Accurate Summation: Towards a Simpler and Formal Proof

Laurent Fousse [a], Paul Zimmermann [b]

[a] *3, impasse Manet, F-57730 Folschviller*

[b] *LORIA/INRIA Lorraine, 615, rue du jardin botanique, F-54602 Villers-lès-Nancy Cedex*

Abstract

This paper provides a simpler proof of the "accurate summation" algorithm proposed by Demmel and Hida in [1]. It also gives improved bounds in some cases, and examples showing that those new bounds are optimal. This simpler proof will be used to obtain a computer-checked proof of Demmel-Hida's algorithm.

Key words: Floating point summation, bounded error, formal proof.

1 Introduction

Adding several floating-point numbers with good accuracy is an important problem of scientific computing. The best possible result is to provide *correct rounding* within the target precision and the given rounding mode. For a sum of two numbers, this can be obtained with a configuration following the IEEE 754 standard [3]. For a sum of three or more numbers, not only the floating-point addition is not associative, but some cancellations may occur in the intermediate sums, that will make the final result completely wrong. Several algorithms have been proposed to overcome this problem. Floating-point expansions [5] enable one to emulate an arbitrary-precision arithmetic using fixed-precision floating-point numbers: an intermediate result is represented by a sum of disjoint floating-point numbers, so that the most significant one represents the rounding to nearest of the result.

Email addresses: `laurent@fousse.info` (Laurent Fousse), `zimmerma@loria.fr` (Paul Zimmermann).

We focus in this paper on algorithms that use an internal register, with a precision that is larger than that of the summands and the final result (which we assume to have same precision for simplicity). Kulish and Bohlender propose in [4] an algorithm that provides correct rounding of a dot product of n-digit vectors, with an accumulator of $2n + 2$ digits plus two bits. However, Kulish-Bohlender's algorithm is quite expensive to implement, since it requires to sort the inputs by decreasing exponent, and to combine those of same exponent (in a way similar to floating-point expansions).

More recently, Demmel and Hida proposed in [1,2] a simple algorithm that yields a sum correct to within about 1.5 ulps, as long as the number of summands is not too large. The proof of the main theorem from [1] is quite complex; quoting [2]: "*it* [the proof of Theorem 1] *involves a detailed case analysis requiring* 19 *pages*". Such a long proof is difficult to completely grasp by a human-being, and is therefore a good candidate for the use of a proof assistant.

Our goal is to provide an alternate proof of Theorem 1 from [1], which could be computer-checked by a proof assistant like HOL, PVS or Coq. This paper is a first step towards this goal. A first objective is to reduce the number of cases (6 cases and 11 subcases) from Demmel-Hida's proof. A second objective is to understand if the bounds given by Demmel and Hida are all optimal, since examples that attain them are not given in all cases. A third objective is to transform the intermediate lemmas from the proof to better computer-readable statements. For example the following statement is hard to translate literally in computer form:

> ***Fact 1.*** *The rightmost nonzero bit of $\widehat{SUM}_k$ must lie at or to the left of the trailing bit of s_k. This is because $\widehat{SUM}_k$ is gotten by summing s_1 through s_k, whose trailing bits are at or to the left of the trailing bit of s_k, since the exponents of s_1 through s_k are in decreasing order.*

Our paper is organized as follows. Section 2 introduces the notations and assumptions used in the rest of the paper, recalls Demmel-Hida's algorithm and Theorem 1. Section 3 provides some simpler proofs of the different cases considered by Demmel and Hida. Section 4 gives examples reaching those bounds, thus showing that they are optimal. Finally, Section 6 describes our workplan to complete the computer-checked proof of Demmel-Hida's algorithm.

2 Demmel-Hida's Algorithm

2.1 Notations

Exponent. For x a non-zero real number (not necessarily a floating-point number), we define $E(x) := 1 + \lfloor \log_2 |x| \rfloor$, such that $2^{E(x)-1} \le |x| < 2^{E(x)}$.

Unit in last place. For x a non-zero real number (not necessarily a floating-point number), and an integer $f \ge 1$, we define $\mathrm{ulp}_f(x) := 2^{E(x)-f}$. When x is a floating-point number with a mantissa of f bits, $\mathrm{ulp}_f(x)$ corresponds to the weight of the last mantissa bit.

Rounding. We denote $\circ_F(x)$ the F-bit floating-point number that is nearest from the real x, with ties broken to even mantissa. (We consider here only rounding to nearest.) When the destination variable t has known precision F, we may omit the index F and write simply $t \leftarrow \circ(x)$.

Exact/Inexact. We say that a computation $\circ(x + y)$ is *exact* (resp. *inexact*) when the result is (resp. is not) exactly $x + y$. When that result is stored in a variable, like $z \leftarrow \circ(x + y)$, we say that "$z$ is exact", meaning that $z = x + y$.

In the following we consider floating-point numbers with *unbounded exponent*, i.e. no underflow nor overflow occurs. (Demmel and Hida allow gradual underflow in their proof, but for sake of simplicity we disallow it here.)

2.2 Demmel-Hida's algorithm

Consider the following sequence of operations, where all s_i and S have precision f, the t_i have precision $F > f$, and the s_i have non-increasing exponents:

Algorithm 1.

```
$t_0 \leftarrow 0$
for $i \leftarrow 1$ to $n$
    $t_i \leftarrow \circ_F(t_{i-1} + s_i)$
$S \leftarrow \circ_f(t_n)$
```

Remark. Replacing the two last operations of Algorithm 1 — $t_n \leftarrow \circ_F(t_{n-1} + s_n)$ and $S \leftarrow \circ_f(t_n)$ — by $S \leftarrow \circ_f(t_{n-1} + s_n)$ may avoid the "double rounding" problem, and thus reduce the final error by $\frac{1}{2}\mathrm{ulp}_F(t_n)$. However we don't consider this improvement here, and stick to the original algorithm from Demmel and Hida.

Theorem 1 *[1] Assume $F > f \geq 2$, the exponent range for the F-bit format is at least as wide as that for the f-bit format, rounding is to nearest (ties may be broken arbitrarily), underflow is gradual if it occurs, and no overflow occurs. Let*

$$\bar{n} = 1 + \left\lfloor \frac{2^F}{2^f - 1} \right\rfloor$$

then one of the following four cases holds for the error ε in ulps on the computed sum S at the end of Algorithm 1:

(1) if $n \leq \bar{n}$, then $\varepsilon \leq \frac{1}{1-2^{1-f}} + \frac{1}{2}$;
(2) if $n = \bar{n}+1$, $F \geq 2f$, and s_2 is normalized, then $\varepsilon \leq \max\{2.5, \frac{1.5}{1-2^{1-f}} + \frac{1}{2}\}$;
(3) if $n = \bar{n} + 1$, and either $F < 2f$ or s_2 is unnormalized, then $\varepsilon \leq \max\{3.5, 2^r + \frac{1}{2}\}$ where $r = f - (F \bmod f)$, $1 \leq r \leq f$;
(4) if $n \geq \bar{n}+2$, then ε can be larger or equal to 1. In particular, the computed sum may be zero when the true sum is not.

In addition to Demmel-Hida notations, we define $k := F - f$. Then $\bar{n} = 1+2^k+\lfloor 2^{k-f}+2^{k-2f}+\cdots\rfloor$. For $F < 2f$, we have $\bar{n} = 1+2^k$; for $2f \leq F < 3f$, we have $\bar{n} = 1 + 2^k + 2^{k-f}$; and so on.

3 Bounds

3.1 Notations

Error and total error. Let $\rho_j := |t_j - (t_{j-1} + s_j)|$ be the rounding error on t_j — we will simply say the "error on t_j" —, $\varepsilon_j := \sum_{l=1}^{j} \rho_l$ the *total error* on t_j, in terms of $\mathrm{ulp}_F(t_j)$, and ε the final error on S, in terms of $\mathrm{ulp}_f(S)$. We will also sometimes consider partial sums: $\varepsilon_{i,j} := \sum_{i<l\leq j} \rho_l$.

Exponent difference. For $1 \leq i \leq j \leq n$, we define $e_j := E(t_j) - E(t_i)$ to be the exponent difference between t_j and t_i.

3.2 Case (1): $n \leq \bar{n}$

We prove here a few lemmas that will help us to prove Theorem 2, which corresponds to case (1) of Theorem 1. The first result is that after the first inexact partial sum t_i, the summands s_j cannot be too large (Lemma 1), and thus the partial sums t_j cannot decrease too rapidly (Lemma 4). The key idea of the proof is that when the exponent decreases by one from t_{j-1} to t_j, the total error accumulated so far is multiplied by two, since it is now expressed in terms of $\text{ulp}_F(t_j)$, but on the other side the maximum absolute value of further summands s_j that may produce a rounding error is divided by two. Lemma 5 expresses this trade-off between having large s_j (and thus making t_j decrease rapidly) and accumulating rounding errors.

Lemma 1 *Let $i \leq n$ be the smallest index such that t_i is inexact, i.e. $t_i \neq t_{i-1} + s_i$. Then $\text{ulp}_F(t_i) > \text{ulp}_f(s_i)$.*

(If no such i exists, then case (1) of Theorem 1 is trivial.)

PROOF. Since $\text{ulp}_f(s_j) \geq \text{ulp}_f(s_i)$ for $j \leq i$, we can write $s_1 + \cdots + s_i = m \cdot \text{ulp}_f(s_i)$ with m an integer. Since t_{i-1} is exact, t_i is the rounding to nearest of $m \cdot \text{ulp}_f(s_i)$. Then t_i inexact implies $|m| \geq 2^F$, which implies $|\circ(m)| \geq 2^F$, and $\text{ulp}_F(t_i) > \text{ulp}_f(s_i)$. □

The following result is sort of a converse of Lemma 1:

Lemma 2 *If $\text{ulp}(s_j) \geq \text{ulp}(t_j)$, then t_j is exact.*

PROOF. By definition of rounding to nearest, the rounding error on t_j is at most $\frac{1}{2}\text{ulp}(t_j)$. Since t_{j-1} is the (rounded) sum of s_1 to s_{j-1}, which have an exponent greater or equal to that of s_j, t_{j-1} is an integer multiple of $\text{ulp}(s_j)$. Then $t_{j-1} + s_j$ is also an integer multiple of $\text{ulp}(s_j)$, and of $\text{ulp}(t_j)$ since $\text{ulp}(s_j) \geq \text{ulp}(t_j)$. The only one such multiple at distance $\frac{1}{2}\text{ulp}(t_j)$ of t_j is precisely t_j, which is then exact. □REMARK. The fact that t_{j-1} is the sum of numbers with exponent larger or equal to that of s_j is crucial here: the above result is wrong for $t' := \circ(t + s)$. Take for example $f = 2$, $F = 3$, $t = 0.111$, $s = 0.11$, then $t' = \circ_3(1.101) = 1.10$ is inexact, while $\text{ulp}(s) \geq \text{ulp}(t')$ holds. (However, a sufficient auxiliary condition is $\text{ulp}(t') \leq \text{ulp}(t)$.)

Lemma 3 *Let t be a floating-point number of precision F. Let s be a real number such that $|s| < 2^l$, with $E(t) - F \leq l < E(t)$. Let $t' = \circ_F(t + s)$, then $|t' - t| \leq 2^l$.*

PROOF. Without loss of generality we can consider that t is positive. The condition $|s| < 2^l$ implies $|s| < t$ since $t \geq 2^{E(t)-1} \geq 2^l$.

First consider s negative. The condition $E(t) - F \leq l < E(t)$ implies $1 \leq \frac{2^l}{\text{ulp}(t)} \leq 2^{F-1}$. Since t can be written $m \cdot \text{ulp}(t)$ with $2^{F-1} \leq m < 2^F$, we deduce $0 \leq \frac{t-2^l}{\text{ulp}(t)} < 2^F$, thus $t - 2^l$ is exactly representable with a precision of F bits. The real $t + s$ is enclosed by two representable numbers $t - 2^l$ and t, thus the same holds for its rounding: $t - 2^l \leq t' \leq t$, and the statement of the lemma follows.

Now consider s positive. If $t + 2^l$ is exactly representable, the same reasoning as above holds. If $t + 2^l$ is not representable on F bits, then it is the middle of two consecutive F-bit representable numbers $u < u^+$. Since $t + s < t + 2^l$, necessarily $t' := \circ(t + s) \le u < t + 2^l$. □

Corollary 1 *With the notations of Lemma 3, if s is a f-bit floating-point number, and* $\mathrm{ulp}_f(s) \le \mathrm{ulp}_F(t')$, *then* $|t'| \ge |t| - (2^f - 1)\mathrm{ulp}_F(t')$.

PROOF. Assume t positive. If s is positive, the corollary is trivial. If $\mathrm{ulp}_f(s) < \mathrm{ulp}_F(t')$, Lemma 3 gives $|t' - t| \le 2^{f-1}\mathrm{ulp}_F(t')$. If $\mathrm{ulp}_f(s) = \mathrm{ulp}_F(t')$, since $E(t') \le E(t)$, t' is necessarily exact (see the remark following Lemma 2). □

Lemma 4 *Let i be the first index such that t_i is inexact, and $i' \ge i$ such that $E(t_j) < E(t_i)$ for $i' < j \le n$ (take $i' = n$ if $E(t_j) \ge E(t_i)$ for $i < j < n$). Then for $j \le i' + \bar{n} - 2$, we have $E(t_j) \ge E(t_i) - k$.*

PROOF. (Remember that $k := F - f$.) From Lemma 1 and the decrease of the exponents of the s_j, we deduce $\mathrm{ulp}_f(s_j) < \mathrm{ulp}_F(t_i)$ for $j \ge i'$. Thus from Corollary 1 the maximum decrease due to s_j is $\frac{1}{2}(2^f - 1)\mathrm{ulp}_F(t_i)$. When summing $\bar{n} - 2$ terms, we thus have:

$$\begin{aligned}|t_j| &\ge |t_{i'}| - \tfrac{1}{2}(\bar{n} - 2)(2^f - 1)\mathrm{ulp}_F(t_i)\\ &\ge \tfrac{1}{2}[2^F - (\bar{n} - 2)(2^f - 1)]\mathrm{ulp}_F(t_i) \ge 2^{f-1}\mathrm{ulp}_F(t_i),\end{aligned}$$

since $2^F > (\bar{n} - 1)(2^f - 1)$, because $2^f - 1$ cannot divide exactly 2^F (remember $f \ge 2$). □

We define for $j > i'$ the "decrease default" d_j as the difference between the maximal decrease between t_{j-1} and t_j — which is $2^{f-1}\mathrm{ulp}_F(t_i)$, which may occur only when $E(s_j) = E(s_i)$ — and the real decrease; we also define $D_n := \sum_{i'<j<n} d_j$ to be the cumulative decrease default from $i' + 1$ to n.

Lemma 5 *Let i and i' as in Lemma 4. Then for $n < i' + \bar{n} - 1$, the total error when summing s_j for $i' < j \le n$ satisfies*

$$\varepsilon_{i',j} < \frac{2^k - 2^{-e_n - 1}(2^k - (n - i')(1 - 2^{-f}))}{1 - 2^{1-f}}\,\mathrm{ulp}_F(t_n).$$

PROOF. (If $i' = n$, the error is zero, and the statement is true.) Let $\delta_j := \log_2 \frac{\mathrm{ulp}_F(t_j)}{\mathrm{ulp}_f(s_j)}$ be the logarithmic difference between $\mathrm{ulp}_F(t_j)$ and $\mathrm{ulp}_f(s_j)$. If $\delta_j \le 0$, then $\mathrm{ulp}_F(t_j) \le \mathrm{ulp}_f(s_j)$, thus t_j is exact from Lemma 2. If $\delta_j > 0$, then $\mathrm{ulp}_f(s_j) \le \frac{1}{2}\mathrm{ulp}_F(t_j)$, thus by Lemma 3 the decrease from t_{j-1} to t_j is bounded by $2^{f-1}\mathrm{ulp}_F(t_j)$. Since $E(t_j) < E(t_i)$ for $j > i'$, this maximal decrease is about half of the value $\frac{1}{2}(2^f - 1)\mathrm{ulp}_F(t_i)$ taken into account in Lemma 4.

Thus the decrease default d_j is at least $(2^{f-1}-1)\mathrm{ulp}_F(t_j)$ per index $j > i'$ such that $\delta_j > 0$:

$$D_n \geq \sum_{\substack{i'<j\leq n\\ \delta_j>0}} (2^{f-1}-1)\mathrm{ulp}_F(t_j) = (2^{f-1}-1) \sum_{\substack{i'<j\leq n\\ \delta_j>0}} 2^{e_j}\mathrm{ulp}_F(t_i).$$

Without taking into account the decrease default, we would have:

$$\begin{aligned}|t_n| &\geq |t_{i'}| - \tfrac{1}{2}(n-i')(2^f-1)\mathrm{ulp}_F(t_i) \geq \tfrac{1}{2}[2^F - (n-i')(2^f-1)]\mathrm{ulp}_F(t_i)\\ &\geq [2^k - (n-i')(1-2^{-f})]2^{f-1}\mathrm{ulp}_F(t_i).\end{aligned}$$

Thus, taking into account the decrease default, we have $[2^k-(n-i')(1-2^{-f})]2^{f-1}\mathrm{ulp}_F(t_i) + D_n \leq |t_n| < 2^F\mathrm{ulp}_F(t_n) = 2^{F+e_n}\mathrm{ulp}_F(t_i)$. It follows:

$$2^k - (n-i')(1-2^{-f}) + (1-2^{1-f}) \sum_{\substack{i'<j\leq n\\ \delta_j>0}} 2^{e_j} \quad < \quad 2^{k+1+e_n}. \tag{1}$$

The sum $\varepsilon_{i',n}$ of rounding errors on t_j for $i' < j \leq n$ is at most:

$$\varepsilon_{i',n} \leq \sum_{\substack{i'<j\leq n\\ \delta_j>0}} \frac{1}{2}\mathrm{ulp}_F(t_j) = 2^{-e_n-1}\mathrm{ulp}_F(t_n) \sum_{\substack{i'<j\leq n\\ \delta_j>0}} 2^{e_j}.$$

Using Eq. (1), this gives

$$\begin{aligned}(1-2^{1-f})\varepsilon_{i',n} &\leq 2^{-e_n-1}[2^{k+1+e_n} - 2^k + (n-i')(1-2^{-f})]\,\mathrm{ulp}_F(t_n)\\ &\leq [2^k - 2^{-e_n-1}(2^k-(n-i')(1-2^{-f}))]\,\mathrm{ulp}_F(t_n). \qquad \square\end{aligned}$$

Theorem 2 *Let i be the smallest index such that t_i is inexact. Then for $n \leq i + \bar{n} - 2$, the total error on t_n satisfies $\varepsilon_n < \frac{2^k}{1-2^{1-f}}\mathrm{ulp}_F(t_n)$.*

PROOF. First assume that $E(t_j) \leq E(t_i)$ for $j \geq i$. Let i' as in Lemma 5. The maximal error when summing s_i up to $s_{i'}$ (both included) is $\frac{i'-i+1}{2}\mathrm{ulp}_F(t_i) = (i'-i+1)2^{-e_n-1}\mathrm{ulp}_F(t_n)$. Together with the bound on the cumulated error when adding $s_{i'+1}$ to s_n given by Lemma 5, this gives a total error on t_n in terms of $\mathrm{ulp}_F(t_n)$ less than:

$$\begin{aligned}&\tfrac{2^k-2^{-e_n-1}(2^k-(n-i')(1-2^{-f}))}{1-2^{1-f}} + (i'-i+1)2^{-e_n-1}\tfrac{1-2^{-f}}{1-2^{1-f}}\\ &\leq \tfrac{2^k-2^{-e_n-1}(2^k-(n-i+1)(1-2^{-f}))}{1-2^{1-f}}.\end{aligned}$$

For $n \le i + \bar{n} - 2$, we have $(n - i + 1)(1 - 2^{-f}) \le (\bar{n} - 1)(1 - 2^{-f}) < 2^k$, and the theorem follows.

Consider now that there exists $j > i$ such that $E(t_j) > E(t_i)$. Then for $l > i$, $|t_l| \ge |t_j| - (\bar{n} - 1)\frac{2^f-1}{2}\mathrm{ulp}_F(t_i) \ge [2^F - \lfloor \frac{2^F}{2^f-1} \rfloor \frac{2^f-1}{2}]\mathrm{ulp}_F(t_i) \ge 2^{F-1}\mathrm{ulp}_F(t_i)$. Thus $E(t_l) \ge E(t_i)$. On the other side, $|t_l| \le |t_i| + (\bar{n} - 1)2^{f-1}\mathrm{ulp}_F(t_i) < 2^F[1 + \frac{1}{2(1-2^{-f})}]\mathrm{ulp}_F(t_i) \le 2^{F+1}\mathrm{ulp}_F(t_i)$. It follows $E(t_l) \le E(t_i) + 1$. We thus deduce $E(t_l) \le E(t_n) + 1$ for $i \le l \le n$, and the total error on t_n satisfies:

$$\frac{\varepsilon_n}{\mathrm{ulp}_F(t_n)} < \frac{1}{2}\sum_{l=i}^{n} 2^{E(t_l)-E(t_n)} \le n - i + 1 \le \bar{n} - 1 \le \frac{2^F}{2^f - 1} \le \frac{2^k}{1 - 2^{1-f}}. \square$$

Since $i \ge 2$ (t_1 is always exact), we get the following corollary.

Corollary 2 *For $n \le \bar{n}$, the total error on t_n satisfies $\varepsilon_n < \frac{2^k}{1-2^{1-f}}$ ulps.*

3.3 Cases (2) and (3): $n = \bar{n} + 1$

Theorem 3 *When $n = \bar{n} + 1$, the error on S is $\varepsilon \le \max\{\frac{2}{1-2^{1-f}} + \frac{1}{2^{k+1}} + \frac{1}{2}, 2^{r-1}\}$ with $r = f - (F \bmod f)$.*

Proof. With the notations of Lemma 5, assume $i' \ge 3$, then Lemma 5 applies since $n < i' + \bar{n} - 1$, and $E(t_n) \ge E(t_i) - k$, thus Theorem 2 applies to bound the rounding errors on t_3 to t_n, and the error on t_2 is at most $2^{k-1}\mathrm{ulp}_F(t_n)$, which gives a total error $\le 2^k[\frac{1}{1-2^{1-f}} + \frac{1}{2}]\mathrm{ulp}_F(t_n)$, thus at most $\frac{1}{1-2^{1-f}} + 1$ ulps on S.

If $i' = i = 2$, we distinguish two cases: (i) $\mathrm{ulp}_f(s_j) = \mathrm{ulp}_F(t_3)$ for $3 \le j \le n$, and (ii) at least one $\mathrm{ulp}_f(s_j) < \mathrm{ulp}_F(t_3)$. In case (ii), we have $|s_{\bar{n}+1}| \le 2^{f-2}\mathrm{ulp}_F(t_2)$, so $|t_{\bar{n}+1}| \ge |t_{\bar{n}}| - 2^{f-2}\mathrm{ulp}_F(t_2)$, and with $|t_{\bar{n}}| \ge 2^{f-1}\mathrm{ulp}_F(t_2)$ this yields: $\frac{|t_{\bar{n}+1}|}{|t_{\bar{n}}|} \ge 1 - \frac{2^{f-2}\mathrm{ulp}_F(t_2)}{|t_{\bar{n}}|} \ge 1 - \frac{1}{2} = \frac{1}{2}$ thus $E(t_{\bar{n}+1}) \ge E(t_{\bar{n}}) - 1$, and the final error is at most twice that on $t_{\bar{n}}$ (from Theorem 2) plus $\frac{1}{2}$, i.e. $\le \frac{2^{k+1}}{1-2^{1-f}} + \frac{1}{2}$ ulps on t_n. After division by 2^k, and addition of the maximal error of $\frac{1}{2}$ on S, this gives $\frac{2}{1-2^{1-f}} + \frac{1}{2^{k+1}} + \frac{1}{2}$, which is always larger than the bound in case $i' \ge 3$.

In case (i), $\mathrm{ulp}_f(s_j) = \mathrm{ulp}_F(t_3)$, we have $\mathrm{ulp}_f(s_j) \ge \mathrm{ulp}_F(t_j)$ for $3 \le j \le n$, since $\mathrm{ulp}_F(t_j)$ cannot exceed $\mathrm{ulp}_F(t_3)$.[1] This by Lemma 2 all t_j are exact for

[1] Remember that $forj > i' = 2$, $E(t_j) < E(t_{i'}$, and necessarily $E(t_{i'+1}) = E(t_{i'}) - 1$, since the exponent cannot decrease by more than one from $t_{i'}$ to $t_{i'+1}$, because $|s_{i'+1}| < 2^{f-1}\mathrm{ulp}_F(t_{i'}) \le \frac{1}{2}|t_{i'}|$.

$j \geq 3$, thus the error only comes from the rounding error on t_2, multiplied by $2^{E(t_2)-E(t_n)}$. We have the following inequalities:

$$
\begin{aligned}
|t_2| &\geq 2^{F-1}\mathrm{ulp}_F(t_2) && (2)\\
|s_j| &\leq \frac{1}{2}(2^f - 1)\mathrm{ulp}_F(t_2) \text{ for } 3 \leq j \leq \bar{n} + 2 && (3)\\
t_n &\geq \frac{1}{2}[2^F - (\bar{n} - 1)(2^f - 1)]\mathrm{ulp}_F(t_2) \\
&\geq \frac{1}{2}[2^F - \lfloor \frac{2^F}{2^f - 1} \rfloor (2^f - 1)]\mathrm{ulp}_F(t_2) \\
&= 2^{(F \bmod f)-1}\mathrm{ulp}_F(t_2) && (4)
\end{aligned}
$$

Thus $E(t_n) \geq E(t_2) + (F \bmod f) - F$, and the error on t_n satisfies $\varepsilon_n \leq \frac{1}{2}2^{F-(F \bmod f)}\mathrm{ulp}_F(t_n)$, and that on S is $\varepsilon \leq 2^{f-1-(F \bmod f)} = 2^{r-1}$.

If at least one inequality of (2) or (3) is strict, then (4) is strict as well, so the worst exponent decrease occurs only in this particular case where there is no rounding error on the final step (see Section 4.2 for an example). This explains why we don't have the $\frac{1}{2}$ rounding error as usual. In all other cases we can bound the total error on S by $2^{r-2} + \frac{1}{2} \leq 2^{r-1}$ because $r \geq 1$.

4 Reaching the Bounds

We assume here that even rounding, i.e. ties are broken by rounding to the floating-point number with a mantissa ending by a zero.[2]

4.1 Case (1): $n \leq \bar{n}$

Lemma 6 *When summing $n = 1 + 2^k$ terms, we can get as near as desired to the error bound of $1.5\ \mathrm{ulp}_f(S)$.*

Proof. Let $s_1 = 1 + 2^{k-f+1} - 2^{1-f}$, $s_i = 2^{-F} - 2^{1-f}$ for $1 < i < n$, and $s_n = 2^{-F-1} - 2^{-f} - 2^{-f-1}$. Then for $1 \leq i < n$, $t_i = 1 + 2^{k-f+1} - i \cdot 2^{1-f}$. (easy induction)
Since $n - 1 = 2^k$, we get $t_{n-1} = 1 + 2^{k-f+1} - 2^k \cdot 2^{1-f} = 1$, and each of the $n-2$ summations gives an error of $\frac{1}{2}\ \mathrm{ulp}_F(t_1)$. The total error on S is therefore $\frac{1}{2}(2^k - 1)\mathrm{ulp}_F(t_1) + \frac{1}{2}\mathrm{ulp}_f(S) = (2^k - 1)\ \mathrm{ulp}_F(t_n) + \frac{1}{2}\ \mathrm{ulp}_F(t_n) + \frac{1}{2}\ \mathrm{ulp}_f(S) = (\frac{3}{2} - \frac{1}{2^{k+1}})\ \mathrm{ulp}_f(S)$. □

[2] This implies $f \geq 2$ since for $f = 1$ all non-zero numbers have an odd mantissa.

Here is an example for $f = 7$, $F = 10$, $n = 1 + 2^3 = 9$:

$$\begin{aligned}
s_1 = t_1 &= 1.000111000 \\
s_2 &= -\,0.0000001111000 \\
t_2 &= 1.000110000 \\
s_3 &= -\,0.0000001111000 \\
t_3 &= 1.000101000 \\
s_4 &= -\,0.0000001111000 \\
t_4 &= 1.000100000 \\
s_5 &= -\,0.0000001111000 \\
t_5 &= 1.000011000 \\
s_6 &= -\,0.0000001111000 \\
t_6 &= 1.000010000 \\
s_7 &= -\,0.0000001111000 \\
t_7 &= 1.000001000 \\
s_8 &= -\,0.0000001111000 \\
t_8 &= 1.000000000 \\
s_9 &= -\,0.0000001011100 \\
t_9 &= 0.1111110100 \\
S &= 0.1111110 \\
\text{Correct Sum} &= 0.111111101110
\end{aligned}$$

REMARK. Taking the same example, but starting from t_i instead of t_1, we get an error of $\frac{2n-3}{2^{k+1}} + \frac{1}{2}$ for $n \leq 1 + 2^k$.

4.2 Case (3): $n = \bar{n} + 1$ and $F < 2f$

Lemma 7 *Assume $f < F < 2f$, and let $r = 2f - F$. When summing $n = 2^k + 2$ terms we can reach an error bound of $2^{r-1}\mathrm{ulp}_f(S)$.*

PROOF. Let $s_1 = 1 + 2^{-k}$ and $s_i = -(1 - 2^{-f})2^{-k}$ for $2 \leq i \leq n$. Then $t_2 = 1$ with an error of $\frac{1}{2}\mathrm{ulp}_F(t_1)$, and the exponent of t_3 is decreased by 1. With Lemma 1, t_3 is exact (no rounding error occurs at this step) and we can check by induction that t_3 through t_n are exact.

Last value is $t_n = 1 - 2^k(1 - 2^{-f})2^{-k} = 2^{-f}$, with no rounding error when putting it back to f bits, whereas the true sum is

$$\begin{aligned}
1 + 2^{-k} - (2^k + 1)(1 - 2^{-f})2^{-k} &= 2^{-k} + 2^{-f} - 2^{-k} + 2^{-F} \\
&= 2^{-f} + 2^{-F}
\end{aligned}$$

and thus the error of $2^{-k-(-f+1)}\mathrm{ulp}_f(S) = 2^{f-k-1}\mathrm{ulp}_f(S) = 2^{r-1}\mathrm{ulp}_f(S)$. $\square$

Here is an example for $f = 7$, $F = 10$, $n = 2 + 2^3 = 10$:

$$\begin{aligned}
s_1 &= \,1.001000 \\
t_1 &= \,1.001000000 \\
s_2 &= -\,0.0001111111 \\
t_2 &= \,1.000000000 \\
s_3 &= -\,0.0001111111 \\
t_3 &= \,0.1110000001 \\
s_4 &= -\,0.0001111111 \\
t_4 &= \,0.1100000010 \\
s_5 &= -\,0.0001111111 \\
t_5 &= \,0.1010000011 \\
s_6 &= -\,0.0001111111 \\
t_6 &= \,0.1000000100 \\
s_7 &= -\,0.0001111111 \\
t_7 &= \,0.01100001010 \\
s_8 &= -\,0.0001111111 \\
t_8 &= \,0.01000001100 \\
s_9 &= -\,0.0001111111 \\
t_9 &= \,0.001000011100 \\
s_{10} &= -\,0.0001111111 \\
t_{10} &= \,0.0000001000000000 \\
S &= \,0.0000001000000 \\
\text{Correct Sum} &= \,0.0000001001000000000000
\end{aligned}$$

This constructed worst case achieves an error that is about half the error predicted by [1], and Theorem 3 shows that the bound of 2^{r-1} is optimal.

Remark. This is the same example as in [1], Section 8.8.1. Section 8.8.2 gives another example with $n = \bar{n} + 1$, $F \geq 2f$ and s_2 unnormalized: however, here again, the error is only 2^{r-1} ulps, and not $2^r + \frac{1}{2}$ ulps.

Note: the second case from 8.8.3 is just the same as above, with another summand $s_{n+1} = -2^{-f}$ that cancels the computed sum, while the true sum is 2^{-F}.

4.3 Case (4): $n \geq \bar{n} + 2$

Here is a generic example — from [1], multiplied by 2^F — where the computed sum is zero whereas the true sum is non zero for $F < 2f$: take $s_1 = 2^F + 2^f$,

$s_j = -2^f + 1$ for $2 \leq j \leq \bar{n}+1$, and $s_{\bar{n}+2} = -2^k$. Since $\bar{n} = 2^k + 1$ in that case, the true sum is $2^F + 2^f - (2^k + 1)(2^f - 1) - 2^k = 1$, whereas the computed sum is 0. Indeed, we have $t_2 = 2^F$ with a rounding error of 1, and then all subsequent computations are exact.

5 Implementation

Algorithm 1 was written in C using the *MPFR* library[6]. In order to compute error bounds the algorithm was first written straight from its definition in [1], then some improvements were added:

- the "double rounding" error is avoided as described in section 2.2,
- the precision F is automatically initialized from the precision f of the inputs,
- the precision F is then increased until the relative error given by theorem 1 is small enough to give the correct rounding.

Figure 1 shows some benchmarks made on a 1.4 GHz computer. The sorting algorithm used is *heap sort* and we don't use the automatic increase of the F parameter. We chose *heap sort* because it performed well – better than *quick-sort* – and because we didn't want to make assumptions about the number of different exponents present in the inputs. Further ideas on how to reduce the cost of the sorting by using for example a *bucket sort* may be found in [1].

6 Future Work

In this paper, we proposed an alternate proof of Demmel-Hida's main result about accurate summation [1]. This is a first step towards a formal proof using a proof assistant like HOL, PVS or Coq. While simplifying that proof, we improved some of the bounds given by Demmel and Hida (mainly in the $n = \bar{n} + 1$ case). We also provide some examples that attain the new bounds, giving evidence that those are optimal.

References

[1] Demmel, J., and Hida, Y. Accurate floating point summation. `http://www.cs.berkeley.edu/~demmel/AccurateSummation.ps`, May 2002. 37 pages.

[2] Demmel, J., and Hida, Y. Fast and accurate floating point summation with application to computational geometry, Jan. 2003. Submitted to Proceedings

n \ f	512	1024	2048	4096	8192	16384	32768	65536
512	46 0.19	42 0.22	35 0.30	20 0.66	6 2.34	3 4.52	1 9.09	0 17.51
1024	49 0.37	44 0.46	29 0.87	12 2.61	7 4.79	3 9.16	1 18.04	1 35.13
2048	50 0.77	36 1.42	20 3.16	12 5.52	7 9.85	4 18.45	2 36.16	1 69.09
4096	41 2.49	28 4.38	19 7.01	13 11.59	8 20.15	4 37.08	2 72.10	1 138.44
8192	35 6.89	27 10.03	20 15.05	13 23.89	8 40.83	4 74.75	2 145.07	1 279.39
16384	37 15.96	30 21.44	23 30.91	16 47.92	10 82.10	5 149.20	3 288.33	1 560.75
32768	47 33.67	40 43.97	32 62.98	23 96.59	15 163.97	9 298.63	5 576.36	
65536	50 68.82	44 88.97	36 125.79	26 192.95	17 327.27	10 597.27		
131072	52 139.46	45 176.76	37 256.02	28 388.63	18 650.90			
262144	53 281.07	46 363.86	38 505.68	29 768.63				
524288	53 566.59	47 727.5	39 1011.81					
1048576	54 1139.77	48 1458.63						
2097152	55 2270.90							

Fig. 1. Percentage of the computing time spent in the sorting phase (first line of each case) and total computing time in millisecond for several values of f (inputs precision) and n (number of input floating point numbers).

of 10th GAMM-IMACS International Symposium on Scientific Computing, Computer Arithmetic, and Validated Numerics (Scan 2002). 10 pages.

[3] IEEE standard for binary floating-point arithmetic. Tech. Rep. ANSI-IEEE Standard 754-1985, New York, 1985. approved March 21, 1985: IEEE Standards Board, approved July 26, 1985: American National Standards Institute, 18 pages.

[4] Kulisch, U., and Bohlender, G. Formalization and implementation of floating-point matrix operations. *Computing 16* (1976), 239–261.

[5] Priest, D. M. Algorithms for arbitrary precision floating point arithmetic. In *Proceedings of the 10th Symposium on Computer Arithmetic* (Grenoble, France, 1991), P. Kornerup and D. Matula, Eds., IEEE Computer Society Press, pp. 132–144.

[6] Spaces Project, INRIA. The MPFR Library `http://www.mpfr.org/`