Tight and rigourous error bounds for basic building blocks of double-word arithmetic

Mioara Joldes   Jean-Michel Muller   Valentina Popescu

Most of today's numerical computations are done using floating-point (FP) formats for representing real numbers. The two most widely implemented formats defined by the IEEE 754-2008 standard [1] for FP arithmetic are *single*-precision (*binary32*) and *double*-precision (*binary64*). For some critical problems, such precisions do not suffice. Since higher precisions, such as *quad*-precision (*binary128*) is not hardware implemented on widely distributed processors, the only solution is to use software emulation for extended precision.

There are mainly two ways of representing numbers in higher precision. The first one, the *multiple-digit* method, uses integers for representing numbers as a sequence of possibly high-radix digits coupled with an exponent. The second one, the *multiple-term* method, represents real numbers as unevaluated sums of several FP numbers, allowing for computations to be carried out naturally using the underlying FP hardware level. The later one is called a *floating-point expansion* when made up with an arbitrary number of terms and it makes use of well known *error-free-transforms* algorithms such as *2Sum, Fast2Sum* and *Fast2Mult* (see [2]).

In this work we focus on the "double-double" (DD) precision, i.e., the number x is represented as the sum of two *double*-precision FP numbers, $x_h + x_\ell$, the high and the low part, that satisfy $|x_\ell| \leq \frac{1}{2} ulp|x_h|$. As a matter of fact, we should better talk about "double-word" precision, since the algorithms and proofs we discuss can be used with any precision-p underlying FP format ($p = 53$ for *double*-precision). More precisely we talk about two algorithms that were first presented by Kahan et. al. in [3]. The first one (Alg. 1) computes the sum of two DD numbers and returns also a DD number. In the initial paper they claim to have a relative error bound of $2 \cdot 2^{-2p}$, but no rigorous proof is given. During our tests we observed that this bound is too optimistic, so we present here a slightly larger bound, $2^{-2p} \cdot (5 + 9 \cdot 2^{-p} + 7 \cdot 2^{-2p} + 6 \cdot 2^{-3p})$, for which we can provide a rigorous proof.

Alg. 2 computes the product of a DD number with a standard *double*-precision one and returns also a DD number. In this case we proved a tighter error bound that the one given in [3]. The initial error bound was $4 \cdot 2^{-2p}$ and we provide a proof for $2^{-2p}|xy| \left(3 + 4 \cdot 2^{-p} + 2 \cdot 2^{-2p}\right)$.

Algorithm 1 $(z_h, z_\ell) = (x_h, x_\ell) + (y_h, y_\ell)$.

```
1: $(s_h, s_\ell) \leftarrow \mathrm{2Sum}(x_h, y_h)$
2: $(t_h, t_\ell) \leftarrow \mathrm{2Sum}(x_\ell, y_\ell)$
3: $c \leftarrow \mathrm{RN}(s_\ell + t_h)$
4: $(v_h, v_\ell) \leftarrow \mathrm{Fast2Sum}(s_h, c)$
5: $w \leftarrow \mathrm{RN}(t_\ell + v_\ell)$
6: $(z_h, z_\ell) \leftarrow \mathrm{Fast2Sum}(v_h, w)$
7: return $(z_h, z_\ell)$
```

Algorithm 2 $(z_h, z_\ell) = (x_h, x_\ell) * y$.

```
1: $(c_h, c_{\ell 1}) \leftarrow \mathrm{Fast2Mult}(x_h, y)$
2: $c_{\ell 2} \leftarrow \mathrm{RN}(x_\ell \cdot y)$
3: $(t_h, t_{\ell 1}) \leftarrow \mathrm{Fast2Sum}(c_h, c_{\ell 2})$
4: $t_{\ell 2} \leftarrow \mathrm{RN}(t_{\ell 1} + c_{\ell 1})$
5: $(z_h, z_\ell) \leftarrow \mathrm{Fast2Sum}(t_h, t_{\ell 2})$
6: return $(z_h, z_\ell)$
```

References

[1] IEEE Computer Society, *IEEE Standard for Floating-Point Arithmetic*, IEEE Standard 754-2008, Aug. 2008.

[2] J.-M. Muller, N. Brisebarre, F. de Dinechin, C.-P. Jeannerod, V. Lefevre, G. Melquiond, N. Revol, D. Stehle, and S. Torres, *Handbook of Floating-Point Arithmetic*, Birkhauser Boston, 2010.

[3] X.S.Li, J.W.Demmel, D.H.Bailey, G.Henry, Y.Hida, J.Iskandar, W.Kahan, A.Kapur, M.C.Martin, T.Tung, D.J.Yoo, *Design, Implementation and Testing of Extended and Mixed Precision BLAS*, ACM TOMS, vol. 28, no. 2, 2002.

[4] M. Joldes, J.-M. Muller, and V.Popescu, *Tight and rigourous error bounds for basic building blocks of double-word arithmetic*, https://hal.archives-ouvertes.fr/hal-01351529.