PARADOXES IN NUMERICAL CALCULATIONS

J. Brandts[*], *M. Křížek*[†], *Z. Zhang*[‡]

Abstract: When solving problems of mathematical physics using numerical methods we always encounter three basic types of errors: modeling error, discretization error, and round-off errors. In this survey, we present several pathological examples which may appear during numerical calculations. We will mostly concentrate on the influence of round-off errors.

Key words: *round-off errors, numerical instability, recurrence formulae, Gram-Schmidt orthogonalization*

Received: October 20, 2015 **DOI:** 10.14311/NNW.2016.26.018
Revised and accepted: June 20, 2016

Dedicated to Prof. Ivo Babuška on his 90th birthday

1. Introduction

The general computational scheme for solving problems of mathematical physics is sketched in Fig. 1. In general, three basic types of errors e_0, e_1, and e_2 are introduced.

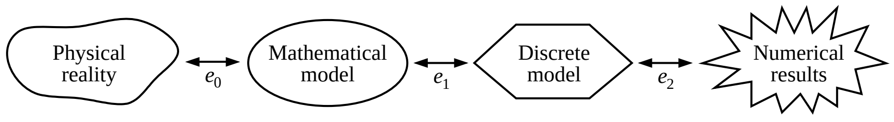

Fig. 1 *General computational scheme: The modeling error e_0 is the difference between physical reality and its mathematical model. The difference between the mathematical model and the discrete model is called the discretization error e_1. Finally, round-off errors are included in e_2.*

[*]Jan Brandts, Korteweg-de Vries Institute for Mathematics, University of Amsterdam, P.O. Box 92248, 1090 GE Amsterdam, The Netherlands, E-mail: `janbrandts@gmail.com`

[†]Michal Křížek – Corresponding author, Institute of Mathematics, Czech Academy of Sciences, Žitná 25, CZ-115 67 Prague 1, Czech Republic, E-mail: `krizek@cesnet.cz`

[‡]Zhimin Zhang, Beijing Computational Science Research Center, Beijing 100094, China, E-mail: `zmzhang@csrc.ac.cn`; Department of Mathematics, Wayne State University, 1131 Faculty/Administration Bldg., 656 West Kirby, Detroit, MI 48202, U.S.A., E-mail: `zzhang@math.wayne.edu`

We should never identify any mathematical model with reality, since no equation describes physical phenomena in our world exactly. This leads to the so-called *modeling error* e_0.

Mathematical models are usually expressed as infinite dimensional problems. They are given by ordinary or partial differential equations, integro-differential or integral equations, systems of these equations, variational inequalities, systems of differential-algebraic equations, and so on. To implement such models on the computer we have to approximate them by finite dimensional problems, which yields the error e_1 (*discretization error*). This error includes for instance the error of numerical integration, and the error of approximation of the boundary of the examined region.

Finally, the error e_2 arises during a computation in the discrete model. It contains, of course, rounding errors, but may include other errors (like iteration errors).

In this paper we draw attention to hidden dangers that may appear in mechanical use of numerical calculations without any knowledge of theory. There is a large amount of other works on similar topics — see e.g. [4,7–9,18,19]. Below we present the most drastic examples. Consider, for instance, the integral

$$I_n = \frac{1}{\mathrm{e}} \int_0^1 x^n \mathrm{e}^x \,\mathrm{d}x > 0.$$

Using integration by parts, we obtain the following recurrence relation [15, p. 505]

$$I_n = 1 - nI_{n-1}, \quad n = 1, 2, \dots, \tag{1}$$

where $I_0 = 1 - \mathrm{e}^{-1}$. Renata Babušková in her 1964 paper [2] examines its numerical instability. After a few steps of the above recurrence we surprisingly get negative values in single-precision arithmetic (4 bytes) even though I_n is positive for all n. Running the iteration in double-precision arithmetic (indicated by $\tilde{}$ throughout this paper) on a standard PC, we observe a decreasing series until $\tilde{I}_{16} = 0.0374$, and then $\tilde{I}_{17} = 0.3259$, $\tilde{I}_{18} = -5.1930$ is negative, $\tilde{I}_{19} = 104.8628, \dots$ which leads to an alternating divergent sequence. This numerical phenomenon happens due to the fact that at each step we subtract two numbers of almost the same size. Then the difference contains only a few nonzero significant digits in computer arithmetic that necessarily leads to loss of accuracy. Note that I_n may be calculated via a fast convergent series

$$I_n = \frac{1}{n+1} - \frac{1}{(n+1)(n+2)} + \frac{1}{(n+1)(n+2)(n+3)} - \dots$$

Similar examples are collected in the classical 1966 monograph *Numerical Processes in Differential Equations* [1] by Ivo Babuška, Milan Práger, and Emil Vitásek. For instance, they investigate numerical instability of successive evaluation of the expression

$$\dots(((((1:2)\cdot 2):3)\cdot 3):4)\cdot 4 \dots \tag{2}$$

Their colleague Karel Segeth obtained various results for this expression on different computers involving thousands of divisions and multiplications [1, p. 6]. For examples of large accumulations of rounding errors in calculating infinite sums or limits see also [6, pp. 14–17].

The aim of this paper is to reveal several cautionary examples, which one should have in mind before performing numerical calculations. We also recalculated three shocking examples from a recently published monograph *Handbook of Floating-Point Arithmetic* [12].

2. Can a decreasing sequence be increasing in computer arithmetic?

Set $a_0 = 1$, $a_1 = \frac{1}{11}$, and let

$$a_{n+2} = \frac{34}{11} a_{n+1} - \frac{3}{11} a_n \quad \text{for } n = 0, 1, 2, \ldots \tag{3}$$

The exact solution

$$a_n = \frac{1}{11^n}$$

is clearly a decreasing sequence. However, calculating (3) with MATLAB (Matrix Laboratory) on a standard PC computer, we observe that the associated series decreases until $\tilde{a}_{12} = 0.2068 \cdot 10^{-11}$, then increases and reaches $\tilde{a}_{40} = 40.44$ and grows quite rapidly.

This example demonstrates that subtraction of two numbers of almost the same size is not a good practice in scientific computing. Below we list some data for (3) with different precision arithmetic,

$\underline{a}_{50} \approx 2 \cdot 10^{11}$	in single-precision arithmetic (4 bytes),
$\tilde{a}_{50} \approx 5 \cdot 10^{5}$	in double-precision arithmetic (8 bytes),
$\overline{a}_{50} \approx 10^{3}$	in extended-precision arithmetic (10 bytes).

Moreover, the corresponding sequences $(\underline{a}_n)$, $(\tilde{a}_n)$, and $(\overline{a}_n)$ are increasing from $n = 8, 12$, and 14, respectively. Rounding to two significant digits only yields an increasing sequence from $n = 2$.

It is interesting to observe that if we implement (3) in a slightly different way

$$a_{n+2} = \frac{34 a_{n+1} - 3 a_n}{11},$$

then the resulting sequence turns to be negative at $\tilde{a}_{12}$ and remains negative afterwards with increasing magnitudes.

3. Babuška's example

Consider the Lebesgue integrable function f on $[0, 1]$ which is zero at all rational numbers and $f = 1$ otherwise. Then obviously

$$\int_0^1 f(x) \mathrm{d}x = 1,$$

whereas any numerical quadrature formula

$$\int_0^1 f(x)\mathrm{d}x \approx \sum_{i=1}^{n} w_i f(x_i)$$

with real weights w_i and nodes x_i from $[0, 1]$ yields a zero approximate value for this integral, since the nodes are approximated by rational numbers (with a finite number of decimal places).

This result, in which the numerical integration error always equals one, is due to the fact that f is not a smooth function. Some regularity of f is usually required in standard theorems on convergence of quadrature formulas.

4. Kahan's example

Set $a_0 = 2$, $a_1 = -4$, and consider the recurrence (see [12])

$$a_n = 111 - \frac{1130}{a_{n-1}} + \frac{3000}{a_{n-1}a_{n-2}} \quad \text{for } n = 2, 3, \dots \tag{4}$$

Its general solution has the form

$$a_n = \frac{\alpha 100^{n+1} + \beta 6^{n+1} + \gamma 5^{n+1}}{\alpha 100^n + \beta 6^n + \gamma 5^n},$$

where α, β, and γ depend on the initial values a_0 and a_1.

Thus, if $\alpha \neq 0$ then the limit of the sequence (a_n) is 100. Nevertheless, we observe that if $\alpha = 0$ and $\beta \neq 0$, then the limit is 6. For the initial values $a_0 = 2$ and $a_1 = -4$ we find that

$$a_{20} \doteq 6.034, \quad a_{30} \doteq 6.006.$$

However, calculating (4) in double precision, we get

$$\tilde{a}_{20} \doteq 98.351, \quad \tilde{a}_{30} \doteq 99.999$$

due to rounding errors (cf. also [6, p. 45]).

5. Muller's example

Let $a_1 = \mathrm{e} - 1 = 1.718281828\dots$ and consider the thoroughly innocent-looking sequence (see [12])

$$a_n = n(a_{n-1} - 1) \quad \text{for } n = 2, 3, \dots \tag{5}$$

Notice that this recurrence is only a slight modification of (1). By induction we can easily prove that

$$a_n = n!\Big(\frac{1}{n!} + \frac{1}{(n+1)!} + \frac{1}{(n+2)!} + \cdots\Big). \tag{6}$$

Indeed, for $n = 1$ we have $a_1 = \mathrm{e} - 1$ and assuming the validity of (6) for $n - 1 \geq 0$, we get by (5)

$$\begin{aligned} a_n &= n(a_{n-1} - 1) = n\Big((n-1)!\Big(\frac{1}{(n-1)!} + \frac{1}{n!} + \frac{1}{(n+1)!} + \cdots\Big) - \frac{(n-1)!}{(n-1)!}\Big) \\ &= n!\Big(\frac{1}{n!} + \frac{1}{(n+1)!} + \frac{1}{(n+2)!} + \cdots\Big). \end{aligned}$$

From (6) we immediately see that the sequence (a_n) is decreasing, all a_n have a very reasonable size in the interval $(1, 2)$, and

$$\lim_{n\to\infty} a_n = 1.$$

However, if we run the iteration on MATLAB, we observe that the sequence decreases until $\tilde{a}_{15} = 1.0668$, and then a strange thing happens: $\tilde{a}_{16} = 1.0685$, $\tilde{a}_{17} = 1.1652$, $\tilde{a}_{18} = 2.9731$, $\tilde{a}_{19} = 37.4882$, $\tilde{a}_{15} = 729.7637, \dots$ The sequence grows out of control.

Performing only 24 subtractions and 24 multiplications in (5) with different numbers of significant digits, we obtain the following table:

$\underline{a}_{25} \approx -6.204484 \cdot 10^{23}$	in single-precision arithmetic (4 bytes),
$\tilde{a}_{25} \approx 1.201807248 \cdot 10^{9}$	in double-precision arithmetic (8 bytes),
$\overline{a}_{25} \approx -7.3557319606 \cdot 10^{6}$	in extended-precision arithmetic (10 bytes).

However, by (6) we have

$$1.038 < 1 + \frac{1}{26} < a_{25} < 1 + \frac{1}{26} + \frac{1}{26^2} + \cdots = \frac{26}{25} = 1.04,$$

where the standard formula for the sum of a geometric sequence was applied. Now, let us extend the number of significant digits. In arithmetic with D decimal digits, the first twelve significant digits are:

D	$a_{25}(D)$
20	615990.413139,
21	$-$ 4457.98859386,
22	$-$ 4457.98859386,
23	195.374419140,
24	40.2623187072,
25	$-$ 6.27131142281,
26	1.48429359885.

The values corresponding $D = 21$ and $D = 22$ are the same, since the 21st decimal digit of the Euler number e= 2.71828182845904523536028745... is equal to zero. Only for $D > 25$ do the numerical results start to resemble the exact value $a_{25} = 1.03993872967\dots$ For instance, if $D = 30$ we get $a_{25}(30) = 1.039897\dots$

Now let us take a closer look at the main reason of this strange behavior. Denote by ε_i the rounding error at the ith step. Then we have

$$\begin{aligned}
\tilde{a}_1 &= \mathrm{e} - 1 + \varepsilon_1 = a_1 + \varepsilon_1,\\
\tilde{a}_2 &= 2(\tilde{a}_1 - 1) + \varepsilon_2 = 2(a_1 + \varepsilon_1 - 1) + \varepsilon_2 = a_2 + 2!\varepsilon_1 + \varepsilon_2,\\
\tilde{a}_3 &= 3(\tilde{a}_2 - 1) + \varepsilon_3 = 3(a_2 + 2\varepsilon_1 + \varepsilon_2 - 1) + \varepsilon_3 = a_3 + 3!\varepsilon_1 + 3\varepsilon_2 + \varepsilon_3,\\
&\;\vdots\\
\tilde{a}_{25} &= 25(\tilde{a}_{24} - 1) + \varepsilon_{25} = a_{25} + 25!\varepsilon_1 + \frac{25!}{2!}\varepsilon_2 + \frac{25!}{3!}\varepsilon_3 + \cdots + \varepsilon_{25}.
\end{aligned}$$

Therefore, the total rounding error is

$$a_{25} - \tilde{a}_{25} = -25!\Big(\varepsilon_1 + \frac{1}{2!}\varepsilon_2 + \frac{1}{3!}\varepsilon_3 + \cdots + \frac{1}{25!}\varepsilon_{25}\Big) \tag{7}$$

and its size depends particularly on the several initial rounding errors ε_1, $\varepsilon_2, \ldots$

This sophisticated example shows why it is necessary to try to avoid the subtraction of two numbers that are almost equal, which happens for instance in long-term numerical simulations of the N-body problem (see (15) below). We observe from (7) why the above recurrence (5) is so sensitive to round-off errors. Computer arithmetic with variable length does not improve this numerical effect for large n. An interesting application of the sequence (5) in banking can be found in [12] and [13].

6. Rump's example

The use of a much smaller number of subtractions than in the previous example may also lead to absolutely catastrophic numerical results. Evaluate the rational function

$$u(x, y) = 333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2) + 5.5y^8 + \frac{x}{2y} \tag{8}$$

at $x = 77617.0$ and $y = 33096.0$. Note that this is a polynomial of degree eight plus a simple rational function $x/(2y)$. We observe that no recurrence relation as in (5) is evaluated and we perform only three subtractions and a few other arithmetic operations. Contrary to the previous example, we get almost the same numbers:

$\underline{u}(x, y) = 1.172603$ in single-precision arithmetic (4 bytes),
$\tilde{u}(x, y) = 1.1726039400531$ in double-precision arithmetic (8 bytes),
$\overline{u}(x, y) = 1.172603940053178$ in extended-precision arithmetic (10 bytes)

on an outmoded IMB 370 computer (see Rump [16]). The programming package MAPLE, with

$$D = 7, 8, 9, 10, 12, 18, 26, 27$$

decimal digits produces very similar results. However, we should not rejoice over the above results, since the exact value is

$$u(x, y) = -0.827396\ldots \text{ NEGATIVE!}$$

As discussed in [3], for $z = 333.75y^6 + x^2(11x^2y^2 - y^6 - 121y^4 - 2)$ and $w = 5.5y^8$ we can evaluate exactly that

$$\begin{aligned} z &= -7917111340668961361101134701524942850, \\ w &= 7917111340668961361101134701524942848, \end{aligned}$$

which have 35 out of 37 digits in common. Thus we are dealing with huge numbers with insufficient floating points. The exact value is

$$u(x,y) = z + w + \frac{x}{2y} = -2 + \frac{x}{2y} = -\frac{54767}{66192} = -0.827396\ldots$$

Numerical results by MAPLE only begin to be realistic starting at $D = 37, 38, \ldots$ The following MATLAB code:

```
x = 77617.0; y = 33096.0;
y2 = y * y; y4 = y2 * y2; y6 = y4 * y2; y8 = y4 * y4; x2 = x * x;
ff = 11 * x2 * y2 - y6 - 121 * y4 - 2;
u = 333.75 * y6 + x2 * ff + 5.5 * y8 + x/y/2
```

produces $\tilde{u}(x,y) = -1.1806\text{E} + 021$, a completely false result!

If we use D decimal digits in the following MATLAB code:

```
digits(D);
y = vpa(33096, D);
x = vpa(77617, D);
u = vpa(333.75 * y^6 + x^2 * (11 * x^2 * y^2 - y^6 - 121 * y^4 - 2) + 5.5 * y^8 + x/y/2, D)
```

we obtain

$$\begin{aligned} D &= 20, \quad u_D(x,y) = 1073741825.1726039401, \\ D &= 21, \quad u_D(x,y) = 1.17260394005317863186, \\ D &= 28, \quad u_D(x,y) = 1.1726039400531786318588349045, \\ D &= 29, \quad u_D(x,y) = -0.82739605994682136814116509548, \\ D &= 37, \quad u_D(x,y) = -0.8273960599468213681411650954798162920, \end{aligned}$$

where the subscript D stands for the use of computer arithmetic with D decimal digits. We see that when D =21 to 28, the result is close to Rump's 1988 data, and the "correct" answer appears starting from D=29. This example shows that the arithmetic of large numbers should be performed very cautiously in scientific computing. A detailed numerical analysis of this catastrophic behavior of rounding errors is presented in [3] and [10].

Thus, we should keep in mind that a very small number of subtractions and roundings (see (5) and (8)) may completely destroy the exact solution [17]. Examples showing deterministic chaos are sometimes wrongly understood, since rounding errors were not taken into account. In many of these examples chaos appears just due to rounding errors.

7. Gram-Schmidt versus Modified Gram-Schmidt

Let $\mathbf{a_1}, \mathbf{a_2}, \mathbf{a_3} \in \mathbb{R}^n$ be linearly independent. We orthonormalize them by the Gram-Schmidt (GS) process [5] and the Modified Gram-Schmidt (MGS) process. The first two vectors resulting from GS and MGS are the same, being

$$\mathbf{q}_1 = \frac{\mathbf{a}_1}{\|\mathbf{a}_1\|}$$

and

$$\mathbf{q}_2 = \frac{\hat{\mathbf{q}}_2}{\|\hat{\mathbf{q}}_2\|}, \quad \text{where} \quad \hat{\mathbf{q}}_2 = \mathbf{a}_2 - \mathbf{q}_1\mathbf{q}_1^\top \mathbf{a}_2.$$

Now, the difference between GS and MGS is the computation of the third vector, which in the respective cases is computed as follows,

$$\mathbf{q}_3^{\mathrm{GS}} = \frac{\hat{\mathbf{q}}_3^{\mathrm{GS}}}{\|\hat{\mathbf{q}}_3^{\mathrm{GS}}\|}, \quad \text{where} \quad \hat{\mathbf{q}}_3^{\mathrm{GS}} = \mathbf{a}_3 - \mathbf{q}_1\mathbf{q}_1^\top \mathbf{a}_3 - \mathbf{q}_2\mathbf{q}_2^\top \mathbf{a}_3,$$

and

$$\mathbf{q}_3^{\mathrm{MGS}} = \frac{\hat{\mathbf{q}}_3^{\mathrm{MGS}}}{\|\hat{\mathbf{q}}_3^{\mathrm{MGS}}\|}, \quad \text{where} \quad \hat{\mathbf{q}}_3^{\mathrm{MGS}} = (\mathbf{I} - \mathbf{q}_2\mathbf{q}_2^\top)(\mathbf{I} - \mathbf{q}_1\mathbf{q}_1^\top)\mathbf{a}_3.$$

Note that in exact arithmetic, both vectors are the same. The fundamental difference between MGS and GS is, that in MGS, the vector $\mathbf{a}_3$ is orthogonalized against $\mathbf{q}_1$, after which the *orthogonalized result* is orthogonalized against $\mathbf{q}_2$.

To make the difference better visible, both vectors $\hat{\mathbf{q}}_3^{\mathrm{GS}}$ and $\hat{\mathbf{q}}_3^{\mathrm{MGS}}$ can be computed in two consecutive steps as follows,

$$\hat{\hat{\mathbf{q}}}_3^{\mathrm{GS}} = \mathbf{a}_3 - \mathbf{q}_1(\mathbf{q}_1^\top \mathbf{a}_3), \quad \hat{\mathbf{q}}_3^{\mathrm{GS}} = \hat{\hat{\mathbf{q}}}_3^{\mathrm{GS}} - \mathbf{q}_2(\mathbf{q}_2^\top \mathbf{a}_3), \tag{9}$$

and

$$\hat{\hat{\mathbf{q}}}_3^{\mathrm{MGS}} = \mathbf{a}_3 - \mathbf{q}_1(\mathbf{q}_1^\top \mathbf{a}_3), \quad \hat{\mathbf{q}}_3^{\mathrm{MGS}} = \hat{\hat{\mathbf{q}}}_3^{\mathrm{MGS}} - \mathbf{q}_2(\mathbf{q}_2^\top \hat{\hat{\mathbf{q}}}_3^{\mathrm{MGS}}). \tag{10}$$

In further steps of GS and MGS, the difference between the two methods is similar.

In a practical implementation of MGS, as soon as $\mathbf{q}_1$ is computed, all further vectors $\mathbf{a}_2$, $\mathbf{a}_3$, $\mathbf{a}_4, \ldots$ are orthogonalized against $\mathbf{q}_1$. Then, as soon as $\mathbf{q}_2$ is computed, all the vectors that resulted from the orthogonalization against $\mathbf{q}_1$, are orthogonalized against $\mathbf{q}_2$, and so on.

Example. We shall present the difference between GS and MGS in finite precision arithmetic. Let ε be a small number with the property that $1 + \varepsilon^2$ is rounded to 1 on the machine, whereas ε itself is a machine number. Let

$$\mathbf{a}_1 = \begin{bmatrix} 1 \\ \varepsilon \\ \varepsilon \end{bmatrix}, \quad \mathbf{a}_2 = \begin{bmatrix} 1 \\ \varepsilon \\ 0 \end{bmatrix}, \quad \mathbf{a}_3 = \begin{bmatrix} 1 \\ 0 \\ \varepsilon \end{bmatrix}$$

be the vectors to be orthonormalized by GS and MGS. Then both GS and MGS compute the same vectors $\mathbf{q}_1$ and $\mathbf{q}_2$ as follows,

$$\|\mathbf{a}_1\| = \sqrt{1 + 2\varepsilon^2} \sim 1,$$

hence $\mathbf{q}_1 = \mathbf{a}_1$. Next, in order to compute $\hat{\mathbf{q}}_2$, we evaluate the inner product

$$\mathbf{q}_1^\top \mathbf{a}_2 = 1 + \varepsilon^2 \sim 1,$$

and thus,

$$\hat{\mathbf{q}}_2 = \mathbf{a}_2 - 1 \cdot \mathbf{q}_1 = \begin{bmatrix} 0 \\ 0 \\ -\varepsilon \end{bmatrix}.$$

Normalization can be done exactly, and thus

$$\mathbf{q}_2 = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}.$$

We use the two-step formula (9) above to compute $\hat{\hat{\mathbf{q}}}_3^{\mathrm{GS}}$ as follows:

$$\hat{\hat{\mathbf{q}}}_3^{\mathrm{GS}} = \begin{bmatrix} 1 \\ 0 \\ \varepsilon \end{bmatrix} - \begin{bmatrix} 1 \\ \varepsilon \\ \varepsilon \end{bmatrix} [1\ \varepsilon\ \varepsilon] \begin{bmatrix} 1 \\ 0 \\ \varepsilon \end{bmatrix} \sim \begin{bmatrix} 1 \\ 0 \\ \varepsilon \end{bmatrix} - \begin{bmatrix} 1 \\ \varepsilon \\ \varepsilon \end{bmatrix} = \begin{bmatrix} 0 \\ -\varepsilon \\ 0 \end{bmatrix},$$

followed by

$$\hat{\mathbf{q}}_3^{\mathrm{GS}} = \begin{bmatrix} 0 \\ -\varepsilon \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} [0\ 0\ -1] \begin{bmatrix} 1 \\ 0 \\ \varepsilon \end{bmatrix} \sim \begin{bmatrix} 0 \\ -\varepsilon \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ \varepsilon \end{bmatrix} = \begin{bmatrix} 0 \\ -\varepsilon \\ -\varepsilon \end{bmatrix},$$

after which normalization gives the final result

$$\mathbf{q}_3^{\mathrm{GS}} = \begin{bmatrix} 0 \\ -\frac{1}{2}\sqrt{2} \\ -\frac{1}{2}\sqrt{2} \end{bmatrix}.$$

Summarizing, the three vectors obtained by GS are

$$\mathbf{q}_1 = \begin{bmatrix} 1 \\ \varepsilon \\ \varepsilon \end{bmatrix}, \quad \mathbf{q}_2 = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}, \quad \text{and} \quad \mathbf{q}_3^{\mathrm{GS}} = \begin{bmatrix} 0 \\ -\frac{1}{2}\sqrt{2} \\ -\frac{1}{2}\sqrt{2} \end{bmatrix}, \tag{11}$$

and this is obviously far from an orthonormal basis. We observe that the third vector $\mathbf{q}_3^{\mathrm{GS}}$ sweeps approximately the angle 45° with the plane given by $\mathbf{q}_1$ and $\mathbf{q}_2$.

On the other hand, if we use the two-step formula (10) to compute the third vector using MGS, we find

$$\hat{\hat{\mathbf{q}}}_3^{\mathrm{MGS}} = \hat{\hat{\mathbf{q}}}_3^{\mathrm{GS}} = \begin{bmatrix} 0 \\ -\varepsilon \\ 0 \end{bmatrix},$$

and now the fundamentally different next step,

$$\hat{\mathbf{q}}_3^{\mathrm{MGS}} = \begin{bmatrix} 0 \\ -\varepsilon \\ 0 \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} [0\ 0\ -1] \begin{bmatrix} 0 \\ -\varepsilon \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ -\varepsilon \\ 0 \end{bmatrix},$$

resulting after normalization in

$$\mathbf{q}_3^{\mathrm{MGS}} = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}.$$

Thus, as the final result the three vectors

$$\mathbf{q}_1 = \begin{bmatrix} 1 \\ \varepsilon \\ \varepsilon \end{bmatrix}, \quad \mathbf{q}_2 = \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix}, \quad \text{and} \quad \mathbf{q}_3^{\mathrm{MGS}} = \begin{bmatrix} 0 \\ -1 \\ 0 \end{bmatrix}.$$

In spite of the fact that these three vectors are also not orthonormal, they are much closer to orthonormal than the three vectors (11) resulting from GS. Due to different manners of projection of round-off errors, MGS is a very helpful remedy against the instability of GS.

8. Eigenvalues

Consider the 3×3 matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 4 & 2 \\ -1 & 1 & 1 \\ 1 & 2 & 1 \end{bmatrix},$$

whose entries are small integers. Its eigenvalues can be computed by solving the characteristic equation $\det(\mathbf{A} - \lambda\mathbf{I}) = 0$, which reduces by Sarrus' rule to

$$\det(\mathbf{A} - \lambda\mathbf{I}) = \det \begin{bmatrix} 1-\lambda & 4 & 2 \\ -1 & 1-\lambda & 1 \\ 1 & 2 & 1-\lambda \end{bmatrix}$$

$$= (1-\lambda)^3 + 4 - 4 + 4(1-\lambda) - 2(1-\lambda) - 2(1-\lambda) = (1-\lambda)^3 = 0.$$

Thus, $\mathbf{A}$ has all three eigenvalues equal to one, just like the identity matrix. However, when Matlab is used to compute these eigenvalues using the command `eig(A)` the resulting values are

$$0.9999934375759070 \quad \text{and} \quad 1.000003281212047 \pm 5.683168987282289 \cdot 10^{-6}\mathrm{i}.$$

We see that even though the computations were performed in IEEE double precision, not sixteen but only six significant digits are correct. The cause for this is not a bug in Matlab's software, but the inherent ill-conditioning of the eigenvalues of this particular matrix $\mathbf{A}$. Indeed, suppose that a rounding error causes that the roots of $\tilde{p}(x) = (1-\lambda)^3 + \varepsilon$ to be computed instead of those of $p(x) = (1-\lambda)^3$, with $\varepsilon \approx 10^{-16}$ (see also [7, p. 17]). Then those roots are quite close to the numbers computed by Matlab. Thus, Matlab is doing as well as it can, given the mathematical ill-conditioning of the problem.

9. Numerical differentiation

Numerical differentiation represents another very instable operation. To illustrate it, consider the rational function (see [9])

$$g(x) = \frac{4970x - 4923}{4970x^2 - 9799x + 4830}, \quad x \geq 0.99, \tag{12}$$

and calculate its second derivative at the point 1 using the second central differences, i.e.

$$\delta_h^2 g(x) = \frac{g(1+h) - 2g(1) + g(1-h)}{h^2} \tag{13}$$

which tends to $\ddot{g}(1)$ as $h \to 0$. However, from the second row of Tab. I we can only hardly deduce which value of h is the best approximation of $\ddot{g}(1)$ without knowing that the exact value is $\ddot{g}(1) = 94$. For "large" h the function $g(1-h)$ quickly changes. On the other hand, for "small" h total loss of accuracy appears, since we subtract almost the same numbers in the nominator of (13).

$\delta_h^2 g(x)$	$h = 10^{-2}$	$h = 10^{-3}$	$h = 10^{-4}$	$h = 10^{-5}$	$h = 10^{-6}$	$h = 10^{-7}$
(13)	−91769.95	−2250.2	70.94	93.71	116.42	−151345
(14)	−91769.95	−2250.2	70.79	93.77	94.00	94.00

Tab. I *Numerical values of* $\ddot{g}(1)$ *calculated by (13) and (14), respectively.*

The remedy is to rearrange formula (13) by means of (12) as follows:

$$\delta_h^2 g(x) = \frac{94(1 - 70^2 71^2 h^2)}{(1 - 71^2 h^2)(1 - 70^2 h^2)} \tag{14}$$

which produces relatively good numerical results — see the last row of the above Tab. I.

Finally, let us emphasize that calculation of the second derivatives from biased data as done e.g. in [14] is a very ill-conditioned problem.

10. The N-body problem

For an integer $N \geq 2$ consider an isolated system of N mass-point bodies that mutually interact only gravitationally. Let $r_i = r_i(t)$, $i \in \{1, \ldots, N\}$, be the so-called radius-vector in $\mathbb{R}^3$ describing the position of the ith body with mass m_i in time $t \geq 0$. Denoting the gravitational constant by G, the classical N-body problem is described by the following nonlinear system of ordinary differential equations for the unknown trajectories r_i (see [11])

$$\ddot{r}_i = G \sum_{i \neq j}^{N} \frac{m_j (r_j - r_i)}{|r_j - r_i|^3} \tag{15}$$

with some initial conditions at $t_0 = 0$ and over a time interval $[0, T]$ in which it is assumed that bodies do not collide. This system is autonomous, since the right-hand side of (15) does not explicitly depend on time.

Can we believe numerical simulations of the evolution of the Solar system based on (15) for billions of years in the past or future? The answer is NO from several reasons. Newtonian mechanics does not allow any delay given by the finite speed of gravitational interaction. System (15) is only an ordinary differential equation whose solution on the interval $[0, T]$ depends only on the value at point $t_0 = 0$ and not on the history. An extremely small modeling error $\varepsilon > 0$ during one year may yield after 10^9 years an error of order at least $\varepsilon 10^9$ which may be a quite large number. Also various nongravitational forces are not included in (15).

The right-hand side of (15) does not satisfy the famous Caratheodory conditions. Moreover, system (15) is not Lyapunov stable, i.e., extremely small changes of initial conditions or other perturbations cause very large changes in the final state provided the time interval is long enough. This also makes the numerical solution unstable. Hence, the N-body problem should be treated very carefully. The above examples should be a sufficient warning.

If an integration step h gives almost the same numerical results as $h/2$, the integration of the system (15) is usually stopped. However, this heuristic criterion need not work properly. Here we present another way how to check whether our numerical results are good. It is based on backward integration of (15) as sketched on Fig. 2. Let $r = (r_1, \dots, r_N)$ denote a vector with $3N$ entries and let $f = f(r)$ stand for the right-hand hand side of (15).

Theorem. *Let $r = r(t)$ be the unique solution of the system*

$$\ddot{r} = f(r) \tag{16}$$

on the interval $[0, T]$ with given initial conditions

$$r(0) = r_0 \quad \text{and} \quad \dot{r}(0) = v_0. \tag{17}$$

Then the function $s = s(t)$ defined by

$$s(t) = r(T - t) \tag{18}$$

solves the same system (16) with initial conditions $s(0) = r(T)$, $\dot{s}(0) = -\dot{r}(T)$ and we have

$$s(T) = r_0 \quad \text{and} \quad \dot{s}(T) = -v_0. \tag{19}$$

P r o o f. According to (18) and (16), we obtain

$$\ddot{s}(t) = (-\dot{r}(T - t))^{\cdot} = \ddot{r}(T - t) = f(r(T - t)) = f(s(t)).$$

We see that s satisfies the same system (16) like r. For the final conditions by (18) and (17) we get relations (19),

$$s(T) = r(0) = r_0 \quad \text{and} \quad \dot{s}(T) = -\dot{r}(T - T) = -\dot{r}(0) = -v_0. \qquad \square$$

The above theorem can be applied to long-term intervals as follows. Denote by r^* and s^* a numerical solution of the system (15) with initial conditions (18) and

$$s(0) = r^*(T) \quad \text{and} \quad \dot{s}(0) = -\dot{r}^*(T),$$

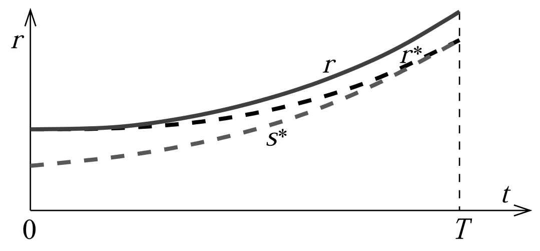

Fig. 2 *Application of the above theorem in numerical solution of the N-body problem. The symbol r stands for the true solution, r^* is the numerical solution, and s^* is the control backward solution.*

respectively. If $\delta > 0$ is a given tolerance and

$$|s^*(T) - r_0| + |\dot{s}^*(T) + v_0| \gg \delta,$$

then it cannot hold $|r(T) - r^*(T)| + |\dot{r}(T) - \dot{r}^*(T)| < \delta$, where r is the exact solution, i.e., the numerical error of the original problem (15) and (18) would be also large, as it is schematically depicted on Fig. 2.

Finally, let us point out that the previous theorem and computational strategy can be easily generalized to evolution partial differential equations.

From (15) we observe that when two bodies are close to each other ($r_j \approx r_i$) which is an important case e.g. in space aeronautics, then we subtract in the denominator two numbers of almost the same size. This may lead to a catastrophic loss of accuracy.

11. Conclusions

In computer arithmetic, only a finite number of significant digits is used. This fact may lead to a catastrophic loss of accuracy, in particular, when numerical schemes are not stable. The aim of this paper was to present several thoroughly innocent-looking examples that produce completely nonsensical results. They are of several types “in limit”:

1) the expression in (2) is of type 1^∞,
2) the right-hand side of (8) is of type $\infty - \infty$,
3) the fraction in (13) is of type $0/0$,
4) the product in (5) is of type $0 \cdot \infty$.

In preforming any real-life calculations in noninteger computer arithmetic, we should keep in mind that we always produce three basic kinds of errors: modeling error, discretization error, and rounding errors. They usually grow exponentially with time. Hence, we should very carefully and critically evaluate the numerical output and should not blindly trust computer results as it is often done.

Acknowledgement

The authors would like to thank Ľubomíra Balková, Jan Chleboun, Edita Pelantová, Lawrence Somer, and Tomáš Vejchodský for useful discussions and extensive calculations. The second author was supported by Grant no. P101/14-02067S and RVO 67985840. The research of the third author was supported in part by the National Natural Science Foundation of China under grants 11471031, 91430216, and the U.S. National Science Foundation through grant DMS-1419040.

References

[1] BABUŠKA I., PRÁGER M., VITÁSEK E. *Numerical processes in differential equations.* John Willey & Sons, London, New York, Sydney, 1966.

[2] BABUŠKOVÁ R. *Über numerische Stabilität einiger Rekursionsformeln.* Apl. Mat. 1964, 9, pp. 186–193.

[3] CUYT A., VERDONK B., BECUWE S., KUTERNA P. *A remarkable example of catastrophic cancellation unraveled.* Computing 2011, 66, pp. 309–320, doi: 10.1007/s006070170028.

[4] FORSYTHE G.E. *Pitfalls in computation, or why a math book isn't enough.* Amer. Math. Monthly 1970, 77, pp. 931–956, doi: 10.2307/2318109.

[5] HAZEWINKEL M. (ed.) *Orthogonalization.* Springer, 2001.

[6] HIGHAM N. J. *Accuracy and stability of numerical algorithms.* SIAM, Philadelphia, 2002.

[7] KŘÍŽEK M., PRÁGER M., VITÁSEK E. *Reliability of numerical calculations.* Pokroky Mat. Fyz. Astronom. 1997, 42(2), pp. 8–23. In Czech.

[8] KŘÍŽEK M., ZHANG Z. *Reliability of numerical calculations.* Math. Culture 2015, 6(1), pp. 34–40. In Chinese.

[9] KULISH U., MIRANKER W. L. *The arithmetic of the digital computer: a new approach.* SIAM Rev. 1986, 28(1), pp. 1–40, doi: 10.1137/1028001.

[10] LOH E., WALSTER G. W. *Rump's example revisited.* Reliable Comput. 2002, 8, pp. 245–248.

[11] MARCHAL C. *The three-body problem.* Elsevier, Amsterdam, 1991.

[12] MULLER J.-M. et al. *Handbook of floating-point arithmetic.* Birkhäuser, 2009.

[13] PELANTOVÁ E., ZNOJIL M. *Can we believe our own calculator?* Rozhledy mat.-fyz. 2010, 85, pp. 11–18. In Czech.

[14] PERLMUTTER S. *Supernovae, dark energy, and the accelerating universe.* Physics Today 2003, 56, April, pp. 53–60, doi: 10.1063/1.1580050.

[15] REKTORYS K. *Survey of applicable mathematics I.* Kluwer Acad. Publ., Dordrecht, 1994.

[16] RUMP S.M. *Algorithms for verified inclusions — theory and practice.* In: Reliability in Computation (ed. R. E. Moore), Academic Press, New York, 1988, pp. 109–126.

[17] RUMP S.M. *Verification methods: Rigorous results using floating-point arithmetic.* Acta Numerica 2010, 19, pp. 287–449, doi: 10.1017/S096249291000005X.

[18] STEGUN I.A., ABRAMOWITZ M. *Pitfalls in computation.* J. Soc. Indust. Appl. Math. 1956, 4(4), pp. 207–219.

[19] WILKINSON J.H. *Rounding errors in algebraic processes.* Prentice-Hall, New York, 1963.