$$\bar{x} = \text{approximation to true value } x \quad \Rightarrow \quad \text{error} = x - \bar{x}\,, \quad \text{relative error} = \frac{x - \bar{x}}{x}\,.$$

### 2.1.1 Fixed-Point (or Integer) Arithmetic

A base 2 (base 10) *fixed-point* number has a fixed number of binary (decimal) places.

1. Commonly, 32-bit integers range from $-2^{31} = -2,147,483,648$ to $2^{31} - 1 = 2,147,483,647$ (IOFL). Integers in this range can be represented exactly, without round-off or underflow. For integers outside this range, overflow occurs, and results are unpredictable and system dependent.

2. No round-off or underflow in addition $+$, subtraction $-$ or multiplication $*$; but overflow possible.

3. Large errors possible in division / and assignment =, since any fractional part is chopped.

   E.g. `1/2` gives zero and `K = -1.2` assigns `-1` to `K`.

   Beware: mixed (integer and real) arithmetic and implicit typing in FORTRAN — variables with first letter `I,J,K,L,M,N` are `INTEGER` unless declared otherwise. Use `IMPLICIT NONE`.

### 2.1.2 Floating Point (or Real) Arithmetic

A floating-point number has a fixed number of leading digits — the *mantissa*, and an *exponent*. E.g. scientific notation,

$$\pm d_1.d_2 \ldots d_t \times b^p = \pm \left\{ d_1 + d_2 \times b^{-1} + d_3 \times b^{-2} \ldots d_t \times b^{-(t-1)} \right\} \times b^p \,,$$

where mantissa $= d_1.d_2 \ldots d_t$, exponent $= p$, base $= b$.

1. Smallest positive floating-point number = *underflow level* (UFL).

   Largest floating-point number = *overflow level* (OFL).

   UFL $\leq$ positive floating-point numbers $\leq$ OFL.

   $|\text{real number}| > \text{OFL} \Rightarrow$ overflow. $\quad$ $|\text{real number}| < \text{UFL} \Rightarrow$ underflow.

   Beware: some machines set numbers smaller than UFL to zero.

2. *Round-off error* arises from using a finite set of numbers to represent the uncountably-many real numbers between UFL and OFL.

   A real number $x$ has a base $b$ representation

   $$x = \pm d_1.d_2 \ldots d_t d_{t+1} \ldots \times b^p \,.$$

   Commonly approximate $x$ with $\bar{x}$ by either *chopping* or *rounding*.

   In chopping, the digits after $d_t$ (say) are simply "chopped-off":

   $$\begin{aligned} \text{absolute round-off error} = |x - \bar{x}| &= d_{t+1}.d_{t+2} \ldots \times b^{p-t} \\ &< (b-1)\{1 + b^{-1} + b^{-2} + \ldots\} b^{p-t} \\ &= (b-1) \left\{ \frac{b}{b-1} \right\} b^{p-t} = b^{p-t+1} \,. \end{aligned}$$

$$\text{absolute relative round-off error} = \left|\frac{x-\bar{x}}{x}\right| = \frac{d_{t+1}.d_{t+2}\ldots\times b^{p-t}}{d_1.d_2\ldots d_t d_{t+1}\ldots\times b^p} < \frac{b^{p-t+1}}{b^p} = b^{1-t}\,.$$

In rounding, the digits after $d_t$ are rounded up or down (avoiding bias):

$$\text{absolute round-off error} = |x-\bar{x}| \le \frac{1}{2}b\times b^{-t}\times b^p = \frac{1}{2}b^{p-t+1}\,.$$

$$\text{absolute relative round-off error} = \left|\frac{x-\bar{x}}{x}\right| \le \frac{\frac{1}{2}b^{p-t+1}}{b^p} = \frac{1}{2}b^{1-t}\,.$$

E.g. if $b=10$, $p=0$ and $t=2$, then

$$0.567 = \begin{cases} 0.56\,, & \text{chopping;} \\ 0.57\,, & \text{rounding;} \end{cases} \qquad \text{relative round-off error} = \begin{cases} 0.007\,, & \text{chopping;} \\ 0.003\,, & \text{rounding.} \end{cases}$$

3. Overflow and round-off possible in floating-point addition $+$, subtraction $-$, multiplication $*$ and division $/$.

   Underflow possible in multiplication and division.

4. Associative and distributive laws may not hold due to round-off.

   E.g. 4-digit base-10 arithmetic with rounding after each arithmetic operation,

$$\begin{aligned} (4.000\times 10^0 + 5.731\times 10^4) - 5.730\times 10^4 &= (5.7314\times 10^4) - 5.730\times 10^4 \\ &= 5.731\times 10^4 - 5.730\times 10^4 \\ &= 1.000\times 10^1 \end{aligned}$$

but

$$4.000 \times 10^0 + (5.731 \times 10^4 - 5.730 \times 10^4) = 4.000 \times 10^0 + 1.000 \times 10^1 = 1.400 \times 10^1 \,.$$

5. *Catastrophic cancellation* occurs in subtraction of nearly equal floating-point numbers. E.g. $x = 0.76545421$, $y = 0.76544200$ — 8 significant figures;

   $x - y = 0.12210000 \times 10^{-5}$ — only 4 significant figures.

   Avoid subtraction of nearly equal numbers where possible. E.g. use secant formula as good approximation plus small correction, i.e.

$$x_{i+1} = x_i - \frac{x_i - x_{i-1}}{f(x_i) - f(x_{i-1})} f(x_i) \qquad \text{not} \qquad x_{i+1} = \frac{x_i f(x_{i-1}) - x_{i-1} f(x_i)}{f(x_{i-1}) - f(x_i)} \,.$$

6. *machine epsilon*: $\epsilon_{\text{mach}}$ = smallest number such that $1.0 + \epsilon_{\text{mach}} > 1.0$.

   - In scientific notation:

$$1.0 = 1.\overbrace{00\ldots0}^{t-1} \times b^0 \,, \quad 1.0 + \epsilon_{\text{mach}} = \text{next number} = 1.\overbrace{00\ldots0}^{t-2}1 \times b^1 \,.$$

   - $\epsilon_{\text{mach}}$ = worst possible relative round-off error:

$$\epsilon_{\text{mach}} = \begin{cases} b^{1-t} \,, & \text{chopping;} \\ \frac{1}{2} b^{1-t} \,, & \text{rounding.} \end{cases}$$

   - Choose relative error tolerances $\geq \epsilon_{\text{mach}}$, since

$$\left| \frac{\bar{x} - \bar{y}}{\bar{x}} \right| < \epsilon_{\text{mach}} \Rightarrow \bar{x} = \bar{y} \,.$$

7. IEEE 32-bit Floating-Point Standard:

   - $\approx 2^{31}$ numbers; chopped or rounded; non-uniformly distributed: $2^{22}$ numbers between each power of 2, greatest concentration near 0.
   - OFL $\approx 10^{38}$, UFL $\approx 10^{-38}$, $\epsilon_{\text{mach}} = 2^{-23} \approx 1.2 \times 10^{-7}$.

8. *Check*: Redo some or all of the calculations in double-precision and check against single-precision answers.

## 2.2 Truncation Error

Truncation error occurs when an approximation is made to a limiting process, such as in stopping an infinite step process after a finite number of steps. Thus limits such as $\lim_{n\to\infty}$ and $\lim_{h\to 0}$ are replaced by $n$ large but finite and $h$ small but non-zero.

E.g. if $x^*$ = root of some function,

$$\lim_{i\to\infty} x_i = x^*\,, \qquad x^* \approx x_{10}\,, \qquad \text{truncation error} = |x^* - x_{10}|\,.$$

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!}\cdots, \quad e^x \approx 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!}, \quad \text{truncation error} = x^4/4! + x^5/5! + \ldots\,.$$

$$f'(x) = \lim_{h\to 0} \frac{f(x+h) - f(x)}{h}, \qquad f'(x) \approx \frac{f(x+0.01) - f(x)}{0.01}\,.$$

*Check*: Include one more step and compare answers.

## 2.3 Stopping Tests

Let $\{x_i\}$ be a sequence of approximations to some numerical quantity $x_\infty$, such as a root of a function $f(x)$ or a definite integral

$$\int_a^b f(x)\,dx\,.$$

The following tests, if satisfied, provide criteria for stopping the numerical method generating the sequence $\{x_i\}$ at $x_n$. Rounding is assumed.

### 2.3.1 Absolute Change Test

$$|x_{n+1} - x_n| < \epsilon\,.$$

If $\epsilon = 0.5 \times 10^{-p}$, then $x_{n+1}$ and $x_n$ then either $x_{n+1}$ and $x_n$, or their $p$-decimal place rounds, agree to $p$ decimal places. If $|x_\infty| < \epsilon$ this test may terminate the sequence prematurely.

### 2.3.2 Relative Change Test

$$|x_{n+1} - x_n| < \epsilon\,|x_{n+1}|\,.$$

If $\epsilon = 0.5 \times 10^{-p}$, then $x_{n+1}$ and $x_n$ agree to $p$ significant figures ($p > 1$).

### 2.3.3 Combination Test

$$|x_{n+1} - x_n| < \epsilon_a + \epsilon_r\,|x_{n+1}|\,.$$

This test is a combination of the absolute and relative change tests.

### 2.3.4 Function Test

$$|f(x_n)| < \epsilon\,.$$

This test applies only if $\{x_i\}$ is a sequence of approximations to a root of $f$ and tests if $f(x_n)$ is close to zero.

None of these tests, if satisfied, guarantees convergence of $\{x_i\}$ to some numerical quantity $x_\infty$, i.e. that $x_\infty$ exists. A divergent counter-example is the sequence with terms given by

$$x_i = \sum_{k=1}^{i} \frac{1}{k}$$

It satisfies the absolute and relative change tests for any $\epsilon > 0$, if $n$ is large enough, but $x_i \to \infty$ as $i \to \infty$. Thus the existence of $x_\infty$ needs to be established, where possible, by some other means, usually theoretical. Even if $x_\infty$ exists, none of the tests, if satisfied, guarantee that $x_{n+1}$ is close to $x_\infty$ in the sense of the test (or any other sense).

## 2.4 Other Errors

**Blunders or gross errors:** programming errors of logic or syntax, wrong input or output, use of an inappropriate technique, etc.

*Checks*: Are the results reasonable ? E.g. correct signs & orders of magnitude ? If possible, *test program on problem with known answers.*

**Original Data Errors:** inherent in real data and cannot be removed from the numerical problem.

*Check*: How do changes in data affect answers ? Perform a sensitivity analysis by rerunning program with slightly different data.

**Model Errors:** due to the approximate nature of mathematical models.

**Propagated Error:** error at one particular step of a multi-step process due to errors at previous steps.

**Machine Errors:** rarities of two kinds:

*hardware errors*, such as a faulty chip (early pentium), disc or tape drive, or faulty main memory;

*software errors*, such as in the compiler.

*Check*: Run program on different machine or with different compiler.

## 2.5 Condition and Stability

A problem is *ill-conditioned*, *badly-posed* or *sensitive* if small changes in the data produce large changes in the answers.

**Example 2.1** The roots of

$$x^4 - 4x^3 + 8x^2 - 16x + 15.99999999 = (x-2)^4 - 10^{-8} = 0$$

are given by $(x-2)^2 = \pm 10^{-4}$ or $x = 2 \pm 10^{-2}, 2 \pm i10^{-2}$. On a machine with $\epsilon_{\text{mach}} > 10^{-10}$, 15.99999999 rounds to 16.0 and the stored polynomial is $(x-2)^4$ with roots $x = 2$, which differ from the exact roots by 0.5%. Thus a data change of about $10^{-8}\%$ produces changes in the answers of 0.5%, independently of the solution method!

**Example 2.2** One measure of the condition of a function $f(x)$ near $x$ is

$$\max_{\bar{x}} \left\{ \frac{\left| \dfrac{f(x) - f(\bar{x})}{f(x)} \right|}{\left| \dfrac{x - \bar{x}}{x} \right|} \right\} \approx \left| \frac{f'(x)x}{f(x)} \right| .$$

In terms of this condition number $f(x) = \sqrt{x+1} - \sqrt{x}$ is well-conditioned near $x = 15,000$ for

$$\left| \frac{f'(x)x}{f(x)} \right| = \frac{1}{2} \frac{|1/\sqrt{x+1} - 1/\sqrt{x}|\,|x|}{|\sqrt{x+1} - \sqrt{x}|} \approx \frac{1}{2} .$$

A numerical method or algorithm is *stable* if it exactly solves a perturbed problem, one in which the data have been slightly changed; otherwise the method is *unstable*.

A stable method will provide a good approximation to the true solution of a well-conditioned problem. For an ill-conditioned problem a stable method may well give a bad approximation to the true solution although the computed solution may mimic the true solution.

The stability of a method depends on the precision of the arithmetic in which it is implemented and the precision of the results required.

**Example 2.3** Computing $e^x$ using the infinite series,

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

is unstable for $x < -10$ on a 32-bit machine and also for $|x| \gg 1$.

Stable method:

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

for $|x|$ not too large.

**Example 2.4** Evaluate the function $f(x) = \sqrt{x+1} - \sqrt{x}$ at $x = 15,000$ as given, using 6 decimal arithmetic.

**Solution** Since $\sqrt{15001} = 122.479$ and $\sqrt{15000} = 122.474$, $f(15000) = 0.00500000$, whereas the correct value is $f(1000) = 0.004082416$. The computed value is in error by about 25%, even though individual round-off errors in the calculations are less than 0.0001%. But the function is well-conditioned near $x = 15,000$. The dramatic amplification of errors from about 0.0001% to 25% has been caused by using an unstable method to evaluate the function. A stable method is obtained by evaluating the function in the form $f(x) = 1/(\sqrt{x+1} + \sqrt{x})$.

Stable methods are good, well-conditioned problems are nice but unstable methods are mostly useless. There are many important ill-conditioned problems, often arising from inverse data analysis problems and certain types of integral equations.