International Journal of Applied Mathematics

Volume 35 No. 1 2022, 181-203
ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)
doi: http://dx.doi.org/10.12732/ijam.v35i1.14

COMPUTATIONAL ERRORS

Mihail Konstantinov[1] §, Petko Petkov[2]

[1]University of Architecture, Civil Engineering and Geodesy
1046 – Sofia, BULGARIA

[2] Technical University of Sofia
1157 – Sofia, BULGARIA

Abstract: Bounds are presented for average rounding errors in double precision binary floating point arithmetic obeying the IEEE Standard [5]. The average errors are up to three times less than the usually used maximal errors and may be more appropriate for practical calculations. For this purpose an alternative representation of machine numbers is proposed. Various numerical computations with unexpectedly large errors are also considered. The computations are done in MATLAB® environment [9].

AMS Subject Classification: 65G50, 97N20
Key Words: rounding errors, average errors, large error computations

1. Introduction and Notation

We define and consider in detail average rounding errors in binary floating point arithmetic (FPA). This arithmetic consists of the set of machine numbers which are binary fractions including zero, and the rules for performing arithmetic operations with these numbers. For definiteness we consider double precision binary FPA. For FPA with different precision and/or base the results are similar to these presented here since they are formulated mainly in terms of the rounding unit of FPA. We also consider large computational errors due to rounding. Some of the results presented had been announced in [6]. A detailed study of the accuracy of numerical algorithms is the monograph [2].

Received: December 18, 2021

§Correspondence author

Let $\mathbb{Z} = \{0, \pm 1, \pm 2, \dots\}$ be the set of integers, $\mathbb{B}$ be the set of binary fractions $p\,2^q$ $(p, q \in \mathbb{Z})$ and $\mathbb{R}$ $(\mathbb{R}_+)$ be the set of real (non-negative real) numbers. For $a, b \in \mathbb{R}$ $(a < b)$ the notation $I(a, b)$ stands for any of the four intervals with end points a and b. Let $\mathbb{M} = \mathbb{M}_- \cup \mathbb{M}_+ \subset \mathbb{B}$ be the set of machine numbers, where $\mathbb{M}_+ \subset \mathbb{R}_+$. We assume that $0 \in \mathbb{M}_+$ and deal with rounding as a mapping from $\mathbb{R}$ to $\mathbb{M}$. The set $\mathbb{M}$ consists of normal and subnormal numbers and the symbols $\pm$`Inf`. The number of elements of a set A is denoted $\mathrm{card}(A)$. The elements of $\mathbb{M}$ are written in some special form, see e.g. relations (1), (4), and (6), (5). The symbol `>>` denotes the start of a MATLAB® command.

2. Another View on Machine Arithmetic

2.1. Normal Machine Numbers

The normal machine numbers (NMN) from $\mathbb{M}_+$ are binary fractions of the form

$$\mu(m, n) = 2^m(1 + n\varepsilon) = 2^{m-52}(2^{52} + n) \ (m, n \in \mathbb{Z}), \tag{1}$$

where $-1022 \le m \le 1023$, $0 \le n \le N - 1$, and

$$\begin{aligned} N &= \mu(52, 0) = 2^{52} = 4,503,599,627,370,496 \simeq 4.5036 \times 10^{15} \\ \varepsilon &= \mu(-52, 0) = 2^{-52} \simeq 2.2204 \times 10^{-16}. \end{aligned} \tag{2}$$

The quantity ε is called machine epsilon of FPA and is denoted in MATLAB® as `eps`. The quantities $\mu = \mu(m, n)$ and

$$\mu^+ = \mu(m, n + 1) = \mu(m, n) + 2^m\varepsilon \ (0 \le n \le N - 2) \tag{3}$$

are two consecutive NMN, e.g. $\mu(0, 0) = 1$ and $\mu(0, 1) = 1 + \varepsilon$. The set of NMN is denoted as $\mathbb{M}_{\mathrm{nor}}$. If we consider $\mu(m, n)$ as formal expression for any $m, n \in \mathbb{Z}$ then for $n = N$ we have $\mu(m, N) = \mu(m + 1, 0)$. Furthermore, we denote by $\mu^- = \mu(m, n - 1) = \mu(m, n) - 2^m\varepsilon$ $(1 \le n \le N - 1)$ the machine number left to μ.

Usually NMN are written in binary positional form

$$\mu(m, n) = 2^m \left(1 + \mathrm{bin}(b_1, b_2, \dots, b_{52})\right), \ b_k \in \{0, 1\}, \tag{4}$$

where

$$\mathrm{bin}(b_1, b_2, \dots, b_{52}) = \sum_{k=1}^{52} b_k 2^{-k} \in [0, 1 - \varepsilon]$$

and $n = 2^{52}\,\mathrm{bin}(b_1, b_2, \ldots, b_{52})$. The presentation (1) of NMN is preferable to (4) in the analysis of rounding errors. The rounding unit of FPA is the quantity

$$\rho = \mu(-53, 0) = 2^{-53} \simeq 1.1102 \times 10^{-16}.$$

We have $\rho = \varepsilon/2$ and

$$\rho^{-1} = 2N = 2^{53} = 9,007,199,254,740,992 \simeq 9.007 \times 10^{15}.$$

For example, $\mu(N, n) = N + n$ $(n = 0, 1, \ldots, N-1)$. In particular $2N-1$, $2N$ and $2N+2$ are three consecutive machine numbers, while $2N+1$ is not a machine number and is rounded to $2N$. Thus all integers from $\mathbb{M}_+$ which are larger than $2N$ are even.

The normal range $\mathbb{N} = [r, R] \subset \mathbb{R}_+$ for FPA consists of the positive real numbers between the machine numbers

$$r = \mu(-1022, 0) = 2^{-1022} \simeq 2.2251 \times 10^{-308} \in \mathbb{M}_+$$

denoted in MATLAB® as `realmin`, and

$$R = \mu(1023, N-1) = 2^{1023}(2-\varepsilon) \simeq 1.7977 \times 10^{308} \in \mathbb{M}_+$$

denoted in MATLAB® as `realmax`. Note that $2^{1024} \notin \mathbb{M}$. The machine numbers from $\mathbb{N}$ are the positive NMN.

2.2. Subnormal Machine Numbers

Positive machine numbers between r and the number

$$\omega = \varepsilon\, r = 2^{-1074} \simeq 4.9407 \times 10^{-324} \in \mathbb{M} \tag{5}$$

are called subnormal [5]. The zero is a machine number which is usually excluded from the set of subnormal numbers. Nevertheless we shall treat 0 as a subnormal number. Therefore the non-negative subnormal machine numbers (SMN) in this paper have the form

$$\xi_n = n\omega \ (n = 0, 1, \ldots, N-1) \tag{6}$$

and the set $\mathbb{M}_{\mathrm{sub}} \subset \mathbb{M}$ of SMN has $\mathrm{card}(\mathbb{M}_{\mathrm{sub}}) = N$ elements. We have $\mathbb{M} = \mathbb{M}_{\mathrm{nor}} \cup \mathbb{M}_{\mathrm{sub}}$ and $\mathbb{M}_{\mathrm{nor}} \cap \mathbb{M}_{\mathrm{sub}} = \emptyset$.

Real numbers which are sufficiently larger than R are rounded in FPA to the symbol `Inf` which is the machine analogue of infinity ∞. Positive numbers

x which are sufficiently smaller than $\xi_1 = \omega$ are rounded to $\xi_0 = 0$. In particular the quantity $\omega/(2-\varepsilon)$ is rounded to ω, while $\omega/2$ is rounded to 0.

Rounding of numbers from the normal range $\mathbb{N}$ of FPA is done with relative error less than ρ. Positive real numbers less than r are rounded with larger relative error reaching up to 1 for numbers close to ω.

3. Rounding Errors in the Normal Range

Let $x \in J_m = [2^m, 2^{m+1}) \subset \mathbb{N}$, where $m \in \mathbb{Z}$ and $-1022 \leq m \leq 1023$. The interval J_m contains N machine numbers $2^m(1+n\varepsilon)$ $(n = 0, 1, \ldots, N-1)$ and there are $2046 = 1023 - (-1022) + 1$ such intervals. Hence

$$\operatorname{card}(\mathbb{M}_{\mathrm{nor}}) = 2046\,N = 9,214,364,837,600,034,816 \simeq 9.2144 \times 10^{18}.$$

Since according to (6) there are N subnormal non-negative numbers, the number of elements of $\mathbb{M}_+$ is

$$\operatorname{card}(\mathbb{M}_+) = 2047\,N = 9,218,868,437,227,405,312 \simeq 9.2189 \times 10^{18}.$$

Since we deal with relative rounding errors then without loss of generality we consider the interval J_0 which contains the machine numbers $x_n := 1 + n\varepsilon \in J_0$ $(n = 0, 1, \ldots, N-1)$, where $x_0 = 1$ and $x_{N-1} = 2 - \varepsilon = 2(1-\rho)$.

Denote by $\operatorname{flo}(x) \in \mathbb{M}_+$ the rounded value of $x \in \mathbb{N}$, and let

$$\operatorname{err}(x) = \frac{|\operatorname{flo}(x) - x|}{x} \tag{7}$$

be the relative error in rounding x to the nearest machine number $\operatorname{flo}(x)$. For $x \in I(a,b)$, where $r \leq a < b \leq R$, the maximal rounding error for FPA obeying the IEEE standard [5] satisfies the bound

$$\sup\{\operatorname{err}(x) \colon x \in I(a,b)\} < \rho. \tag{8}$$

For particular values of a and b, however, the left-hand side of (8) may be considerably smaller than ρ.

The numbers $x_n = 1 + n\varepsilon$ (and only they) are rounded exactly, i.e. $\operatorname{flo}(x_n) = x_n$ and $\operatorname{err}(x_n) = 0$. It may be shown that rounding is performed according to the rule

$$\operatorname{flo}(x) = \begin{cases} 1 & x_0 = 1 \leq x \leq x_0 + \rho \\ x_{2p-1} & x_{2p-1} - \rho < x < x_{2p-1} + \rho \\ x_{2p} & x_{2p} - \rho \leq x \leq x_{2p} + \rho \\ 2 & x_{N-1} + \rho = 2 - \rho \leq x < 2 \end{cases} \tag{9}$$

for $p = 1, 2, \ldots, N/2 - 1$. Note that numbers $x \in [2 - \rho, 2) \subset J_0$ are rounded to $2 \notin J_0$.

Let x_n, x_{n+1} be two consecutive NMN and set

$$y_n = \frac{1}{2}(x_n + x_{n+1}) = 1 + (2n + 1)\rho. \tag{10}$$

Consider the restriction err_n of the function err on the interval $[x_n, x_{n+1}]$. Here $\mathrm{flo}(x)$ is either x_n or x_{n+1}, see (9), and the graph of the function err_n consists of two hyperbolas $1 - x_n/x$, $x_n \le x \le y_n$, and $x_{n+1}/x - 1$, $y_n < x \le x_{n+1}$. Hence the function err_n has maximum

$$e_n = \mathrm{err}_n(y_n) = \frac{\rho}{y_n} = \frac{\rho}{1 + (2n + 1)\,\rho} = \frac{1}{2n + 2N + 1}. \tag{11}$$

The error e_n decreases in n. It is maximal for $n = 0$,

$$e_0 = \mathrm{err}(1 + \rho) = \frac{\rho}{1 + \rho} < \rho \tag{12}$$

and minimal for $n = N - 1$,

$$e_{N-1} = \mathrm{err}(2 - \rho) = \frac{\rho}{2 - \rho} = \frac{\rho}{2} + \mathrm{O}(\rho^2). \tag{13}$$

The relations (9) define an increasing step function $\mathrm{flo}\colon \mathbb{N} \to \mathbb{M}_+$ with jumps of magnitude 2ρ at the points $x = y_n$. In turn, the graph of the piece-wise differentiable function $\mathrm{err}\colon \mathbb{N} \to \mathbb{N}$ resembles a saw.

In view of (12), the analysis of rounding errors is usually based [4] on the estimate $\mathrm{err}(x) < \rho = 2^{-53}$ which is almost achievable near to the left end of the interval $[1, 2)$. However, near to the right end of this interval the estimate for err is rather of the type $\mathrm{err}(x) \le \rho/2 = 2^{-54}$, see (13). Moreover, statistically the behavior of the relative rounding errors may not correspond to the upper bounds e_n. Hence it is useful to introduce estimates for average rather than for maximal errors. Such estimates are inequalities of the form

$$\mathrm{err}(x) \le E = C\rho + \mathrm{O}(\rho^2) \tag{14}$$

for certain positive constants $C < 1$. Two average bounds that we consider below are

$$E_{\max} = \frac{1}{N} \sum_{n=0}^{N-1} e_n \tag{15}$$

and

$$E_{\text{int}} = \int_1^2 \text{err}(x)\,\mathrm{d}x. \tag{16}$$

In what follows we compute the constants C in (14) for the average rounding bounds (15) and (16). In particular we have $C = C_{\max} = \log(2) \simeq 0.6931$ for the bound (15) and $C = C_{\text{int}} = \frac{1}{2}\log(2) \simeq 0.3466$ for the bound (16). Although the exact value of C may not be significant in practical computations, reducing C from its commonly used maximal value $C = 1$ to an average value which is almost three times less (!) is of theoretical interest. The estimates presented below use the inequalities given in Section 14.

4. Average Maximal Error in the Normal Range

Consider first the average bound (15), where the local maximal errors e_n are defined in (11). We have

$$E_{\max} = \frac{E}{N} = 2\rho E,\ E = \sum_{n=0}^{N-1} e_n.$$

In view of (33), Section 14, the following relations hold

$$\begin{aligned} E &= \sum_{n=0}^{N-1} \frac{1}{2n+2N+1} = \sum_{k=N+1}^{2N} \frac{1}{2k-1} \\ &= X_{2N} - X_N = Y_N = \frac{1}{2}\log(2) + \mathrm{O}(\rho^2). \end{aligned}$$

Hence $2\rho E < \rho\log(2) + \mathrm{O}(\rho^3)$ and

$$E_{\max} = C_{\max}\,\rho + \mathrm{O}(\rho^3),\ C_{\max} = \log(2) \simeq 0.6931. \tag{17}$$

5. Average Integral Error in the Normal Range

The average integral rounding error for $a \le x \le b$ is defined as

$$E_{\text{int}}(a,b) = \frac{1}{b-a}\int_a^b \text{err}(x)\,\mathrm{d}x.$$

In our case $a = 1$, $b = 2$ and hence

$$E_{\text{int}} = E_{\text{int}}(1,2) = \int_1^2 \text{err}(x)\,\mathrm{d}x = \sum_{n=0}^{N-1} E_n,$$

where

$$E_n = \int_{x_n}^{x_{n+1}} \text{err}(x)\,\mathrm{d}x \ (n = 0, 1, \ldots, N-1) \tag{18}$$

and $x_N = 2$. Furthermore we have $E_n = F_n + G_n$, where

$$F_n = \int_{x_n}^{y_n} \frac{x - x_n}{x}\,\mathrm{d}x = \rho - x_n \log\left(\frac{y_n}{x_n}\right)$$

and

$$G_n = \int_{y_n}^{x_{n+1}} \frac{x_{n+1} - x}{x}\,\mathrm{d}x = x_{n+1}\log\left(\frac{x_{n+1}}{y_n}\right) - \rho.$$

Setting

$$\rho_n = \frac{\rho}{x_n} = \frac{\rho}{1 + n\varepsilon},$$

we obtain

$$\begin{aligned}\frac{F_n + G_n}{x_n} &= (1 + 2\rho_n)\log(1 + 2\rho_n)) - 2(1 + \rho_n)\log(1 + \rho_n) \\ &= \rho_n^2 - \rho_n^3 + \mathrm{O}\big(\rho_n^4\big) = \frac{\rho^2}{x_n^2} - \frac{\rho^3}{x_n^3} + \mathrm{O}\big(\rho^4\big)\,.\end{aligned}$$

Hence for small ρ we have the estimate

$$\frac{F_n + G_n}{\rho} < \rho_n + \mathrm{O}\big(\rho^3\big)$$

and

$$\frac{1}{\rho}\sum_{n=0}^{N-1}(F_n + G_n) < \sum_{n=0}^{N-1}\rho_n + \mathrm{O}(\rho^2).$$

Next we have

$$\begin{aligned}\sum_{n=0}^{N-1}\rho_n &= \frac{1}{2}\sum_{n=0}^{N-1}\frac{1}{n+N} = \frac{1}{2}(H_{2N-1} - H_{N-1}) \\ &= \frac{1}{2}(\log(2N-1) + \gamma_{2N-1} - \log(N-1) - \gamma_{N-1})\end{aligned}$$

$$< \frac{1}{2}\log\left(\frac{2N-1}{N-1}\right) + \frac{1}{4(2N-1)} - \frac{1}{4N}$$
$$= \frac{1}{2}\log\left(\frac{2-2\rho}{1-2\rho}\right) - \frac{\rho-2\rho^2}{4(1-\rho)} = \frac{1}{2}\log(2) + \frac{1}{4}\rho + \mathrm{O}(\rho^2).$$

Therefore

$$E_{\mathrm{int}} \le C_{\mathrm{int}}\rho + \mathrm{O}(\rho^2),\ C_{\mathrm{int}} := \frac{1}{2}\log(2) \simeq 0.3466. \tag{19}$$

An intuitive way to obtain the estimate (19) is to observe that the integral E_n in (18) is approximately equal to the area $\rho\rho_n$ of the triangle with vertexes $(x_n, 0)$, $(x_{n+1}, 0)$ and (y_n, e_n). This argument yields $E_{\mathrm{int}} \simeq \rho\sum_{n=0}^{N-1} e_n = E\rho < C_{\mathrm{int}}\rho + \mathrm{O}(\rho^3)$ which coincides with (19) within first order terms of ρ.

6. Rounding Errors for Large Numbers

As mentioned above, the number

$$R = \mu(1023, N-1) = 2^{1023}(2-\varepsilon) = 2^{971}(2^{53}-1) \tag{20}$$

(written as `realmax` in MATLAB®) is the maximal machine number in FPA, or the maximal element of $\mathbb{M}$. It may be computed by the commands

`>> 2^1023*(2-eps)` or `>> 2^971*(2^53-1)`.

The previous relative to R machine number is

$$R^- = \mu(1023, N-2) = 2^{1023}(2-2\varepsilon) = 2^{971}(2^{53}-2)$$

while the quantity 2^{1024} is not a machine number and is rounded to the symbol `Inf`. Hence if we try to compute R as

`>> 2^1024-2^971` or `>> 2^1024*(1-eps/2)`

the answer shall be `Inf`.

The quantity

$$S = \mu(969, N-1) = 2^{969}(2-\varepsilon) = 2^{917}(2^{53}-1) \in \mathbb{N}$$

has the following property. The number $R+S$ is rounded to R while the number $R+S^+$ is rounded to Inf, where $S^+ = \mu(970, 0)$ is the machine number right to S. The equivalent commands `>>realmax+2^969*(2-eps)`

and `>> realmax+2^917*(2^53-1)` give answer `realmax`, while the command `>> realmax+2^970` returns `Inf`.

As a summary, real numbers x from the interval $(R, R+S]$ are rounded to $\mathrm{flo}(x) = R$ with a relative error

$$\left|\frac{x - \mathrm{flo}(x)}{x}\right| \le \frac{R+S-R}{R+S} = \frac{S}{R+S} = \frac{\rho}{2+\rho} \simeq \frac{1}{2}\rho. \tag{21}$$

In turn, the average integral rounding error in this interval is

$$\frac{1}{S}\int_R^{R+S} \frac{x-R}{x}\,\mathrm{d}x = \frac{1}{S}\left(S - R\log\left(1+\frac{S}{R}\right)\right) \simeq \frac{S}{2R} = \frac{1}{4}\rho. \tag{22}$$

The interval $(R, R+S] \subset \mathbb{R}_+$ is the supnormal range for FPA.

Finally, real numbers $x > R+S$ are rounded to `Inf` and the relative error in this case may be considered as infinite, or indefinite. The interval $(R+S, \infty) \subset \mathbb{R}_+$ is said to be the *infinity range* for the FPA .

7. Rounding Errors for Small Numbers

A real number $x \ge 0$ with $|x| < r$ is rounded to the closest subnormal machine number $\xi_n = n\omega$ of the form (6) as follows:

$$\mathrm{flo}^*(x) = \begin{cases} \xi_0 = 0 & 0 & \le & x & \le & \omega/2 \\ \xi_{2k-1} & \xi_{2k-1} - \omega/2 & < & x & < & \xi_{2k-1} + \omega/2 \\ \xi_{2k} & \xi_{2k} - \omega/2 & \le & x & \le & \xi_{2k} + \omega/2 \\ r & r - \omega/2 & < & x & < & r \end{cases}. \tag{23}$$

Here $k = 1, 2, \ldots, N/2$ and the value of $\omega = \varepsilon\, r$ is given in (5). Since numbers from the interval $\mathbb{M}_0 = [0, \omega/2] \subset \mathbb{R}_+$, $\omega/2 = 2^{-1075} \simeq 2.4703 \times 10^{-324}$, are rounded to 0 we shall refer to $\mathbb{M}_0$ as the *machine zero*.

It follows from (23) that the maximum δ_n of the relative rounding error

$$\mathrm{err}^*(x) := \left|\frac{\mathrm{flo}^*(x) - x}{x}\right| \tag{24}$$

for $x \in (\xi_n, \xi_{n+1})$ is achieved for $x = \eta_n := (\xi_n + \xi_{n+1})/2$. We have

$$\delta_n = \mathrm{err}^*(\eta_n) = \frac{1}{2n+1}\ (n = 0, 1, \ldots, N-1). \tag{25}$$

8. Average Maximal Error for Small Numbers

It follows from (25) that the maximal relative error δ_n decreases from $\delta_0 = 1$ to

$$\delta_{N-1} = \frac{1}{2N-1} = \frac{\rho}{1-\rho}$$

being always larger than ρ. The mean of the maximal errors is

$$E^*_{\max} = \frac{1}{N}\sum_{n=0}^{N-1}\delta_n = \frac{1}{N}\sum_{k=1}^{N}\frac{1}{2k-1} = \frac{1}{N}X_N = 2\rho X_N,$$

or

$$\begin{aligned} E^*_{\max} &= \rho\left(2\log(2) + \gamma + \log(N) + \mathrm{O}\left(N^{-2}\right)\right) \\ &= C(\rho)\rho + \mathrm{O}(\rho^3), \end{aligned} \tag{26}$$

where

$$C(\rho) = C_1 - \log(\rho),\ C_1 = \log(2) + \gamma \simeq 1.2704. \tag{27}$$

We see that the mean term $C(\rho)\rho$ in the upper bound for $E^*_{\max}$, defined by (26) and (27), is not a linear function of ρ. This is a major difference in the properties of the rounding function in the subnormal and the normal ranges. We stress finally that for FPA with $\rho = 2^{-53}$ we have $C(\rho) \leq 38.0072$ and hence $E^*_{\max} \leq 38\,\rho$.

9. Average Integral Error for Small Numbers

The average integral rounding error for numbers from the subnormal range $[0, r]$ of the FPA is

$$E^*_{\mathrm{int}} := \frac{1}{r}\int_0^r \mathrm{err}^*(x)\mathrm{d}x = \sum_{n=0}^{N-1} E^*_n. \tag{28}$$

Here

$$E^*_n := \frac{1}{r}\int_{\xi_n}^{\xi_{n+1}} \mathrm{err}^*(x)\mathrm{d}x = F^*_n + G^*_n$$

and

$$F^*_n = \frac{1}{r}\int_{\xi_n}^{\eta_n}\left(1 - \frac{\xi_n}{x}\right)\mathrm{d}x = \rho - 2\rho n\log\left(1 + \frac{1}{2n}\right),$$

$$G_n^* = \frac{1}{r}\int_{\eta_n}^{\xi_{n+1}} \left(\frac{\xi_{n+1}}{x} - 1\right) \mathrm{d}x = -\rho + 2\rho(n+1)\log\left(1 + \frac{1}{1+2n}\right).$$

Hence $E_n^* = 2\rho\psi(n)$ and

$$E_{\mathrm{int}}^* = 2\rho\,\Phi(\rho),\ \Phi(\rho) := \sum_{n=0}^{N-1} \psi(n),$$

where

$$\psi(n) := (n+1)\log\left(1 + \frac{1}{1+2n}\right) - n\log\left(1 + \frac{1}{2n}\right).$$

It may be shown that $\psi(n) < (1+2n)^{-1}$ and

$$\Phi(\rho) < \sum_{k=1}^{N} \frac{1}{2k-1} = X_N < X_N'' = \frac{1}{2}C(\rho).$$

Hence

$$E_{\mathrm{int}}^* \le C(\rho)\rho + \mathrm{O}(\rho^3), \tag{29}$$

where the expression $C(\rho)$ is determined by (27). We see that the estimates for the maximal average error and the integral average error for rounding of numbers from the subnormal range of FPA coincide.

10. Summary of Rounding Errors

The above results may be summarized as follows. For this particular standard of machine arithmetic (double precision binary FPA) a rounding function $\mathrm{flo} : \mathbb{R} \to \mathbb{M} \cup \{\pm\mathrm{Inf}\}$ is defined with the following properties, illustrated for the case of non-negative numbers in Table 1.

11. Large Error Computations

11.1. Violation of Arithmetic Laws

The exact value of a number $x \in \mathbb{R}$ and its rounded value $\mathrm{flo}(x) \in \mathbb{M}$ in FPA are different unless $x \in \mathbb{M}$. This is true also for $x \in \mathbb{B}$. At the same time the rounding rule $\mathrm{flo} : \mathbb{R} \to \mathbb{M}$ is not a function in the strict mathematical sense. In particular the rounded value $\mathrm{flo}(x)$ of x depends on two main factors:

Range	Max. err.	Integ. err.	Remarks
Machine zero $[0, \omega/2]$	1	1	Rounding to `0`
Subnormal range $(\omega/2, r)$	$C(\rho)\rho$	$C(\rho)\rho$	$C(\rho) = C_1 - \log(\rho)$ $C_1 = \log(2) + \gamma \simeq 1.2704$
Normal range $[r, R]$	$C_{\max}\,\rho$	$0.5\,C_{\max}\,\rho$	$C_{\max} = \log(2) \simeq 0.6931$
Supnormal range $(R, R+S]$	$0.5\,\rho$	$0.25\,\rho$	Rounding to `realmax`
Infinity range $(R+S, \infty)$	∞	∞	Rounding to `Inf`

Table 1: Average relative rounding errors

1. The way (or the syntax of the code) the quantity x is written.

2. The software and hardware realization of the particular FPA.

The second factor is out of user's control. The user may only hope that the requirements of the IEEE Standard [5] are fulfilled. The first factor, however, may lead to unexpected results. This is due to the fact that in FPA the associative laws for multiplication and addition as well as the distributive law are (or may be) violated as shown below.

Recalling that $\rho = \text{eps}/2$ we have $\text{flo}(1 + \rho + \rho) = 1$ (wrong!) and $\text{flo}(\rho + \rho + 1) = \text{flo}(1 + 2\rho) = 1 + 2\rho$ (correct!). Moreover, $\text{flo}(1 + \rho + \rho + \cdots + \rho) = 1$, where the sum contains any number of terms. This is due to the fact that $\text{flo}(1 + \rho) = 1$ and thus $1 + \rho$ is repeatedly rounded to 1. Next, if x satisfies $1 + \rho < x < 1 + 2\rho$ then $\text{flo}(x)$ should be equal to $1 + 2\rho$ according to the definition of the rounding rule `flo`. This however depends on how the quantity x is written. Let for example $x = 1 + \rho + 2\rho^2$. The rounded value of x should be $\text{flo}(x) = 1 + 2\rho$. But writing x in this way we get the wrong result `>>1+run+2*run^2 = 1`, where run $= \rho$. To round x correctly we should use the expression `>>1+run*(1+2*run)` yielding the answer `1+eps`.

Other interesting rounding effects may be observed for numbers far from the boundaries of the normal range of FPA, e.g. for large numbers that are close to R. The quantity $A = R + 2^{970} = 2^{1024} - 2^{970}$ computed by the command `>> realmax + 2^970` gives `Inf`. At the same time if we try to compute A by the command `>>realmax+2^969+2^969` the answer is `realmax` because $R + 2^{969}$

is twice rounded to R. Thus we shall get `realmax` if we add any number of quantities `2^969` to `realmax`.

Also, `2^1024` is set to `Inf` in FPA and the same is valid for quantities written in the form `2^1024-D` for any $D \in \mathbb{N}$. In particular, the exact value of `realmax` is $R = 2^{1024} - 2^{971}$ but, written in this form in FPA, it will be set to `Inf`. To input the correct value of `realmax` in FPA we must write it in the form `>>2^(1024-m)*(2^m-2^(m-53))` for some $1 \le m \le 1023$. Another example is the calculation `>>2^1024-2^1023-2^1023`. The exact result is zero but written in this form in FPA it will be interpreted as `Inf` and we have the somehow surprising result `Inf = 0`. In turn, the ratio `2^1024/2^1024` and the difference `2^1024-2^1024` are interpreted in FPA as NaN (from Not a Number) instead of 1 and 0, respectively.

Other examples are `>>2^53+1-2^53` and `>>2^53+pi-2^53` which give $1 = 0$ and $\pi = 4$, respectively. The reason is that $2^{53} + 1$ and $2^{53} + \pi$ are not machine numbers and they are rounded to 2^{53} and $2^{53}+4$, respectively. Thus an analogue of the famous Indiana Pi Bill # 246 (1897), stating that $\pi = 3.2$, is "obtained" by rounding in FPA.

11.2. Other Large Error Computations

Sometimes simple computations may be contaminated with large rounding errors. Below we consider an example involving a positive integer ν. For $\nu = 1$ the relative error may reach 1 (or 100%), while for $\nu \ge 2$ this error is practically unlimited. Consider the expression

$$F_\nu(x) = \frac{(1+x)^\nu - \sum_{k=0}^{\nu-1} \binom{\nu}{k} x^k}{x^\nu}, \ x \neq 0$$

which is equal to 1. However, the computed value $\mathrm{flo}(F_\nu(x))$ exposes strange behavior for small x.

Consider first the simplest case $F_1(x) = (1 + x - 1)/x$, $x \neq 0$, which illustrates the rounding process in FPA. It may be coded in MATLAB® as `>>x=-3*eps:eps/1000:3*eps;F1=(1+x-1)./x;` and is plotted by the command `>>plot(x,F1,'b',x,ones(1,length(x)),'r')`. The graph of the function $x \mapsto \mathrm{flo}(F_1(x))$ in the neighborhood of the point $x = 0$ resembles a saw. We should have $F_1(x) = 1$ but in fact $\mathrm{flo}(F_1(x))$ varies from 0 to 2. More precisely, for $x \ge 0$ and $k = 0, 1, 2, \ldots$, we have

$$\mathrm{flo}(F_1(x)) = \begin{cases} 0 & 0 \le x \le \rho \\ (4+4k)\rho\, x^{-1} & (3+4k)\rho \le x \le (5+4k)\rho \\ (2+4k)\rho\, x^{-1} & (1+4k)\rho < x < (3+4k)\rho \end{cases}.$$

Similar expressions are valid for $x < 0$. Note that this behavior of the rounded value $\mathrm{flo}(F_1(x))$ of $F_1(x)$ is inherited by the expression

$$D_h(x_0) = \frac{f(x_0 + h) - f(x_0)}{h}$$

which approximates the derivative $f'(x_0)$ of a differentiable function f at the point x_0 for small steps h. Indeed, for $f(x_0)f'(x_0) \neq 0$ we have

$$D_h(x_0) \simeq f'(x_0)F_1(x),\ x = h\frac{f'(x_0)}{f(x_0)}.$$

Weird behavior demonstrates the expression $\mathrm{flo}(F_\nu(x))$ for $\nu \geq 2$ whose values may differ substantially from the exact value $F_\nu(x) = 1$. In particular we have the awful result $\mathrm{flo}(F_\nu(x)) = -\nu/x$, $-\rho/2 \leq x \leq \rho$, $x \neq 0$, with practically unlimited relative error (for $0 < x \leq \rho$ the sign of the computed result is also wrong). For example $\mathrm{flo}(F_2(\rho)) = -2/\rho = -2^{54} \simeq -1.8014 \times 10^{16}$ instead of $F_2(\rho) = 1$. In addition, the computed value of $F_\nu(x)$ strongly depends on the order of summation in the corresponding formula. In particular, for $\nu = 2$ the three expressions

```
>>y1=((1+x).^2-1-2*x)./x.^2,
>>y2=((1+x).^2-2*x-1)./x.^2
```

and

```
>>y3=((1+x).^2-(1+2*x))./x.^2
```

for computing $F_2(x)$ are theoretically equivalent but in FPA they produce quite different results for small values of x, see Table 2.

In order to avoid such catastrophic computations, attempts to "cheat" the FPA may be applied by preliminary symbolic simplification of the corresponding formulas. For example, the simplification of $F_\nu(x)$ directly gives the answer $F_\nu(x) = 1$, see e.g. [8]. If however similar expressions arise in the execution of complicated computational algorithms this approach may not be applicable.

11.3. Subtraction without Cancellation

The machine subtraction of binary fractions is done either exactly or with rounding errors. Usually subtraction $b - a$ of close numbers $a, b \in \mathbb{B}$ $(a, b > 0)$ is considered as a dangerous operation due to possible cancellation of information coded in left most digits of the operands a, b. Rounding is often neglected as

x	y_1	y_2	y_3
$2^{26} \times \varepsilon$	1	1	1
$2^{25.5} \times \varepsilon$	2.751	2.000	2.000
$2^{25} \times \varepsilon$	0	0	0
$\varepsilon/2$	-1.801×10^{16}	-1.801×10^{16}	-1.801×10^{16}
$\varepsilon/3$	-2.702×10^{16}	-2.027×10^{16}	-4.053×10^{16}
$\varepsilon/4$	-3.603×10^{16}	-3.603×10^{16}	0
$\varepsilon/5$	-4.504×10^{16}	-5.630×10^{16}	0
$\varepsilon/8$	-7.206×10^{16}	0	0
2^{-537}	-8.998×10^{161}	0	0
2^{-538}	$-\mathrm{Inf}$	NaN	NaN

Table 2: Unlimited chaotic errors

intermediate reason for loss of accuracy since the machine subtraction of close *machine numbers* $a, b \in \mathbb{M}_+$ is done exactly. But if at least one of the operands is not a machine number then the relative error of subtraction may be very large even if there is no cancellation. This fact is not well known.

Setting $x^* = \mathrm{flo}(x) \in \mathbb{M}_+$ the machine subtraction of $a, b \in \mathbb{N} \cap \mathbb{B}$ is done as $\mathrm{flo}(b-a) = (b^* - a^*)^*$. Let $a = 1+\rho$ and $b = 1+\rho(1+\rho(1+2\rho))$. The exact result is $b - a = \rho^2(1+2\rho)$. We have $a^* = 1$, $b^* = 1+2\rho$ and

$$\mathrm{flo}(b-a) = b^* - a^* = 2\rho = (b-a)\left(\frac{2}{\rho} + \mathrm{O}(1)\right).$$

The computed difference is $2^{54} \simeq 1.8014 \times 10^{16}$ times larger than the exact difference and this is not a result of cancellation! This brutal example is slightly artificial. More common example is the subtraction of close numbers a, b belonging to the set of attraction of a given machine number c. Here $a^* = b^* = c$ and the computed difference is $\mathrm{flo}(b-a) = 0$ with "only" 100% relative error.

11.4. Codes Computing Maximal Element

The MATLAB® code `>>[x_max,p]=max(x)` computes the maximal element $x_{\max}$ of the vector x with elements $x(k)$ so that $x(p) = x_{\max}$. In case of equal maximal elements p is the minimal index with this property. Similar is the action of the code `>>[x_min,q]=min(x)` for computing the minimal element $x_{\min}$ of the vector x and its index q with $x(q) = x_{\min}$.

If the vector x is computed in FPA with rounding unit ρ then some of its elements which should be equal may in fact differ within quantities of order ρ. For example, the vector $x = [4/3 - 1, 1/3]$ has elements equal to $1/3$ and the indexes p, q should be equal to 1. But the command `>>[x_max,p]=max(x)` gives `x_max=0.3333` and `p=2`. The reason is that the first element $x(1) = 4/3 - 1$ of x is rounded to $\mathrm{flo}(4/3 - 1) \simeq 1/3 - \mathrm{eps}/3$ which is less than the rounded value $\mathrm{flo}(1/3) \simeq 1/3 - \mathrm{eps}/12$ of the second element $x(2) = 1/3$. As a result MATLAB® "decides" that the index of the maximal element is 2 instead of 1.

This phenomenon is not well understood. It may lead to errors with large political effect in automatic voting calculations using the method of the largest remainder [6]. There is a simple way to overcome this potential troublemaker: one has to use integer rather than fractional remainders in the voting calculations as proposed in [6].

11.5. Evaluation of Lipschitz Functions

Consider the computational problem

$$y = f(x) \neq 0,\ x \neq 0$$

and let $\mathrm{flo}(x)$ be the rounded value of x such that $|x - \mathrm{flo}(x)| < \rho|x|$. Let the function f be Lipschitz continuous in a neighborhood of the argument x, i.e.

$$|f(x_1) - f(x_2)| \leq L|x_1 - x_2|,$$

where $L = L(\beta) > 0$ is the Lipschitz constant of f, $|x - x_k| \leq \beta$.

When a stable computational algorithm [11, 10, 4] is used to evaluate $f(x)$ then the computed solution $\widetilde{y}$ is near to the solution $f(\widetilde{x})$ of a close problem in the sense that

$$|\widetilde{y} - f(\widetilde{x})| \leq \rho A|y|,\ |\widetilde{x} - x| \leq \rho B|x|\ (A, B \geq 0).$$

We have

$$\begin{aligned} |\widetilde{y} - y| &= |\widetilde{y} - f(\widetilde{x}) + f(\widetilde{x}) - f(x)| \\ &\leq |\widetilde{y} - f(\widetilde{x})| + |f(\widetilde{x}) - f(x)| \\ &\leq \rho A|y| + L|\widetilde{x} - x| \\ &\leq \rho A|y| + \rho LB|x| \end{aligned}$$

Hence the relative computational error satisfies the bound

$$\left|\frac{\widetilde{y} - y}{y}\right| \leq K\rho,\ K = A + BL\frac{|x|}{|y|}, \tag{30}$$

where K is the relative condition number of the problem $y = f(x)$. This remarkable formula shows the three main factors determining the accuracy of computations as follows:

1. The sensitivity of the computational problem relative to perturbations in the data measured by the Lipschitz constant L.

2. The stability properties of the computational algorithm expressed by the constants A and B.

3. The computational environment characterized by the rounding unit ρ and the requirement that the intermediate computational results belong to the normal range of FPA.

In old times the requirement that computational results remain in the normal range of FPA was known as "avoiding over- and underflows". This approach to take into account the errors in the performance of computational algorithms had been proposed in [11, 10, 4].

How the bound (30) is used? For a number of problems the Lipshitz constant L may be calculated or estimated. Also, the value of ρ is usually known or may be estimated easily (in our case $\rho = 2^{-53} \simeq 10^{-16}$). To estimate the constants A, B, however, may be a problem. In this case the heuristic assumption $A = 0$ and $B = 1$ may be used which gives $K = L|x|/|y|$. The computational problem is well conditioned if $K\rho \ll 1$ and in this case we may expect about $-\log_{10}(K\rho)$ true decimal digits in the computed solution $\widetilde{y}$. When $K\rho$ is close to 1 the problem is poorly conditioned and there may be no true digits in the computed solution. Conditioning of computational problems is considered in more detail in [4]. The above considerations lead to the following fundamental rule in numerical computations:

If the data x and/or some of the intermediate results in the computation of $f(x)$ are large and/or the result y is small then large relative errors in the computed solution $\widetilde{y}$ are to be expected.

The famous cancellation argument in the subtraction of close numbers of equal sign is a particular case of this fundamental rule. As an illustration, we have $\mathrm{flo}(2^{53} + 1 - 2^{53}) = 0$ instead of 1 with a 100% percent relative error. The reason is that the data of order $2^{53} \simeq 10^{16}$ is large relative to the result equal to 1.

There are several computational tricks to avoid calculations with very large or very small arguments. For example, the evaluation of T-periodic functions

for large arguments may be done by shifting the argument modulo T. A simple case is $\sin(\pi/2 + 2m\pi) = \sin(\pi/2) = 1$ for m being a positive integer.

Next, the computation of $y = \sqrt{(x_1^2 + x_2^2)}$ gives `Inf` for large arguments $|x_k| \geq 10^{155}$ and 0 for small arguments $|x_k| \leq 10^{-155}$ even if x_k and y are in the normal range of FPA. To avoid such numerical disasters we may scale the arguments as $\xi_k = x_k/s$, $s = |x_1| + |x_2|$, and compute the result from $y = s\eta$, where $\eta = \sqrt{\xi_1^2 + \xi_2^2}$.

The calculation of the matrix exponential $\exp(A)$ for a matrix argument A either by Padé approximation or by truncated Taylor series requires calculation of matrix powers which may be accompanied with large errors. This calculation may be done reliably scaling the matrix argument A by powers of 2 until the norm of the scaled matrix becomes less than 1, see e.g. [3].

Some problems may require particular tricks to avoid large computational errors. The calculation of the scalar $y = \exp(a)$ by the Taylor formula may be a catastrophe if $|a|$ is large and $a < 0$. To avoid this we may use directly the Taylor formula if $a > 0$ and the expression $y = 1/\exp(|a|)$ if $a < 0$.

A popular way to avoid cancellation is to reformulate algebraic expressions, e.g. the quantity $y = z-1$, $z = \sqrt{1 + x^2}$, should be computed as $x^2/(z+1)$. This approach may be used to develop closed form solutions of algebraic equations of order up to 4 (if such solutions are ever needed). The list of such tricks may be continued.

11.6. Evaluation of Hölder Functions

Consider again the computational problem $y = f(x)$, where the function f is Hölder continuous, i.e.

$$|f(x_1) - f(x_2)| \leq H|x_1 - x_2|^h.$$

Here the exponent h satisfies $0 < h < 1$. The machine computation of y may be accomplished with large errors, especially when h is close to 0. Such cases arise e.g. in the calculation of multiple zero z of a smooth function F, where $F(z) = F'(z) = \cdots = F^{(k-1)}(z) = 0$, $F^{(k)}(z) \neq 0$ ($k \geq 2$). In this case $h = 1/k$.

Let $\widetilde{y}$ be the computed value of y and consider for simplicity the case when $A = 0$ and $B = 1$. Then we have the bound

$$\frac{|\widetilde{y} - y|}{|y|} \leq M\rho^h, \; M = H\frac{|x|^h}{|y|}.$$

An example of a Hölder problem is the solution of m-th degree algebraic equation with multiple roots, where $m \geq 3$ (for $m = 2$ there are specific reliable

computer codes which find the roots with high accuracy for data from the normal range of FPA). There are three codes (`roots`, `vpasolve` and `fzero`) in MATLAB® which may be used for solving algebraic equations.

1. The code `roots` is fast and works with equations of order up to several thousand but gives large errors for multiple roots.

2. The code `vpasolve` works with very high accuracy with equations of order up to several hundred but may be slow in some cases.

3. The code `fzero` is fast and solves general finite equations. It finds one root at a time. The code does not work properly in case of multiple roots of even multiplicity.

12. More Numerical Experiments

Consider the following numerical test. Choose an integer $p > 1$ such that $p+1$ is not an integer power of 2 and let V be a p–vector with rational elements $V_k = 1+kh$ between 1 and 2, i.e. $V = [1+h, 1+2h, \ldots, 1+ph]$, where $h = 1/(p+1)$. The vector V is computed by the commands `>>h=1/(p+1);V=1+h:h:2-h;`

The aim of the numerical test is to estimate the mean value

$$U(p) = p^{-1} \sum_{k=1}^{p} U_k(p)$$

of the quantities $U_k(p) = |\mathrm{flo}(V_k) - V_k|/(V_k\rho)$. The computed values `Up` of $U(p)$ are expected to be close to the coefficient C in the bound $C\rho$ for the average rounding errors in the normal range of FPA. The experiment is performed by the MATLAB® command

```
>>Up=double(mean(abs(sym(V,'e')./V-ones(1,length(p))))*2/eps)
```

Here the MATLAB® function `sym(a,'e')` (`sym` is actually a Maple function [7]) computes a binary machine approximation $\mathrm{flo}(a)$ of the numerical array a plus an approximation of the form eps $\times$ A of the absolute rounding error. Here where A is an array of the same type as a with rational elements of order 1.

For instance, for the 2-vector $a = [1 + 1/3, 1 + 2/3]$ the command

```
>> sym([1+1/3,1+2/3],'e')
```

gives `[4/3-eps/3, 5/3-(2*eps)/3]`, i.e. in this case $A = [-1/3, -2/3]$. The result is different if we use the command `>> sym([4/3,5/3],'e')` which gives `[4/3-eps/3, eps/3+5/3]` with $A = [-1/3, 1/3]$.

The results of this numerical experiment are shown at Tables 3 and 4. We stress that if $p+1$ was an integer degree of 2 then the rounding errors would be zero and $U(p) = 0$. The same is true if we choose V by a quasi-random number generator, e.g. `V=1+rand(1,p)` since in this case the elements of V are machine numbers.

p	10	20	30	40	50	60	70	80	90	100
U	.503	.408	.387	.392	.507	.377	.371	.418	.375	.361

Table 3: Coefficients for relative rounding errors

p	9	99	999	999
h	0.1	0.01	0.001	0.0001
U	0.613	0.363	0.672	0.360

Table 4: More coefficients for relative rounding errors

13. Important Constants of FPA

Some important constants of binary FPA obeying the IEEE Standard [5] are listed below.

1. The symbols $\pm$`Inf` for $\pm\infty$ and the symbol `NaN` for Not a Number.
2. The rounding unit
$$\rho = 2^{-53} \simeq 1.1102 \times 10^{-16} \in \mathbb{M}_+$$
such that $\mathrm{flo}(1+\rho) = 1$.
3. The machine epsilon (denoted also as `eps`)
$$\varepsilon = 2\rho = 2^{-52} \simeq 2.2204 \times 10^{-16} \in \mathbb{M}_+$$
such that $\mathrm{flo}(1+\varepsilon) = 1+\varepsilon$.

4. The number
$$N = 2^{52} \simeq 4.5036 \times 10^{15} \in \mathbb{M}_+$$
of NMN in any interval $[2^m, 2^{m+1}) \subset \mathbb{R}_+$ with $m \in \mathbb{Z}$ and $-1022 \le m \le 1023$; we have $\mathrm{flo}(2N+1) = 2N$.

5. The maximal NMN (denoted also as `realmax`)
$$R = 2^{1023}(2 - \mathrm{eps}) \simeq 1.7977 \times 10^{308} \in \mathbb{M}_+$$
such that $\mathrm{flo}(R + 2^{969}) = R$ and $\mathrm{flo}(R + 2^{970}) =$ `Inf`.

6. The minimal positive NMN (denoted also as `realmin`)
$$r = 2^{-1022} \simeq 2.2251 \times 10^{-308} \in \mathbb{M}_+$$

7. The minimal positive subnormal MN
$$\omega = 2^{-1074} \simeq 4.9407 \times 10^{-324} \in \mathbb{M}_+$$
such that $\mathrm{flo}(\omega/(2-\varepsilon)) = \omega$ and $\mathrm{flo}(\omega/2) = 0$.

14. On Sums of Inverse Integers

Here we present some facts about finite sums of fractions [1]. Set
$$H_n = \sum_{k=1}^{n} \frac{1}{k}\,,\ X_n := \sum_{k=1}^{n} \frac{1}{2k-1},\ Y_n = X_{2n} - X_n = \sum_{k=n+1}^{2n} \frac{1}{2k-1}. \tag{31}$$
Then
$$H_n = \log(n) + \gamma + \gamma_n,\ X_n = H_{2n} - \frac{1}{2}H_n,\ Y_n = H_{4n} - \frac{3}{2}H_{2n} + \frac{1}{2}H_n, \tag{32}$$
where $\gamma = 0.5772156649\ldots$ is the Ouler-Mascheroni constant and the quantity γ_n satisfies the inequalities $1/(2n+2) < \gamma_n < 1/(2n)$. We have
$$\begin{aligned} X_n &= \log(2n) + \gamma + \gamma_{2n} - \frac{1}{2}(\log(n) + \gamma + \gamma_n) \\ &= \log(2) + \frac{1}{2}\,\gamma + \frac{1}{2}\log(n) + \gamma_{2n} - \frac{1}{2}\,\gamma_n \end{aligned} \tag{33}$$

and relations (31)–(32) yield the bounds $X'_n < X_n < X''_n$ for X_n. Here

$$\begin{aligned} X'_n &= \log(2) + \frac{1}{2}\gamma + \frac{1}{2}\log(n) - \frac{1}{4n(2n+1)}, \\ X''_n &= \log(2) + \frac{1}{2}\gamma + \frac{1}{2}\log(n) + \frac{1}{4n(n+1)}. \end{aligned}$$

Hence

$$\begin{aligned} Y_n &< X''_{2n} - X'_n = \frac{1}{2}\log(2) + \frac{3}{8n(2n+1)}, \\ Y_n &> X'_{2n} - X''_n = \frac{1}{2}\log(2) - \frac{3(3n+1)}{8n(n+1)(4n+1)} \end{aligned}$$

and

$$Y_n = \frac{1}{2}\log(2) + \mathrm{O}\left(n^{-2}\right). \tag{34}$$

Acknowledgments

This work is supported by Project DN12/11/20.dec.2017 of the National Science Fund – Bulgarian Ministry of Education and Science.

References

[1] I. Alabdulmohsi, *Summability Calculus: A Comprehensive Theory of Fractional Finite Sums*, Springer, Berlin (2018); doi 10.1007/978-3-319-74648-7.

[2] N. Higham, The scaling and squaring method for the matrix exponential revisited, *SIAM Review*, **51** (2009), 747-764; doi 10.1137/090768539.

[3] N. Higham, *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia (2002), ISBN 0-89871-521-0.

[4] N. Higham, M. Konstantinov, V. Mehrmann, P. Petkov, The sensitivity of computational control problems, *IEEE Control Systems Magazine*, **24** (2004), 28-43; doi 10.1109/MCS.2004.1272744.

[5] IEEE 754–2019, *Standard for Floating-Point Arithmetic*, New York (2019); standards.ieee.org/standard/754-2019.html.

[6] M. Konstantinov, P. Petkov, Large computational errors, *Proceedings of the International Scientific Conference REMIA2021*, Plovdiv (2021), 31-49, ISBN 978-619-202711-7.

[7] Maple (ver. 2020), *Maplesoft*, Waterloo (2020), maplesoft.com.

[8] Mathematica (ver. 12.0), *Wolfram Reseach Inc.*, Champaign (2019), wolfram.com/mathematica.

[9] MATLAB (ver. R2019a), *MathWorks Inc.*, Natick (2019), mathworks.com/products/matlab.

[10] P. Petkov, N. Christov, M. Konstantinov, *Computational Methods for Linear Control Problems*, Prentice Hall, Hemel Hempstead (1991), ISBN 0-13-161803-2.

[11] N. Vulchanov, M. Konstantinov, *Modern Mathematical Methods for Computer Calculations, Part 1: Foundations of Computer Calculations*, Bulgarian Institute for Analyses and Research, Sofia (1996, 2006), ISBN 954-8949-01-6.