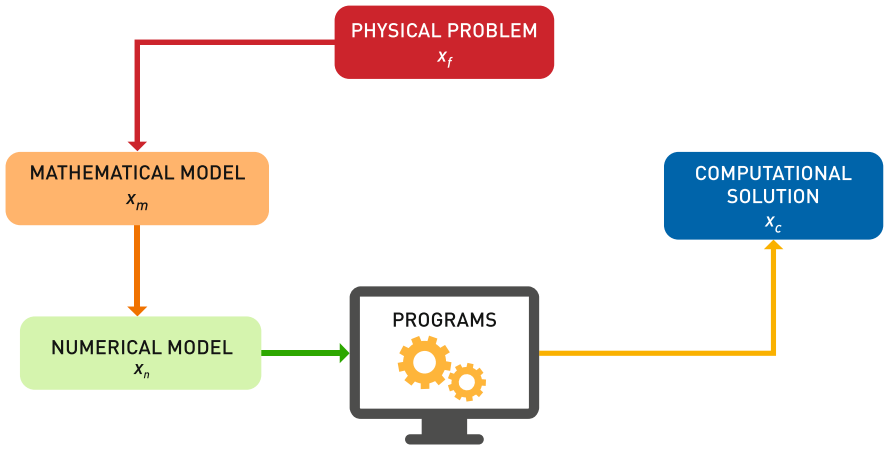


Fig. 1.1 From the physical problem to the computational solution: x_f is the solution to the physical problem, x_m the solution to mathematical model, x_n the solution to the numerical model, x_c the computational solution (produced by the computer)

Therefore numerical models show up in the process, after mathematical models, as Fig. 1.1 shows.

Numerical models represent the bridge between mathematical models and a computing machine, which may be the calculator app on our phones or the most powerful supercomputer in the world, one that is able to perform millions of billions of operations per second.

The final stage of the whole process of modelling a physical problem is the solution generated by the computer, which is called the *computational solution*.

Scientific Computing

Scientific Computing is the subject that takes a mathematical model, cooks up a numerical model, proposes algorithms to determine the computational solution, and then verifies how correct and precise the results of the computer are by comparing them with the real situation.

Here is a more detailed characterisation of mathematical/numerical modelling. It is a process that, starting from a physical problem, aims to

1. detect the aspects that are most relevant and meaningful;
2. translate them into a mathematical model;
3. formulate a numerical model;
4. suggest algorithms for solving the numerical model;
5. translate the algorithms into programs which, once implemented, provide the computational solution;
6. verify the soundness of the computational solution by comparing it with observations and measurements, i.e. with what truly happens with the initial phenomenon.

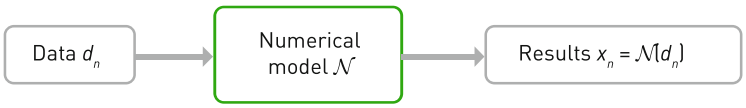
1.3 Data–Model – Results

We may view a mathematical model $\mathcal{M}$ as a tool that acts on the *data* d_m (approximations of the data d_f of the physical problem) and produces the *solution* (or more generally the *results*) x_m. The subscript m reminds us of the word 'mathematical'.

The data d_m are approximations of the physical data d_f, just like x_m is an approximation of the solution x_f of the physical problem we want to solve. We may picture the process by this chart:

The power of the mathematical model $\mathcal{M}$ is that it can act on different data without changing its essence. Clearly the results will depend both on the data and the model we have chosen.

The numerical model $\mathcal{N}$, instead, is a tool acting on an approximation d_n of the data d_m and producing the numerical solution x_n, itself an approximation of the solution x_m. The subscript n refers to *numerical*:



At last, the computational solution generated by the computer, through the program that implements the numerical method, is denoted by x_c. Since, as we shall see later, the computer inevitably introduces errors, *in general the numerical solution and the computational solution are not the same*.

Existence and uniqueness

We are assuming that our mathematical model $\mathcal{M}$ admits one, and only one, solution corresponding to each choice of datum d_m and similarly, that the numerical model has exactly one solution for every choice of datum d_n. This property (the *existence and uniqueness of the solution*) will be a necessary requirement of all mathematical and numerical models we shall meet in the sequel.

1.3.1 Different Physical Problems, the Same Mathematical Model (a Subtle, Yet Crucial, Difference)

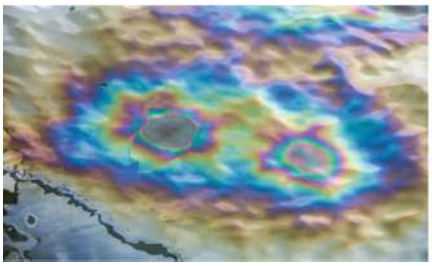
A single mathematical model can be employed for many different problems having common features.

For example, one mathematical model $\mathcal{M}$ that solves several problems in fluid dynamics (air, water, ...) is the so-called system of *Navier-Stokes equations*.[1]

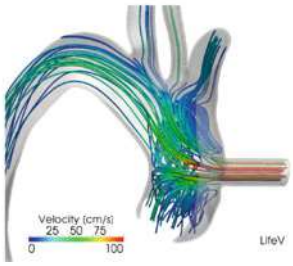
[1] Claude-Louis Navier (1785–1836) was a French engineer and scientist; George Gabriel Stokes (1819–1903) was an Irish mathematician and physicist.

(a) Photo by Ricard Patou from Pixabay, 2018; id 3388188

(b) Photo by Waltteri Paulaharju from Pixabay, 2014; id 1263169

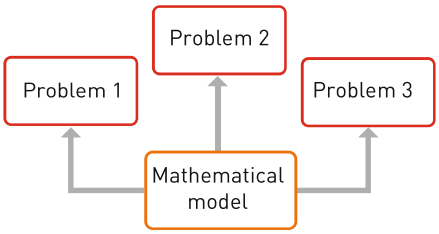


(c) Photo by Dennis Larsen from Pixabay, 2013; id 1438862



(d) Image by CMCS-EPFL, Lausanne (Switzerland) 2017

Fig. 1.2 Different physical problems described by the same mathematical model (the Navier-Stokes equations): trouble waters in a river (**a**); water waves around a rowing hull (**b**); mixing of two fluids with different densities (**c**); streamlines in a bifurcating artery (the computational solution) (**d**)



These equations simulate the motion of fluids in canals, rivers, basins (seas or lakes), around floating bodies (a boat, a canoe or a swimmer), or inside the blood vessels of the human body (see Fig. 1.2).

The data d_f of the physical problem are for instance represented by the shape of the riverbed or the lake, the profile of a swimmer or boat, and the features of the fluid under consideration (such as its viscosity and density). The results x_f are the velocity and pressure of the fluid at each point of the geometrical region in which it flows.

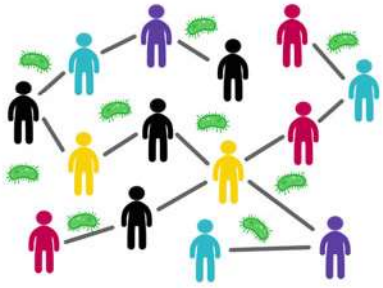
Mathematical models are usually grouped according to the type of problems they can tackle. There are mathematical models to study fluid motion as mentioned above, the deformation of solid constructions (say, buildings and bridges subject to the seismic waves of an earthquake), the dynamics of animal populations competing for survival, the evolution of financial markets, the spread of epidemics, and many more phenomena (see Fig. 1.3).

Let us point out that there are even more general mathematical models which are capable of solving problems of very different nature. A single mathematical model

(a) Photo by Angelo Giordano from Pixabay, Onna (Italy) 2009; id 1665878

(b) Photo by Natalia Kollegova from Pixabay, 2009; id 2514853

(c) Photo by Gerd Altmann from Pixabay, 2016; id 1426331



(d) Image by Caterina Saleri, 2020. Reproduced with permission

Fig. 1.3 Types of real problems: the study of deformations of structures subject to earthquakes (**a**); the study of problems in ecology (**b**); the study of the behaviour of financial markets (**c**); the study of the spread of epidemics (**d**).

may for example allow to understand how heat propagates across the walls of a building, what is the electric potential generated by a certain distribution of charge in the surrounding space, or how a soluble substance disperses in a solution: in this latter case a model could describe the concentration of a drug in a watery solution[2] (see Fig. 1.4).

Model

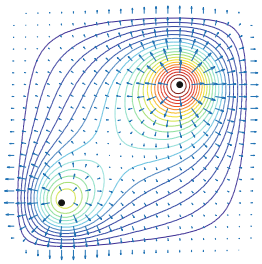
A *model* is a general (and natural) law with major descriptive power.

1.3.2 One Physical Problem, Many Mathematical Models

We claimed above that the solution $x_m = \mathcal{M}(d_m)$ not only depends on the data, but on the chosen model $\mathcal{M}$ as well.

[2]The Poisson equation is $-\Delta u = f$, where f is a given function, u is an unknown function and Δ indicates the sum of the pure second derivatives of u (we shall discuss derivatives in Chap. 6).

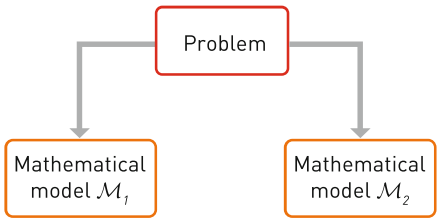
(a) Photo by Marco Danesi, in *Eutopia Urbanscape, the combined redevelopment of social housing*, B. Angi (ed.), LetteraVentidue, Siracusa, ISBN: 9788862421904, 2016. Reproduced with permission



(b)

(c) Photo by Sarah Richter from Pixabay, 2016; id 2427263

Fig. 1.4 The mathematical model given by the Poisson equation, and three physical problems it solves: the thermal insulation of a building and the propagation of heat through the walls (**a**); the electric potential generated by two ideal charges (**b**); the diffusion of a drop of soluble substance in a liquid (**c**)



Let us consider the problem of determining how the polluting particles emitted by a chimney disperse in the atmosphere. We may start by defining a mathematical model $\mathcal{M}_1$ to find out the concentration of a polluting agent by only considering the phenomenon of *diffusion* of a substance (the pollutant) inside another substance (air). This model produces a solution $x_1 = \mathcal{M}_1(d_1)$.

Then we may consider a mathematical model $\mathcal{M}_2$ which, apart from simulating the effects of the diffusion, also considers the *wind* blowing around the chimney. The solution $x_2 = \mathcal{M}_2(d_2)$ produced by the second model differs from solution x_1 when the effect of the wind is significant, and turns out to be more faithful than the solution of the simpler model $\mathcal{M}_1$.

Simplification

We have built two mathematical models to describe the same physical problem, but with different degrees of accuracy. Each time we make a simplification we are actually ignoring one aspect of reality, and thus we are introducing an approximation of the phenomenon. The fewer simplifications we make, the more accurate the solution we compute will be, and the bigger the effort needed to find it.

1.4 Errors, Errors... But No Panic

The Error of the Model

When we pass from a real-life problem to a mathematical model we introduce an error. This happens because reality will not let itself be easily "reduced" to an equation or a collection of equations, irrespective of how sophisticated they are. The price to pay for using a mathematical model is quantified by the discrepancy between the real solution x_f and the (theoretical) solution x_m of the mathematical model (see Fig. 1.5). This difference is the *error of the model*:

$$e_m = x_f - x_m.$$

The Numerical Error

A second type of error, called *numerical error*, measures the difference between the solutions of the mathematical and numerical models:

$$e_n = x_m - x_n.$$

Numerical errors are caused by the approximation of mathematical "operations".

For example, we produce a numerical error when we approximate the area of a *trapezoid*, the region in the plane between the x-axis, two vertical lines and the graph of a function $y = f(x)$ (in blue in Fig. 1.6). Here x_m is the exact value, while x_n is the approximation given by adding the areas of the rectangles (coloured in red in Fig. 1.6). The red rectangles all have basis of the same length $\Delta x > 0$ but differ by height, which in each rectangle equals the value of the function f at the midpoint of its basis.

The numerical error $e_n = x_m - x_n$ measures the difference between the exact area and the approximate area, and depends on the length Δx of the bases. It is intuitively clear that the error will become smaller as Δx becomes small, or equivalently, as the number of rectangles increases.

We also make a numerical error when we set out to determine the speed of a moving car and we approximate the speed v at the instant t using the average speed over a time interval (see Fig. 1.7). The average speed $\bar{v}$ is by definition the ratio of the variation of the position $\Delta s = s_2 - s_1$ over the time interval $\Delta t = t_2 - t_1$ of the

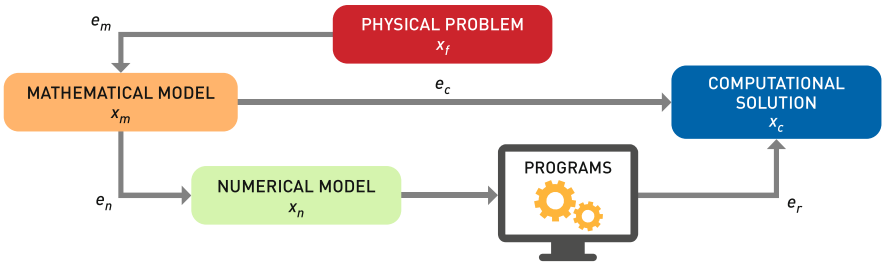


Fig. 1.5 The errors introduced along the path going from the physical problem to the computational solution: the error of the model e_m, the numerical error e_n, the roundoff errors e_r, and the computational errors e_c

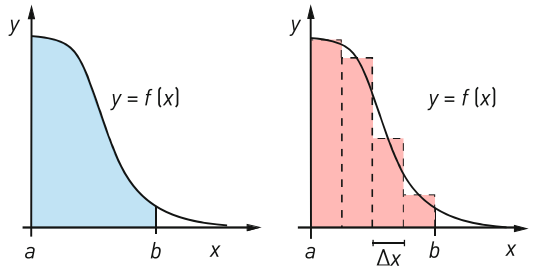


Fig. 1.6 The solution x_m of the mathematical problem is the area of the blue trapezoid, while the solution x_n to the numerical problem is the sum of the areas of the red rectangles. The numerical error $e_n = x_m - x_n$ is the difference between the exact blue area and the approximate area

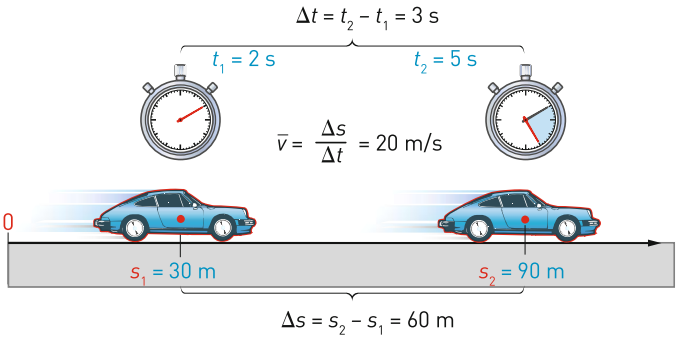


Fig. 1.7 Average speed of an object in motion

displacement; t_1 and t_2 are two instants at which we can measure the position of the car, while the instant t, at which we seek the instant speed, is between t_1 and t_2.

In this case the solution of the mathematical model is $x_m = v$ (the speed at time t), whereas the solution to the numerical model is

$$x_n = \bar{v} = \frac{\Delta s}{\Delta t} = \frac{s_2 - s_1}{t_2 - t_1}.$$

If the speed is not constant, we will make a numerical error

$$e_n = x_m - x_n = v - \bar{v}.$$

The *Tutor* speed-control system on motorways does exactly that: it computes the average speed over a stretch of motorway rather than the instant speed at one point. Two cameras, placed for instance 1 km apart ($\Delta s = 1$ km), detect the instants t_1, t_2 at which a specific vehicle passes by, and transmit the data to a computer that calculates the average speed

$$x_n = \bar{v} = \frac{\Delta s}{\Delta t} = \frac{1 \text{ km}}{t_2 - t_1}$$

over that stretch of road. Obviously this system cannot measure the exact speed at a given instant, and this speed may vary quite a bit, depending on overtaking and braking.

There also are situations where the numerical model coincides with the mathematical model (for example when the mathematical model is a linear system, as we shall see in Chap. 2), $x_n = x_m$, so the numerical error is zero.

Roundoff Errors and Computational Errors

This is not the end of the story. The program used to implement the numerical method on a computer will introduce further *errors* e_r, called *roundoff errors*, due to fact that the algebraic operations performed by the computer are themselves affected by errors.[3]

So we will have:

$$e_r = x_n - x_c.$$

Furthermore, the sum of the numerical and roundoff errors produces *computational errors*:

$$e_c = e_n + e_r = (x_m - x_n) + (x_n - x_c) = x_m - x_c.$$

It is almost by a miracle (so to speak…it is mathematics that enables us to do so) that all these errors can be kept in check. If we follow the rules the computational solution will very much look like the real one[4]

$$x_c \simeq x_f,$$

and only in such case we can say that the mathematical model was helpful.

The Validation of the Process and the Verification of the Numerical Model

The process of comparing the computational solution and the physical solution, that is to say, the control of the error,

$$x_f - x_c = (x_f - x_m) + (x_m - x_c) = e_m + e_c,$$

is also known as *validation of the whole process* consisting in approximating x_f by x_c.

For this process to be validated it must be *verified* first. By *verification of the numerical model* we mean the comparison of the solutions of the mathematical and

[3] Any computer rounds off the input and output of its operations, because all numbers must be stored using a finite number of bits. The resulting arithmetic is called *floating-point arithmetic* to distinguish it from *exact arithmetic*, the one we would do by hand. These notions will be developed further in Sect. 3.2, Chap. 3.

[4] ($a \simeq b$ is read *a is approximately equal to b*).

numerical models

$$x_n \simeq x_m,$$

which amounts to having control on the numerical error e_n. The verification of a numerical model, therefore, is a necessary condition for the validation of the entire process.

What Do Errors Depend On?

The different kinds of errors depend on a number of factors. In a nutshell, for the numerical error we may assume that

$$e_n = e_n(h),$$

in other words that e_n depends on a parameter $h > 0$ that denotes the generic *discretisation parameter* (in the previous examples, h is the width Δx of the intervals used to approximate the area of the trapezoid, or Δt when we approximate the instant speed by the average one).

Similarly, for the roundoff error we may suppose

$$e_r = e_r(\mathtt{u}),$$

so that e_r depends on a parameter $\mathtt{u}$,[5] called *roundoff unit*, which measures how much our computer is accurate in approximating real numbers and in performing algebraic operations.

The error $e_r(\mathtt{u})$ is a measure of how much the calculation depends on the computer itself. This precision is finite, because the machine cannot work using exact arithmetic (as we would, ideally). We shall address numerical errors in Chaps. 2 and 6, and roundoff errors in Chap. 3.

The most complicated error to quantify is the error e_m of the model, for it depends on the physical problem and on how well the adopted mathematical model is able to represent nature faithfully.

Only the final validation, that is, the comparison of the computational solution with the measurements of observable physical quantities (for instance, the temperature predicted by a weather forecast model and the actual temperature detected) will tell us how faithful and reliable the mathematical model we have adopted is. This is the true, ultimate test for a mathematical model.

Approximation

It is inevitable for the computational solution to be imprecise: it will always give an approximation of the exact (and seldomly computable) solution of the mathematical model.

[5] What is $\mathtt{u}$ exactly? When we employ Octave (this program will be useful later) we have $\mathtt{u} \simeq 10^{-16}$. For pocket calculators typically $\mathtt{u} \simeq 10^{-9}$.

The problems discussed in this book have smallish dimension, and the corresponding roundoff errors are indeed negligible. For this reason in the next chapters we will always speak of *numerical solutions*, even when we shall actually be considering computational solutions.

We shall hence assume $x_c = x_n$, that is: $e_r = 0$.

1.5 There Is Error and Error

So, making *errors* when we use mathematical models is inevitable. And as a matter of fact, we want this! Should we worry about it? Sure, but just a little. Or rather, we should not worry too much, provided there are mathematical justifications in place when we do so. What really matters is being able to "quantify" these errors, suggest estimates for their size. We know we are generating errors, but we should also be aware of how much we deviate.

As we have seen in the previous section, concerning the area of a trapezoid, the numerical solution x_n is typically obtained in terms of a parameter $h > 0$, called discretisation parameter. We shall emphasise this dependency by writing $x_n(h)$.

The hope is that by taking increasingly smaller positive values of h (closer and closer to zero) the numerical solution $x_n(h)$ will get closer to the solution of the mathematical model x_m (see Fig. 1.8, where $h = \Delta x$).

If this happens, we write

$$x_n(h) \to x_m \qquad \text{as} \qquad h \to 0.$$

We spell out the above as follows: $x_n(h)$ tends to x_m as h tends to 0.[6]

Consequently, as h becomes smaller the numerical error $e_n(h) = x_m - x_n(h)$ will approach zero:

$$e_n(h) \to 0 \qquad \text{as} \qquad h \to 0,$$

read: $e_n(h)$ tends to 0 as h tends to 0.

Convergence

If

$$e_n(h) \to 0 \qquad \text{as} \qquad h \to 0 \tag{1.1}$$

we say that the numerical method is *convergent*, or that it *converges*.

For example, the numerical method that approximates a trapezoid by a sum of rectangles of width $h = \Delta x$ (Fig. 1.8) is a convergent method, because one can prove that the error $e_n(h) = x_m - x_n(h)$ tends to zero as h tends to zero.

[6] In more refined terms: x_m is the limit of $x_n(h)$ as h tends to 0.

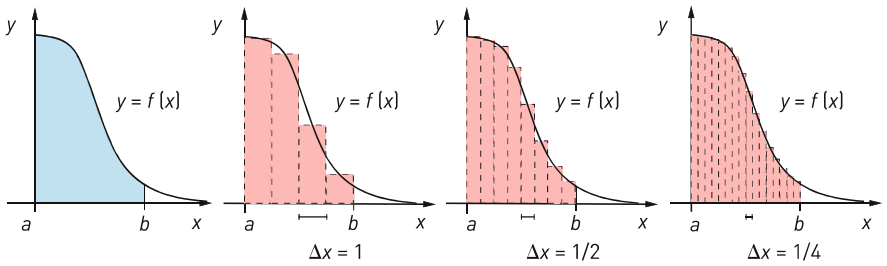


Fig. 1.8 The numerical solution x_n is the sum of the areas of the rectangles, and its value depends on the discretisation parameter $h = \Delta x$. The numerical error $e_n = x_m - x_n$, i.e. the difference between the exact and approximate areas, becomes as small as we want, as $h = \Delta x$ becomes small

In fact, Fig. 1.8 shows that as h decreases, the region obtained by adding the rectangles becomes indistinguishable from the trapezoid.

The proof of this fact requires notions and theorems from infinitesimal calculus.

Convergence of order p

Suppose there exist a positive constant C, independent of h, and a positive integer p (also independent of h) such that the numerical error can be bounded in terms of h as follows

$$\left|e_n(h)\right| \leq Ch^p \quad \text{as} \quad h \to 0. \tag{1.2}$$

Then we say that the *numerical method converges with order* p.

Referring to the approximation of the area of the trapezoid, one can prove that the following error estimate holds:

$$\left|e_n(h)\right| \leq Ch^2 \qquad \text{as} \ \ h \to 0,$$

for a suitable positive constant C, and we say that the method converges with order 2 with respect to h.[7] We shall have the opportunity to understand this method better in Chap. 2.

At times we may also replace the symbol $\leq$ by $\approx$ (read "behaves like") in the error estimate, and thus write

$$\left|e_n(h)\right| \approx h^p \quad \text{as} \quad h \to 0.$$

This amounts to say there exist positive constants C and c, independent of h, for which

$$ch^p \leq \left|e_n(h)\right| \leq Ch^p \quad \text{as} \quad h \to 0.$$

In other words, the absolute value of the error behaves exactly like h^p when $h \to 0$.

[7] Halving h, for example, i.e. taking twice as many intervals, we reduce the error by a factor of 4.

Error Estimators

A careful inspection tells that errors are not computable quantities, for they depend on a constant C that in turn depends on the unknown of the problem itself, which we obviously do not know.

So sometimes it becomes necessary to introduce *error estimators*: computable quantities that give a good indication of the error made. We will meet some instances in the subsequent chapters.

1.6 What We Have Learnt

We have learnt:

1. what it means to make a *mathematical model*;
2. that mathematical models often require approximations by *numerical models*, which are implementable on *computers*;
3. that numerical models, by nature, provide *error-affected* solutions (whence numerical solutions may be different, in principle, from the incomputable solutions of the mathematical model);
4. that, despite everything, errors can be *predicted and measured*.