Module 1 - Roundoff and Truncation Errors

Reading

(1) § 4.1 Errors

(2) § 4.2 Round-off Errors

(3) § 4.3 Truncation Errors

(4) § 4.4 Total Numerical Error

Lesson goals

1. Understanding the distinction between accuracy and precision.
2. Learning how to quantify error.
3. Learning how error estimates can be used to decide when to terminate an iterative calculation.
4. Understanding how roundoff errors occur because digital computers have a limited ability to represent numbers.
5. Understanding why floating-point numbers have limits on their range and precision.
6. Recognizing that truncation errors occur when exact mathematical formulations are represented by approximations.
7. Knowing how to use the Taylor series to estimate truncation errors.

Errors

Accuracy vs. Precision

Accuracy refers to how closely a computed or measured value agrees with the true value.

Precision refers to how closely individual computed or measured values agree with each other.

Example. (Illustration of accuracy and precision) The following graphs show the difference between accuracy and precision. The bullet holes on each target can be thought of as the predictions of a numerical technique, whereas the bull's-eye represents the truth.

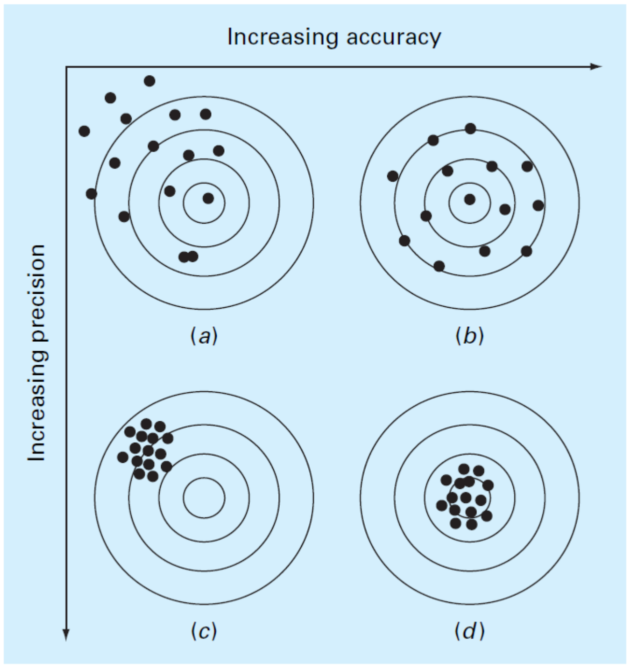


Figure 1: (a) inaccurate (bias) and imprecise (uncertain), **(b)** accurate and imprecise, **(c)** inaccurate and precise, **(d)** accurate and precise.

Types of errors

Absolute error: The relationship between the exact, or true, result and the approximation can be formulated as:

$$\text{True value} = \text{approximation} + \text{error}$$

Thus,

$$E_t = \text{True value} - \text{approximation}$$

The true error is commonly expressed as an absolute value and referred to as the *absolute error*. Thus,

$$\text{Absolute error} = |\text{True value - approximation}|$$

Drawback of absolute error: it takes no account of the order of magnitude of the value under examination. For example, an error of a centimeter is much more significant if we are measuring a rivet than a bridge.

Relative error: To account for the magnitudes of the quantities being evaluated, the relative error normalizes the error to the true value using the following formula:

$$\text{relative error} = \frac{|\text{true value} - \text{approximation}|}{|\text{true value}|}$$

The relative error can also be multiplied by 100% to express it as

$$\varepsilon_t = \frac{|\text{true value} - \text{approximation}|}{|\text{true value}|} \times 100\% \qquad \text{(true percent relative error)}$$

Example. Suppose that we measure the lengths of a bridge and a rivet and come up with 9999 and 9 cm, respectively. If the true values are 10,000 and 10 cm, respectively, then:

- their absolute errors are 1 cm.
- their percent relative errors are 0.01% and 10%, respectively.

Although both measurements have an absolute error of 1 cm, the relative error for the rivet is much greater.

Note. Both absolute and relative errors are based on the true value. However, in actual situations such information is only available when we deal with functions that can be solved analytically.

Alternative: In real-world applications, we obviously do not know the true answer *a priori*. An alternative is to normalize the error using the best available estimate of the true value:

$$\varepsilon_a = \frac{|\text{approximate error}|}{|\text{approximation}|} \times 100\%$$

For example, in *iterative* numerical methods, we use

$$\varepsilon_a = \frac{|\text{present approximaton } - \text{previous approximation}|}{|\text{present approximation}|} \times 100\%$$

and repeat the computations until a *stopping criterion,* such as

$$\varepsilon_a \le \varepsilon_s,$$

is satisfied, where ε_s is a prespecified tolerance.

Remark. We can be assured that the result (solution) is correct to *at least n* significant figures if we set

$$\varepsilon_s = 0.5 \times 10^{-n} \qquad \text{or} \qquad \varepsilon_s = (0.5 \times 10^{2-n})\%$$

Example. Consider the *Maclaurin series expansion* of the function $f(x) = e^x$.

$$e^x = 1 + x + \frac{x^2}{2} + \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots$$

The goal is to approximate $e^{0.5}$ by adding terms of this expansion until the approximate error ε_a falls below a prespecified error criterion ε_s conforming to three significant figures, i.e., $\varepsilon_s = 0.05\%$.

Solution. Note that the true value is: $e^{0.5} = 1.648721 \ldots.$

- By adding the first term, we have: $e^{0.5} = 1 + 0.5 = 1.5$. And the errors are:

$$\varepsilon_t = \frac{|1.648721 \ldots - 1.5|}{|1.648721 \ldots|} \times 100\% = 9.02\%$$

$$\varepsilon_a = \frac{|1.5 - 1|}{|1.5|} \times 100\% = 33.33\%$$

- By adding the second term, we have: $e^{0.5} = 1 + 0.5 + \frac{0.5^2}{2} = 1.625$. And the errors are:

$$\varepsilon_t = \frac{|1.648721 \ldots - 1.625|}{|1.648721 \ldots|} \times 100\% = 1.44\%$$

$$\varepsilon_a = \frac{|1.625 - 1.5|}{|1.625|} \times 100\% = 7.69\%$$

Thus, we have the following table by repeating this procedure.

Terms	Result	ε_t, %	ε_a, %
1	1	39.3	
2	1.5	9.02	33.3
3	1.625	1.44	7.69
4	1.645833333	0.175	1.27
5	1.648437500	0.0172	0.158
6	1.648697917	0.00142	0.0158

Roundoff Errors

Roundoff errors arise because digital computers cannot represent some quantities exactly. They can lead to erroneous results in engineering and scientific problem.

Two major facets of roundoff errors involved in numerical calculations:

1. Digital computers have magnitude and precision limits on their ability to represent numbers.
2. Certain numerical manipulations are highly sensitive to roundoff errors.

Computer Number Representation

A *number system* is merely a convention for representing quantities. Each quantity in a number system is identified by the base and place value.

Example. In decimal system (base-10), a quantity is represented by digits from the set $\{0,1,2,\ldots,9\}$, along with their place values (the rightmost digit in the whole number representation is units or ones, the second rightmost digit is tens, and so on). In this system, the number 6891.25 is represented as:

$6 \times 10^3 + 8 \times 10^2 + 9 \times 10^1 + 1 \times 10^0 + 2 \times 10^{-1} + 5 \times 10^{-2}$ (*Positional notation*)

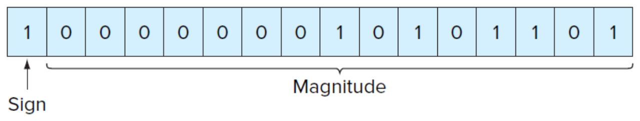
Similarly, in binary system (base-2), the digits are $\{0,1\}$, and a number 1011.01 is equivalent to the following number in the decimal system:

$$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 + 0 \times 2^{-1} + 1 \times 2^{-2} = 11.75$$

Integer Representation

Everyone knows that a whole number can be represented in binary form. One way to store an integer on a computer is to use *signed magnitude method*, in which the first bit indicates the sign (0 for positive and 1 for negative), and the remaining bits are used to store the number.

Example. On a 16-bit computer system, -173 is represented as below:



Remark. On a 16-bit computer system, all integers ranging from $-2^{15} = -32768$ to $2^{15} - 1 = 32767$ can be represented. In general, on an n-bit computer system, all integers ranging from -2^{n-1} to $2^{n-1} - 1$ can be represented

Floating-point Representation

A fractional quantity can be represented by scientific notation as below:

$$\pm s \times b^e$$

where s = the significand (or mantissa), b = the base of the number system being used, and e = the exponent. Note that in this representation the number is normalized by moving the decimal place over so that only one significant digit is to the left of the decimal point. For example:

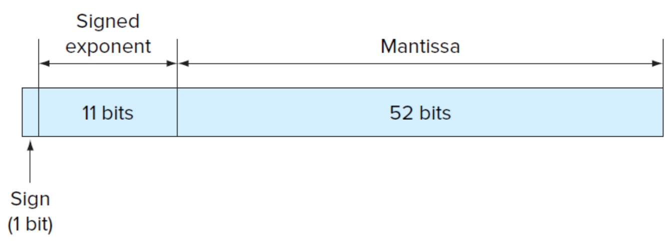
$$0.00047023 = 4.7023 \times 10^{-4}$$

Note. In base-2, the bit to the left of the binary point will always be one and has not been stored. Hence, nonzero binary floating-point numbers can be expressed as

$$\pm(1 + f) \times 2^e$$

where f = the mantissa.

IEEE double-precision format: It uses 8 bytes (64-bits) to store a floating-point number in the computer system as below:



In this fashion:

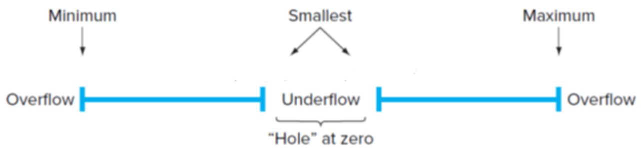
- The integer exponent is ranging from -1022 to 1023 and will be stored in 11 bits
- Largest positive value = $+1.1111\ldots1111 \times 2^{+1023}$
- Smallest positive value = $+1.0000 \ldots 0000 \times 2^{-1022}$
- The 52 bits used for the mantissa correspond to about 15 to 16 base-10 digits. Thus, π would be expressed as 3.14159265358979.
- The machine epsilon is $2^{-52} = 2.2204 \times 10^{-16}$. By definition, the "machine epsilon" is the difference between 1 and the *next* representable number.
- Large positive and negative numbers that fall outside the range would cause an *overflow error*.
- For very small quantities there is a "hole" at zero, and very small quantities would usually be converted to zero.

✓

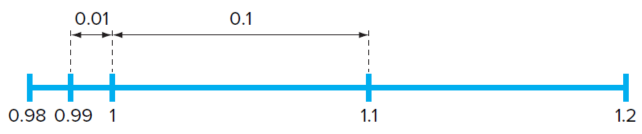
$0.0\cdots0 \times 10^{0}$

$0.0\cdots0\,1 \times 10^{0} = 1.0 \times 10^{-51}$

✓



Remark. In floating-point representation, not every real number is representable. That is, there is always a gap between two consecutive representable numbers on real line. Any number in-between must be stored as the lower or upper number with an error which is called *roundoff error*.



Arithmetic Manipulations of Computer Numbers

The actual arithmetic manipulations involving these numbers can also result in roundoff error. How do the errors in each individual number propagate through the calculations?
Note that, when two floating-point numbers are added or subtracted, they are first expressed in the representable way. For example, to add 1.557 + 0.04341, the computer adds $0.1557 \times 10^1 + 0.004341 \times 10^1 = 0.160041 \times 10^1$. If the computer carries a 4-digit mantissa, the excess number of digits get chopped off and the result is 0.1600×10^1. Similarly, for the subtraction, consider the following cases:

$$0.03621 - 0.2362 = 0.20001$$

While in machine the result will be:

$$0.3621 \times 10^{-1} - 0.2362 \times 10^0 = 0.2000 \times 10^0$$

That means, the subtraction will be subject to roundoff error.

Remark. The subtracting of two nearly equal numbers is called *subtractive cancellation* that can lead to numerical problems. Other cases are:

- *Large computations:* even though an individual roundoff error could be small, the cumulative effect over the course of a large computation can be significant.
- *Adding a large and a small number:* This type of error can occur in the computation of an infinite series. The initial terms in such series are often relatively large in comparison with the later terms. Thus, after a few terms have been added, we are in the situation of adding a small quantity to a large quantity.
- *Smearing:* Smearing occurs whenever the individual terms in a summation are larger than the summation itself (series of mixed signs)
- *Inner products:* Some infinite series are particularly prone to roundoff error as the dot product is. It is often desirable to compute such summations in double precision as is done automatically in MATLAB

Truncation Errors

Truncation errors are those that result from using an approximation in place of an exact mathematical procedure.

Recall. For the infinitely differentiable function $f(x)$ at $x = a$, the function can be written as the infinite sum based on its derivatives by the Taylor expansion:

$$✓\quad f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \cdots + \frac{f^{(n)}(a)}{n!}(x-a)^n + \boxed{R_n}$$

Where $\boxed{R_n = \frac{f^{(n+1)}(\xi)}{(n+1)!}(x-a)^{n+1}}$ is the remainder of the series, and ξ is a value that lies somewhere between x and a. When $\boxed{a = 0}$, the Taylor series becomes the Maclaurin series. One simplification might be to truncate the remainder itself (See Figure 2).

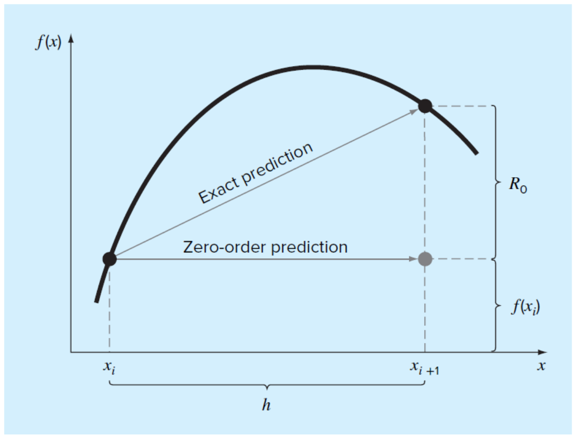


Figure 2. Graphical depiction of a zero-order Taylor Series prediction and remainder

$f(0) + \frac{f'(0)}{1!}x + \frac{f''(0)}{2!}(x)^2 + \cdots$

Example. The Maclaurin series for e^x is given as

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

This series has an infinite number of terms but when using this series to calculate e^x, only a finite number of terms can be used. For example, if one uses three terms to calculate e^x, then,

$$e^x \approx 1 + x + \frac{x^2}{2!}$$

the truncation error for such an approximation is

$$\text{Truncation error} = \left| e^x - \left(1 + x + \frac{x^2}{2!}\right) \right| = \left| \frac{x^3}{3!} + \frac{x^4}{4!} + \cdots \right| = R_2$$

To control the truncation error, we use finite number of terms as long as the relative truncation error is less than a predefined threshold. For example, to estimate $e^{1.2}$ with absolute error to be less than $\varepsilon_s = 1\%$, we use Table 1. This means that, we require 6 terms of the series to approximate $e^{1.2}$ within specified relative error threshold. ■

Definition. Big O notation is a notation that describes the limiting behavior of a function when the argument tends towards a particular value or infinity. When we say,

$$f(x) = O\big(g(x)\big), \text{when } x \to a,$$

It means that $|f(x)| \leq M|g(x)|$, for all x around a.

$$\text{Truncat error} = \left|e^x - \left(1 + x + \frac{x^2}{2}\right)\right| = \left|\frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \cdots\right|$$

$$= O\left(\left|\frac{x^3}{3!}\right|\right)$$

$$= O\left(|x^3|\right)$$

Table 1. The approximation of by $e^{1.2}$ by Maclaurin series

n : number of terms	Approximation of $e^{1.2}$	Relative truncation error $\left\lvert\frac{\text{current value} - \text{previous value}}{\text{current value}}\right\rvert$
1	1	---
2	2.2	0.545454
3	2.92	0.246575
4	3.208	0.089776
5	3.2944	0.026226
6	3.3151	0.006244

$$\longrightarrow \quad v(t) = v(t_i) + \frac{v'(t_i)}{1!}(t-t_i) + \frac{v''(t_i)}{2!}(t-t_i)^2 + \cdots$$

Example. Assume that the interval $[a, b]$ has been partitioned into n-subintervals $\boxed{[t_i, t_{i+1}]}$. To estimate the derivative of velocity of an object at $t = t_i$. We use the Taylor expansion for the function $v(t)$ as follows:

errors

$$t = t_{i+1} \longmapsto \quad \underline{v(t_{i+1})} = \underline{v(t_i)} + \frac{v'(t_i)}{1!}(t_{i+1} - t_i) + \frac{v''(t_i)}{2!}(t_{i+1} - t_i)^2 + \cdots + \frac{v^{(n)}(t_i)}{n!}(t_{i+1} - t_i)^n + R_n$$

Now let us truncate the series after the first derivative term:

$$v(t_{i+1}) = v(t_i) + \frac{v'(t_i)}{1!}(t_{i+1} - t_i) + R_1, \quad \text{where } R_1 = \frac{v''(\xi)}{2!}(t_{i+1} - t_i)^2$$

So, $v(t_{i+1}) = v(t_i) + v'(t_i)(t_{i+1} - t_i) + O\left((t_{i+1} - t_i)^2\right)$

$$v'(t_i) = \underbrace{\frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i}}_{\text{First-order approximation}} - \underbrace{\frac{v''(\xi)}{2!}(t_{i+1} - t_i)}_{\text{truncation error}} = \underbrace{\frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i}}_{\text{First-order approximation}} - \underbrace{O(t_{i+1} - t_i)}_{\text{truncation error}}$$

Thus, the estimate of the derivative has a truncation error of order of step size $t_{i+1} - t_i$. In other words, the error of our derivative approximation should be proportional to the step size. Consequently, if we halve the step size, we would expect to halve the error of the derivative. ■

$$\underline{v'(t_i)} = \frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i} - \frac{O\left((t_{i+1} - t_i)^2\right)}{t_{i+1} - t_i} = \frac{v(t_{i+1}) - v(t_i)}{t_{i+1} - t_i} - \underline{O(t_{i+1} - t_i)}$$

Total numerical error

The total numerical error is the summation of the truncation and roundoff errors. In general, the only way to minimize roundoff errors is to increase the number of significant figures of the computer. Further, we have noted that roundoff error may increase due to subtractive cancellation. In contrast, sometimes the truncation error can be reduced by decreasing the step size. Because a decrease in step size can lead to subtractive cancellation or to an increase in computations, the truncation errors are decreased as the roundoff errors are increased.

Thus, we are faced by the following dilemma: The strategy for decreasing one component of the total error leads to an increase of the other component. In a computation, we could conceivably decrease the step size to minimize truncation errors only to discover that in doing so, the roundoff error begins to dominate the solution and the total error grows!

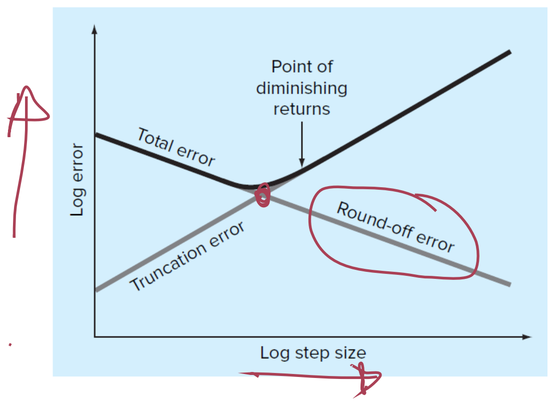


Figure 3. A graphical depiction of the trade-off between roundoff and truncation error that sometimes comes into play in the course of a numerical method.

Example. Error analysis of numerical differentiation

$\overbrace{x_{i-1} \quad x_i}^{h} \overbrace{\quad x_{i+1}}^{h}$

$f_{i-1} \quad f_i \quad f_{i+1}$

The Taylor series can be expanded backward or forward to calculate a previous or next value on the basis of a present value. Let $h = x_{i+1} - x_i$, then

$$✓ f(x_{i+1}) = \cancel{f(x_i)} + \frac{f'(x_i)}{1!}h + \cancel{\frac{f''(x_i)}{2!}h^2} + \cdots + \frac{f^{(n)}(x_i)}{n!}h^n + \frac{f^{(n+1)}(\xi)}{(n+1)!}h^{n+1}$$

$$✓ f(x_{i-1}) = \cancel{f(x_i)} - \frac{f'(x_i)}{1!}h + \cancel{\frac{f''(x_i)}{2!}h^2} + \cdots + (-1)^n\frac{f^{(n)}(x_i)}{n!}h^n + (-1)^{n+1}\frac{f^{(n+1)}(\xi)}{(n+1)!}h^{n+1}$$

By subtracting the second equation from the first one, we obtain:

$$f(x) = f(x_i) + \frac{f'(x_i)}{1!}(x-x_i) + \frac{f''(x_i)}{2!}(x-x_i)^2 + \cdots$$

$$\text{error} = O(h^3)$$

$$\boxed{f(x_{i+1}) - f(x_{i-1}) = 2f'(x_i)h} + \frac{2f'''(x_i)}{3!}h^3 + \cdots$$

$$\frac{O(h^3)}{2h} = O(h^2)$$

$$f'(x_i) = \underbrace{\frac{f(x_{i+1}) - f(x_{i-1})}{2h}}_{\text{centered difference approximation}} - \underbrace{\frac{f^{(3)}(\xi)}{(3)!}h^2}_{\text{Truncation error}}$$

However, because we are using digital computers, the function values do include roundoff error as in

$$f(x_{i+1}) = \tilde{f}(x_{i+1}) + e_{i+1}$$
$$f(x_{i-1}) = \tilde{f}(x_{i-1}) + \boxed{e_{i-1}}$$

roundoff error ✓

$h \uparrow$

$h \downarrow$

where the $\tilde{f}$'s are the rounded function values and the e's are the associated roundoff errors. So,

$$f'(x_i) = \underbrace{\frac{\tilde{f}(x_{i+1}) - \tilde{f}(x_{i-1})}{2h}}_{\text{centered difference approximation}} + \underbrace{\frac{e_{i+1} - e_{i-1}}{2h}}_{\text{roundoff error}} - \underbrace{\frac{f^{(3)}(\xi)}{(3)!}h^2}_{\text{Truncation error}}$$

$O(h^2)$

We can see that the total error of the finite-difference approximation consists of a roundoff error that decreases with step size and a truncation error that increases with step size. An upper bound on the absolute value of the total error can therefore be represented as

$$\text{Total error} = \left| f'(x_i) - \frac{\tilde{f}(x_{i+1}) - \tilde{f}(x_{i-1})}{2h} \right| \le \frac{\varepsilon}{h} + \frac{h^2 M}{6},$$

Where M is an upper bound for the third derivative of $f(x)$ and ε is an upper bound for the roundoff error.

Example. Use forward and backward difference approximations of $O(h)$ and a centered difference approximation of $O(h^2)$ to estimate the first derivative of

$$f(x) = -0.1x^4 - 0.15x^3 - 0.5x^2 - 0.25x + 1.2$$

at $x = 0.5$ using a step size $h = 0.5$. Note that the derivative can be calculated directly as

$$f'(x) = -0.4x^3 - 0.45x^2 - 1.0x - 0.25$$

x	$\overset{h}{x_{i-1}}$	$\overset{h}{x_i}$	x_{i+1}
f	f_{i-1}	f_i	f_{i+1}

and can be used to compute the true value as $\boxed{f'(0.5) = -0.9125}$.

Solution. For $h = 0.5$, the function can be employed to determine

$x_{i-1} = 0, \quad f(x_{i-1}) = 1.2, \qquad x_i = 0.5, \quad f(x_i) = 0.925, \qquad x_{i+1} = 1.0, \quad f(x_{i+1}) = 0.2$

These values can be used to compute the forward difference,

forward diff. formula : $f'(x_i) \simeq \frac{f_{i+1} - f_i}{h} + O(h)$

Backward diff. " : $f'(x_i) \simeq \frac{f_i - f_{i-1}}{h} + O(h)$

Centered diff formula : $f'(x_i) = \frac{f_{i+1} - f_{i-1}}{2h} + O(h^2)$ ✓

$x_{i-1} = x_i - h = 0$ $\quad x_i = 0.5$ $\quad x_{i+1} = x_i + h = 1$ | $f'(0.5) =$ ____

$f(0) =$ $\quad f(0.5) =$ $\quad f(1) =$

✓ $$f'(0.5) = f'(x_i) \cong \frac{f(x_{i+1}) - f(x_i)}{h} = \frac{0.2 - 0.925}{0.5} = \boxed{-1.45}, \qquad |\varepsilon_t| = 58.9\% = 0.589$$

the backward difference,

✓ $$f'(0.5) = f'(x_i) \cong \frac{f(x_i) - f(x_{i-1})}{h} = \frac{0.925 - 1.2}{0.5} = -0.55, \qquad |\varepsilon_t| = 39.7\% = 0.397$$

and the centered difference,

✓ $$f'(0.5) = f'(x_i) \cong \frac{f(x_{i+1}) - f(x_{i-1})}{2h} = \frac{0.2 - 1.2}{1.0} = \boxed{-1.0}, \qquad |\varepsilon_t| = 9.6\% = 0.096 \checkmark$$

Repeat all calculation for $h = 0.25$ (practice at home). Verify the following statements:

For both cases, the centered difference approximation is more accurate than forward or backward differences. Also, as predicted by the Taylor series analysis, halving the step size approximately halves the error of the backward and forward differences and quarters the error of the centered difference. ■

Other Sources of Errors

Although the following sources of error are not directly connected with most of the numerical methods in this book, they can sometimes have great impact on the success of a modeling effort.

- **Blunders:** Blunders can occur at any stage of the mathematical modeling process and can contribute to all the other components of error. They can be avoided only by sound knowledge of fundamental principles and by the care with which you approach and design your solution to a problem.

- **Model errors:** Model errors relate to bias that can be ascribed to incomplete mathematical models. An example of a negligible model error is the fact that Newton's second law does not account for relativistic effects.

- **Data uncertainty:** Errors sometimes enter into an analysis because of uncertainty in the physical data on which a model is based. For instance, suppose we wanted to test the bungee jumper model by having an individual make repeated jumps and then measuring his or her velocity after a specified time interval. Uncertainty would undoubtedly be associated with these measurements, as the parachutist would fall faster during some jumps than during others. These errors can exhibit both inaccuracy and imprecision. If our instruments consistently underestimate or overestimate the velocity, we are dealing with an inaccurate, or biased, device. On the other hand, if the measurements are randomly high and low, we are dealing with a question of precision.

Recommended HomeWorks

Some exercise problems from the textbook that you may wish to try are:

Suggested questions for quizzes

1.

Suggested questions for midterm exam

1.

References

Chapra, Steven C. (2018). *Numerical Methods with* MATLAB *for Engineers and Scientists*, 4th Ed. McGraw Hill.