Logarithm with correct rounding

Master Thesis

Master in *Sciences and Technologies*,
Specialty in *Mathematics*,
Cryptology and Computer Security.

Author
Sidali Zitouni-Terki <sid-ali.zitouni-terki@etu.u-bordeaux.fr>

Supervisor
Paul Zimmermann <Paul.Zimmermann@inria.fr>

Tutor
Gilles Zémor <gilles.zemor@u-bordeaux.fr>

september 2nd 2022

Declaration of authorship of the document

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of the Master in *Sciences and Technologies*, Specialty in *Mathematics* or *Computer Science*, Track *Cryptology and Computer Security*, is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Date and Signature

Remerciement

Remerciements particuliers à mon superviseur Paul Zimmermann pour son aide et son soutien. Je le remercie également pour sa patience à mon égard.
Je remercie mon tuteur Gilles Zémor et tous les autres enseignants pour leurs apports théoriques et méthodologiques et leurs partages de connaissances qui m'ont fait bénéficié tout au long de mon parcours universitaire.
Je tiens également à exprimer mes sincères remerciements et ma gratitude aux membres du jury pour leur lectures et leur évaluation du travail.
Je remercie tout particulièrement :
Mes chers parents pour leur motivation et leur soutien constant et à qui je souhaite une longue vie pleine de santé.
Ma chère femme et ma belle-famille dont les mots d'encouragement m'ont motivé à reprendre et continuer mes études.
Mes filles bien-aimées, Farah et Tesnime.
Mes sœurs et mon frère qui m'ont soutenu dans ce travail.
Mes amis avec qui j'ai partagé des souvenirs inoubliables.

Résumé

L'implémentation des fonctions mathématiques avec arrondi correct est un sujet important en arithmétique flottant. Accéder à de telles fonctions nous permettrais d'avoir plus de précisions sur les calculs avec des nombres flottants. Nous avons des algorithmes, comme **Fast2Sum** et **DEKKER-PRODUCT**, qui nous aide à avoir plus de précisions. Grâce à ces algorithmes, nous pourrions implémenter le logarithme avec arrondi correct et avoir une fonction qui approche aux mieux au résultat du logarithme.
Ce rapport nous montre d'abord: comment on peut utiliser les quatre modes d'arrondis qui sont l'arrondi au plus proche, arrondi au plus proche de zéro, arrondi au plus proche de l'infini et l'arrondi au plus proche de moins l'infini, et comment a été calculé la précision de chaque algorithme utilisé pour le logarithme. Ensuite, nous expliquons comment cette fonction de mathématique a été implanté. Ce logarithme avec arrondi correct a réussi le test sur le millions de pires cas.

Abstract

The implementation of mathematical functions with rounding correct is an important topic in floating point arithmetic. Access to such functions would allow us to have more precision on the calculations with floating numbers. We have algorithms, like **Fast2Sum** and **DEKKER-PRODUCT**, which help us to have more precisions. Due to these algorithms, we could implement the logarithm with rounding correct and have a function that best approximate the result of the logarithm. This report shows us first: how one can use the four modes roundings which are rounding to the nearest, rounding to the nearest to zero, rounding to nearest infinity and rounding to nearest minus infinity, and how was calculated the accuracy of each algorithm used for the logarithm. Inaddition,it explains how this function of mathematics were implanted. This logarithm with correct rounding passed the test on the millions of worst cases.

Contents

Introduction

. This thesis aims to explain in a mathematical way the steps for the implementation of the logarithm with correct rounding on floating numbers. This function is one of the functions of the COREMATH[1] project.
This project implements mathematical functions with correct rounding to be able to integrate into mathematical libraries for the new revision of the standard $IEEE754$.

The definition of correct rounding given by the $IEEE754$ standard is as follows "Given a mathematical function f and a floating point number x, the correct rounding of $f(x)$ is the floating number y closest to $f(x)$ according to the given rounding mode (nearest, towards zero, towards $-\infty$ or towards $+\infty$)". This standard imposes the correct rounding for the four elementary arithmetic operations which are addition, subtraction, multiplication and division. But it does not impose for mathematical functions. For now, there is no mathematical library that gives us exactly the correct rounding.

This project already has the implementation of this logarithm for single precision ($binary32$ format of $IEEE754$). The calculation steps of our logarithm will be in function of double precision ($binary64$ format of $IEEE754$). We will use basic floating point algorithms which are **FastSum** and **DEKKER-PRODUCT**.

My research paper will be devited into five chapters.
In the first chapter, we explain floating point and the $IEEE754$ standard with some definitions. Then we integrate some arithmetic tools that will be used for calculations in chapters 4, and 5. Then we give some notation rules.

In the second chapter, we detail the steps of the $cr\log$. first, we talk about the special cases. Then, we explain the argument reduction with the algorithms of **Tang** and **Gal**. After, we calculate and evaluate the ap-

[1] https://core-math.gitlabpages.inria.fr

proximation polynomial thanks to the formula of **Taylor**.

For the third chapter, we define the addition and multiplication algorithms which will give us results in **Double-Double** type and calculate their relative errors for each of these functions.

In the fourth chapter, we define other addition and multiplication algorithms which give us results in **Triple-Double** type and we calculate their relative errors.

The last chapter explain how the logarithm will unfold with its three *cr* log.

Chapter 1

Floating point arithmetic

The next defintion take from [14], [16] and [19].

1.1 Definitions

Definition 1 *: **Floating point** is a way of writing real numbers. This expression is mostly used for computers. It is different from the fixed point. It is represented with a precision, base(radix), sign, mantissa and exponent. Let x be a real number, so in this way we have $x = s.M.\beta^{e-p+1}$ with s the **sign**, M the **mantissa**, β a **base(radix)**, e the **exponent** and p the **precision**.*

p represents the number of bits of the **mantissa**. In our research, we prefer to use $E = e - bias$ to be possible to have a negative exponent and also to have $E_{min} = 1 - E_{max}$.

Floating points are composed of several precisions given by the **IEEE 754** standard.

The next defintion take from [21].

Definition 2 *: **IEEE 754** is a floating point standard created by **the Institute of Electrical and Electronics Engineers**. It is the most used standard for computer calculations . This standard imposes formats for each floating number with its representations of sign, mantissa, exponent and also the five rounding modes used.*

IEEE 754-2008 defines three possible formats for base of 2 of **floating point**: **Binary32**, **Binary64** and **Binary128**.

- **Binary32** (**single precision**) (Figure 1.1) :
  Binary32 is stored on 32 bits, with 1 bit for the **sign**, 8 bits for the **exponent** and 23 bits for the **mantissa** without its strong bit (24 bits).

- **Binary64** (**double precision**) (Figure 1.2) :
  Binary64 is stored on 64 bits, with 1 bit for the **sign**, 11 bits for the **exponent** and 52 bits for the **mantissa** without its strong bit (53 bits).

- **Binary128** (**Quadruple precision**) (Figure 1.3) :
  Binary128 is stored on 128 bits, with 1 bit for the **sign**, 15 bits for the **exponent** and 112 bits for the **mantissa** without its strong bit (113 bits).

In the 3 Schematics below of the 3 formats, we have the leftmost bit is the strong bit and the rightmost is the weakest bit:

Figure 1.1: Binary32

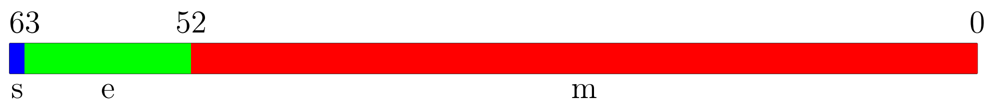

Figure 1.2: Binary64

Figure 1.3: Binary128

We also have **Binary80** which is an **extended double precision** and two others possible formats for base of 10:

- **decimal64** is stored on 64 bits.

- **decimal128** is stored on 128 bits.

In **C** programming, **Binary32** represents **Float**, **Binary64** is **Double**, **Binary80** is **long-double** and **Binary128** is **float128**.
Throughout our research, we will only use **Binary64**.
The real numbers are transformed into a floating number and we need to round them.
According to the **IEEE 754** standard, there are four rounding modes among them, three are direct rounds:

- **RNDN**: Rounding to nearest.
  It has two possibilities and the only difference between them is if the number falls in the midway.
  - **Ties to even**: it rounds to the nearest even floating number.
  - **Ties to away**: it rounds the nearest to the largest absolute value.
- **RNDZ**: Directed Rounding to 0 (truncation).
- **RNDU**: Directed Rounding to $+\infty$ (rounding up or ceiling).
- **RNDD**: Directed Rounding to $-\infty$ (rounding down or floor).

The examples below shows us the four rounding modes.

Real Number	+140.8215064611465	+665.5752955525412
RNDN(Ties to even)	+140.821506461146	+665.575295552541
RNDN(Ties to away)	+140.821506461147	+665.575295552541
RNDZ	+140.821506461146	+665.575295552541
RNDU	+140.821506461147	+665.575295552542
RNDD	+140.821506461146	+665.575295552541

Table 1.1: roundings with positive reals

We have the special values:

- there exist two 0: $+0$ and -0
- Infinity : $-\infty$ and $+\infty$
- NaN (Not a Number): result of the invalid operations.(for example :0/0, $\sqrt{-7}$,..)

Real Number	-140.8215064611465	-665.5752955525412
RNDN(Ties to even)	-140.821506461146	-665.575295552541
RNDN(Ties to away)	-140.821506461147	-665.575295552541
RNDZ	-140.821506461146	-665.575295552541
RNDU	-140.821506461146	-665.575295552541
RNDD	-140.821506461147	-665.575295552542

Table 1.2: roundings with negative reals

- **Subnormal** Number: it is a number which has as exponent $= 0$ and a pseudo-mantissa. (difference between the Mantissa and the pseudo-mantissa those which does not have a hidden 1)
- **Overflow** : according to [1] "This exception is caused when the result is too large in absolute value to be represented".
- **Underflow** : according to [1] "This exception is caused when the actual result is too small in absolute value to be represented in the chosen format".

Definition 3 *: **ulp(x)***
*According to **William Kahan** (1960) in [19] The original definition of ulp() by "ulp(x) is the gap between the two floating-point numbers nearest to x, even if x is one of them".*
*ulp(x) is the acronym of **Unit of the Last Place**.*
*In the following we will use **Goldberg**'s definition:*
*Let x a **floating-point** number,* $x = d_0 d_1 d_2 d_3 d_4 ... d_{p-1} \beta^e$*. we therefore have an error to represent it :*$|d_0 d_1 d_2 d_3 d_4 ... d_{p-1} - \frac{x}{\beta^e}|$ *which is the unit of the place.*

Corollary 1 *If x is a **floating-point** number, so we have :*

$$ulp(x) \leq 2^{-52}|x|$$

.

1.2 Necessary tools

1.2.1 properties

The properties are from [19].

Property 1 *If* $X = RNDN(x) \Rightarrow |X - x| \leq \frac{1}{2}ulp(x)$

Property 2 *If* $X = RNDN(x) \Rightarrow |X - x| \leq \frac{1}{2}ulp(X)$

Property 3 *If* $X \in \{RNDD(x), RNDU(x), RNDN(x)\} \Rightarrow |X - x| \leq ulp(x)$

Property 4 *If* $X \in \{RNDD(x), RNDU(x), RNDN(x)\} \Rightarrow |X - x| \leq ulp(X)$

Property 5 *If* $x > 0$ *then* $RNDZ(x) = RNDD(x)$.
If $x < 0$ *then* $RNDZ(x) = RNDU(x)$.

We need to calculate the relative error to know the accuracy of our algorithms.

Corollary 2 *If* $|X - x| \leq \alpha.ulp(x)$ *with* $\alpha \in \mathbf{R} \Rightarrow \frac{|X-x|}{|x|} \leq \alpha.2^{-52} \Rightarrow |X| \leq |x|(1 + \alpha.2^{-52})$.

Proof We suppose $|X - x| \leq \alpha.ulp(x)$, as $ulp(x) \leq 2^{-52}.|x|$ so we have:

$$|X - x| \leq \alpha.2^{-52}.|x|$$

$$\frac{|X - x|}{|x|} \leq \alpha.2^{-52}$$

According to the triangle inequality, we have that $|X - x| \geq |X| - |x| \Rightarrow$

$$|X| - |x| \leq \alpha.2^{-52}.|x|$$

$$|X| \leq \alpha.2^{-52}.|x| + |x|$$

$$|X| \leq |x|(1 + \alpha.2^{-52}).$$

Also need to calculate with $ulp(X)$

Corollary 3 *If* $|X - x| \leq \alpha.ulp(X)$ *with* $\alpha \in \mathbf{R} \Rightarrow |X| \leq |x| + \alpha.ulp(X)$.

PROOF We suppose $|X - x| \leq \alpha.ulp(x)$, as $ulp(X) \leq 2^{-52}.|X|$ so we have:

$$|X - x| \leq \alpha.2^{-52}.|X|$$

$$\frac{|X - x|}{|X|} \leq \alpha.2^{-52}$$

According to the triangle inequality, we have that $|X - x| \geq |X| - |x| \Rightarrow$

$$|X| - |x| \leq \alpha.2^{-52}.|X|$$

$$|X| \leq |x| + \alpha.2^{-52}.|X|$$

According to the collary 3, we have that $ulp(x) \leq 2^{-52}.|x|$ then $ulp(1) = 2^{-52}$.

The notations of the next part which are drawn from [19], we will simplify the calculations of precisions of our algorithm using **unit Roundoff**.

1.2.2 Unit Roundoff

The **unit Roundoff** has as **base** 2 and as **precisions** 53, so we have that:

$$u = \begin{cases} \frac{1}{2}.ulp(1) = \frac{1}{2}.2^{-52} = 2^{-53} & for \quad RNDN \\ \\ ulp(1) = 2^{-52} & for \quad (RNDU, RNDZ, RNDD) \end{cases}$$

We can do only one calculation and at the end, we can replace u by its values for each rounding mode.

Corollary 4 *If $X = \circ(x)$ according to the properties 1, 2, 3 and 4 , we have his results:*

- $|X - x| \leq u.|x|$
- $|X - x| \leq u.|X|$

Corollary 5 *If $X = \circ(x) \Rightarrow |X| \leq |x|(1 + \epsilon)$ with $|\epsilon| \leq u$.*

Corollary 6 *If $X = \circ(x) \Rightarrow |X| \leq |x| + \epsilon$ with $|\epsilon| \leq u.|X|$.*

To proof the exactness of our operation, we use **Sterbenz**'s lemma.
The next subsection is taken from [19].

1.2.3 Sterbenz's lemma

Lemma 1 (Sterbenz) *In a **radix**[1]-β floating-point system with **subnormal** numbers available, if x and y are finite **floating-point** numbers such that $\frac{y}{2} \leq x \leq 2.y$, then $x - y$ is exactly representable.*

$\beta = 2$ for our research.
Lemma of **Sterbenz** implies for the 4 rounding modes so the result is exact.

Proof According to the technique of proof of Sterbeinz's lemma from [19]. We suppose that $x \geq 0$, $y \geq 0$ and $y \leq x \leq 2.y$.
Let $x = M_x.\beta^{e_x-p+1}$ and $y = M_y.\beta^{e_y-p+1}$ with their **exponents** (e_x, e_y) and their **mantissa** (M_x, M_y). We have:

$$\begin{cases} e_{min} \leq e_x \leq e_{max} \\ e_{min} \leq e_y \leq e_{max} \\ 0 \leq M_x \leq \beta^p - 1 \\ 0 \leq M_y \leq \beta^p - 1 \end{cases}$$

Firstly, we assume that $e_y \leq e_x$, we define $\lambda = e_x - e_y$. So we have :

$$x - y = (M_x.\beta^\lambda - M_y).\beta^{e_y-p+1}$$

We define $M = M_x.\beta^\lambda - M_y$.
According to the conditions at the beginning of the proof, we have:

- $x \geq y \Rightarrow x - y \geq 0 \Rightarrow M.\beta^{e_y-p+1} \geq 0$ as $\beta^{e_y-p+1} > 0 \Rightarrow M \geq 0$.
- $x \leq 2.y \Rightarrow x-y \leq y \Rightarrow M.\beta^{e_y-p+1} \leq M_y.\beta^{e_y-p+1} \Rightarrow M \leq M_y \leq \beta^p - 1$

Secondly, we suppose that $M_y < M_x$ and that $e_y > e_x$, we define an other $\lambda = e_y - e_x$.

$$x - y = (M_x - M_y.\beta^\lambda).\beta^{e_x-p+1}$$

We define $M = M_x - M_y.\beta^\lambda$.
According to the conditions at the beginning of the proof, we have:

- $x \geq y \Rightarrow x - y \geq 0 \Rightarrow M.\beta^{e_x-p+1} \geq 0$ as $\beta^{e_x-p+1} > 0 \Rightarrow M \geq 0$.

[1] base

- $x \leq 2.y \Rightarrow x - y \leq y \Rightarrow M.\beta^{e_x-p+1} \leq M_y.\beta^{e_y-p+1} \Rightarrow M \leq M_y \leq \beta^p - 1$

We have for our 2 cases, the same result that is $x - y = M.\beta^{e-p+1}$ with $e_{min} \leq e_x \leq e_{max}$ and $M \leq \beta^p - 1$.
We have proved that $x - y$ is a floating point number so the calculation is exact.

1.3 Notations

- Function names are composed of 3 letters followed by 3 numbers. If we have Add then it's an addition and if we have Mul then it's a multiplication.
  The first 2 numbers of the 3 represent the argument and the last number represent the output of this function. The number 1 is a **Double** number, 2 is a **Double-Double** and 3 is a **Triple-Double**.

- $RNDN$, $RNDU$, $RNDD$ and $RNDZ$ will be rounded as explained before.(table 1.1 and table 1.2)

- $\circ()$ will represent all roundings.

- $cr\log_{fast-path}$, $cr\log_{accurate-path}$ and $cr\log_{advanced-accurate-path}$ for the sake of writing, we will change them to $cr\log_{fast}$, $cr\log_{accurate}$ and $cr\log_{advanced}$.

- **Double-Double** is a pair of **Double** numbers, **Triple-Double** is a triple of **Double** numbers.

In our research, we use as base $\beta = 2$, precision $p = 53$, $bias = 1023$, $E_{min} = -1022$ and as $E_{max} = 1023$.
The next chapter will be devited to calculate $cr\log$.

Chapter 2

Steps to calculate crlog

Before starting to implement $cr\log$, we will see what are its steps.
According to [13], first, we filter the special cases. Then, we reduce the argument range. After, we use the polynomial approximation and evaluation.

2.1 Special cases

In the special cases, we have five possibilities:

- input is NaN return NaN
- Input is negatif return NaN
- Input is $+0$ or -0 return $-\infty$
- Input is $+\infty$ return $+\infty$
- Input is a **subnormal** number. We transform it into **normal** number then we calculate as if it is normal.

2.2 Argument reduction

We know that $\log(x.y) = \log(x) + \log(y)$. We have in $Input = 2^E \times m$ with E is the **exposant** and m is the **mantissa**.
So we have $\log(Input) = E \times \log(2) + \log(m)$.
First, we calculate $\log(m)$ with **Tang**'s algorithm and **Gal**'s algorithm.

2.2.1 Tang's Algorithm

With $x = m * 2^e$ and so that $\log(x) = \log(m) + e * \log(2)$
First, we calculate $\log(m)$ with **Tang**'s Algorithm ([13]) .
We have two possibilities: either we make a single reduction or two reductions with this algorithm.

```
# The random() function only gives results that are between 0 and 1.
# We add 1 so that 1 <= m
```

```
m = 1 +random();m
```

```
1.672168120969894
```

Figure 2.1: m taken in random to test

2.2.1.1 Tang's algorithm with a single reduction

We have : $1 \leq m < 2$ and we want to reduce $\log(m)$ using **Tang**'s algorithm. First , we use k significant bits after the initial 1. Then, we take i which represents the integer of the k bits:
So we have: $0 \leq i < 2^k$ et $1 + \frac{i}{2^k} \leq m < 1 + \frac{i+1}{2^k}$
We look for α_i such as $1 \leq m * \alpha_i < 1 + \epsilon_i$ with ϵ_i as small as possible. We can write $m = \frac{m*\alpha_i}{\alpha_i}$ and therefore have $\log(m) = \log(m * \alpha_i) - \log(\alpha_i)$. After calculation, we have $\alpha_i = \frac{2^k}{2^k+i}$ and $\epsilon_i = \frac{1}{2^k+i}$ as $0 \leq i < 2^k$ then the value $max(\epsilon_i) = \frac{1}{2^k}$.

If we take $m^{'} = m * \alpha_i$, so we have $1 \leq m^{'} < 1 + \frac{1}{2^k}$. At the end of this algorithm, $m^{'}$ has it's k first zero bits after the initial 1.

The value to be calculated has been reduced to $\log(m) = \log(m^{'}) - \log(\alpha_i)$ with $-\log(\alpha_i)$ which is already calculated and memorized in a table.

Approach thanks to the Tang's algorithm with a single reduction

```
print("representation of m in binary: ",RR(m).str(2))
k = 7
binary = RR(m).str(2)[2:9] # k = 7, we take the 7 bits after initial 1
print("the k bits taken : ",binary)
i = ZZ(binary,2)     # i is integer of k bits
print("i is integer of k bits: ", i)
# We calculate with the formula alpha_i = (2^k)/(2^k + i)
alpha_i = RR(2^k/(2^k +i))
print("alpha_i = ",alpha_i)
# mprime = m*alpha_i  and we know that 1 <= mprime < 1+ epsillon_i
# with epsillon_i = 1/(2^k + i)
epsillon_i = RR(1/(2^k + i))
print("the maximum d'alpha_i not reached : ", epsillon_i)
mprime = m*alpha_i # it can be seen that 1 <= mprime < 1 + epsillon_i
print("mprime = ",mprime)
print("we see the first k bits are set to zero")
print(RR(mprime).str(2))
```

```
representation of m in binary:  1.1010110000010011001101011100000011111010101111100101
the k bits taken :  1010110
i is integer of k bits:  86
alpha_i =  0.598130841121495
the maximum d'alpha_i not reached :  0.00467289719626168
mprime =  1.00017532469227
we see the first k bits are set to zero
1.0000000000001011011111010111010111010001110010100101
```

Figure 2.2: The first reduction with $k = 7$

2.2.1.2 Tang's algorithm with two reductions

After the first reduction explained above, another reduction is made again with the same value of k for the following bits of the first reduction. Then, we take j which represents the integer of the k bits. So we have : $0 \leq j < 2^k$ and $1 + \frac{j}{2^{2k}} \leq m^{'} < 1 + \frac{j+1}{2^{2k}}$.

Now, we are looking for β_j such as $1 \leq m^{'} * \beta_j < 1 + \epsilon_j^{'}$ with $\epsilon_j^{'}$ as small as possible. We have that $\log(m^{'}) = \log(m^{'} * \beta_j) - \log(\beta_j)$.
After calculation : $\beta_j = \frac{2^{2k}}{2^{2k}+j}$ and $\epsilon_j^{'} = \frac{1}{2^{2k}+j}$.
As we have $m^{"} = m^{'} * \beta_j$ then $1 \leq m^{"} < 1 + \frac{1}{2^{2k}}$.
So, we have $\log(m) = \log(m^{"}) - \log(\beta_j) - \log(\alpha_i)$.
$(-\log(\alpha_i))$ and $(-\log(\beta_j))$ are already calculated and are put into memory in a table.

Approach thanks to the Tang's algorithm with a second reduction.

```
print("representation of mprime in binary: ",RR(mprime).str(2))
k = 7
binary = RR(mprime).str(2)[9:16] # k = 7, we take the 7 bits after initial 1
print("the k bits taken : ",binary)
j = ZZ(binary,2)     # j is integer of k bits
print("j is integer of k bits: ", j)
# We calculate with the formula beta_j = (2^2k)/(2^2k + j)
beta_j = RR(2^(2*k)/(2^(2*k) +j))
print("beta_j = ",beta_j)
# msecond = mprime*beta_j  and we know that 1 <= msecond < 1+ epsillonprime_j
# avec epsillonprime_j = 1/(2^(2k) + j)
epsillonprime_j = RR(1/(2^(2*k) + j))
print("the maximum d'alpha_i not reached : ", epsillonprime_j)
msecond = mprime*beta_j;msecond # on voit bien 1 <= msecond < 1 + epsillonprime_j
print("mprime = ",msecond)
print("we see the first 2k bits are set to zero")
print(RR(msecond).str(2))
```

```
representation of mprime in binary:  1.000000000000101101111101011101011101000111001010010 1
the k bits taken :  0000010
j is integer of k bits:  2
beta_j =  0.999877944586842
the maximum d'alpha_i not reached :  0.0000610277065787868
mprime =  1.00005324787979
we see the first 2k bits are set to zero
1.0000000000000011011111010101100111100110111110110010
```

Figure 2.3: The second reduction with $k = 7$

2.2.2 Gal's method with double precision

Gal's method is taken from [8].
We have $x = y * 2^n$ with n integer and $0.75 \leq y < 1.5$
So $\log(x) = \log(y) + n * \log(2)$.
We calculate $\log(y)$ using a table of triplets $(X_i, \log(X_i), \frac{1}{X_i})$ with $0 \leq i \leq 192$.
$X_i = 0.75 + \frac{i}{256} + \frac{1}{512} + E_i$ (with E_i a very small number).
$\log(y) = \log(\frac{X_i * y}{X_i}) = \log(X_i) + \log(1 + \frac{y - X_i}{X_i}) = \log(X_i) + \log(1 + z)$
If X_i is choosen close to y, we have $\frac{-1}{384} < z < \frac{1}{384}$
and if more y is close to 1 so $\frac{-1}{512} < z < \frac{1}{512}$.
We use an approximation polynomial $p(z)$ of degree 6 (for double precision) to approach the function $\log(1 + z)$.
If x is close to 1 so we use an approximation polynomial of $\log(x)$ without the table with a relative error of 2^{-72} which is negligible.
The table of the triplets $(X_i, \log(X_i), \frac{1}{X_i})$ contains 576 elements.
$F_i = \log(X_i)$ and $G_i = \frac{1}{X_i}$ with $0 \leq i \leq 192$.

The numbers X_i are choosen so that F_i and G_i are 56 bits of mantissa. And they have a relative precision of 2^{-65}.
We search to be close to $0,75 + \frac{i}{256} + \frac{1}{512}$ with X_i such that bits 57 to 67 of the mantissa of $\log(X_i)$ and of $\frac{1}{Xi}$ which will be reset either all to 0 or

all to 1. That's why small numbers E_i were introduced. F_i and G_i are the numbers with double precision obtained by a calculation of extended precisions and a symmetric rounding.

2.2.3 method with Tang algorithm and Gal method

Our routine involves the two algorithms seen previously.
We start using **Tang**'s algorithm.
at the beginning as explained in 2.2.1, we only have $x = m*2^e$ and therefore we have $\log(x) = \log(m) + e * \log(2)$.
As mentioned in 2.2.1, we recover α_i with the calculation:
$\alpha_i = \frac{2^k}{2^k+i}$ except for the following we will take $\alpha_i^{'}$ a double close to α_i such that the $\log(\alpha_i^{'})$ is the closest to a double .
This ϵ_i will be used to search for the $\log(\alpha_i^{'})$ such that the bits from 54^{th} to 71^{st} are identical. (To have an approximation approximately 2^{-71}). (Figure 2.4)

method with Tang's algorithm and the Gal method

```
k=8 # we take k = 8,
m=1+random() # we take 1<m<2
M = R200(m).exact_rational()
binary = RR(m).str(2)[2:10] #we recover the 8 bits after the initial 1.
i = int(binary,2)         # i is the 8-bit integer
alpha_i_m = R200(table_alpha_modified[i],16).exact_rational() #table calculated for all i of alpha_i_m
                                             # such that log(alpha_i_m) has a precision of 71 bits
log_alpha_i_m= R200(table_log_alpha_modified[i],16).exact_rational()
A=R200(log(M*alpha_i_m)+log_alpha_i_m) #
B = R200(log(R200(M)))
print("m =",m)
print("i =",i)
print ("alpha_iprime = ",alpha_i_m )
print("log_alpha_i_prime :",log_alpha_i_m)
print("sol    =" ,A)
print("log(m)=",B)
C = R200((A.exact_rational()-B.exact_rational())/B.exact_rational())
print("the precision for the modified method is %f bits" %(R200(-log(abs(C))/log(2.0))))
```

```
m = 1.284464826386863
i = 72
alpha_iprime =  7030009188575477/9007199254740992
log_alpha_i_prime : 8929238770791491/36028797018963968
sol    = 0.25034215407638279200093780192379456668445060635973530859479
log(m)= 0.25034215407638279200088487247857236200406769824813103788058
the precision for the modified method is 72.002249 bits
```

Figure 2.4: The revised method of **Tang** and **Gal**

We are going to verify if we obtain the same result with the **Tang** algorithm and our modified method (Figure 2.5).

Verification of our algorithm with that of Tang

```
k=8 # we take k=8
# m=1+random() # we take 1<m<2
M = R200(m).exact_rational()
binary = RR(m) .str(2) [2:10]
i = int (binary,2)# we recover the 8 bits after the initial 1.
(h1,l1) = table_alpha_i[i] # h1 is the main value and l1 is the error value
(H1,L1)= (R200(h1,16).exact_rational(),R200(l1,16).exact_rational())
(h2, l2)= table_log_alpha_i[i]# h2 is the main value and 12 is the error value
(H2,L2)= (R200(h2,16).exact_rational(),R200(l2,16).exact_rational())
alpha_i_m = R200(table_alpha_modified[i],16) #table calculated for all i of alpha_i_m
                                              # such that log(alpha_i_m) has a precision of 71 bits
log_alpha_i_m= R200(table_log_alpha_modified[i],16)
sol_Tang = R200(log(M*(H1+L1))+(H2+L2))
sol_rema=R200(log(M*alpha_i_m)+log_alpha_i_m)
A = R200(log(R200(M)))
print("m = ",m)
print("i =", i)
print("'alpha,i.m =", (h1, h2))
print("alpha_1_m :",alpha_i_m)
print ("solution of Tang's method = ", sol_Tang)
print ("solution of the reworked method :                ",sol_rema)
print("log(m) =                          ", A)
C = R200((sol_Tang.exact_rational()-A.exact_rational())/A.exact_rational())
D = R200((sol_rema.exact_rational()-A.exact_rational())/A.exact_rational())
print("the precision for Tang's method is %f bits" %(R200(-log(abs(C))/log(2.0))))
print("the precision for the reworked method is %f bits" %(R200(-log(abs(D))/log(2.0))))
```

```
m =  1.486258461986336
i = 124
'alpha,i.m = ('0x1.58ed2308158edp-1', '0x1.947941c2116fbp-2')
alpha_1_m : 0.67368421158545022109365163487382233142852783203125000000000
solution of Tang's method =  0.39626186252142292454373873382587912813786396912842268809010
solution of the reworked method :                0.39626186252142292454384333370528885498555363956773361536439
log(m) =                          0.39626186252142292454373873382587662034589290447866241791757
the precision for Tang's method is 106.961735 bits
the precision for the reworked method is 71.682063 bits
```

Figure 2.5: Verification of the revised method and **Tang**'s method

2.3 polynomial approximation and evaluation

Before calculating the approximation function with the **Sollya** tool, we will look for the interval for which the polynomial will be effective for $m * \alpha_i^{'}$ with $0 \leq i < 256$ and $1 + \frac{i}{256} \leq m < 1 + \frac{i+1}{256}$. The calculations are experimented on **sage**, see the diagram 2.6:

Calculation of the interval for the approximation polynomial

```
Linf = [] # the lower bounds for each m*alpha_i_m
Lsup = [] # the upper bounds for each m*alpha_i_m
for i in range(256):
    Linf.append(R200((1+i/256)*R200(table_alpha_modified[i],16).exact_rational()))
    Lsup.append(R200((1+(1+i)/256)*R200(table_alpha_modified[i],16).exact_rational()))
print("the upper bounds is : ", max(Lsup).exact_rational())
print("the lower bounds is : ", min(Linf).exact_rational())
```

```
the upper bounds is :  257/256
the lower bounds is :  1
```

Figure 2.6: Interval calculation for the approximation polynomial

We find as result $1 < m.\alpha_i^{'} < \frac{257}{256}$, exactly the same bounds as we calculated with the **Tang** method.
At the end of the calculation of the modified algorithm from Tang and Gal's algorithm, we have that $\log(input) = \log(m.\alpha_i) - \log(\alpha_i) + E.\log(2)$. We transform $\log(m.\alpha_i)$ into $\log(1+t)$ to use the approximation function; then we have $t = m.\alpha_i - 1$.
We search the polynomial approximation for $\log(1+t)$ thanks to the **Taylor** formula.
Let $P(t)$ the polynomial approximation for $\log(1+t)$, we have $P(t) \approx t - \frac{t^2}{2}...$, in case we have a constant term, we can reduce it to 0.
This calculation will be done thanks to the **Sollya** tool with its **fpminimax** function with the calculated interval.
Then we evaluate this approximation function with the argument $t = m.\alpha_i - 1$.
Now, we have the operation $\log(input) = P(m.\alpha_i - 1) - \log(\alpha_i)$(Figure 2.7).

Our method with the Tang algorithm and the Gal method with the P function

```
k=8 # we take k = 8
#m=1+random() # We take 1<m<2
M = R200(m).exact_rational()
binary = RR(m).str(2)[2:10] # we recover the 8 bits after the initial 1.
i = int(binary,2)         # i is the 8-bit integer
alpha_i_m = R200(table_alpha_modified[i],16).exact_rational() #table calculated for all i of alpha_i_m
                                               # such that log(alpha_i_m) has a precision of 71 bits
log_alpha_i_m= R200(table_log_alpha_modified[i],16).exact_rational()
A=R200(P(M*alpha_i_m-1)+log_alpha_i_m) #
B = R200(log(R200(M)))
print("m =",m)
print("i =",i)
print ("alpha_iprime = ",alpha_i_m )
print("log_alpha_i_prime :",log_alpha_i_m)
print("sol   =" ,A)
print("log(m)=",B)
C = R200((A.exact_rational()-B.exact_rational())/B.exact_rational())
print("the precision for the reworked method is %f bits" %(R200(-log(abs(C))/log(2.0))))
```

```
m = 1.019019152905743
i = 4
alpha_iprime =  8868626965695373/9007199254740992
log_alpha_i_prime : 8937554567599383/576460752303423488
sol   = 0.018840549849761655851050549366714043792850403560626948245975
log(m)= 0.018840549849761655851046655024229337473701007314919333872294
the precision for the reworked method is 72.034879 bits
```

Figure 2.7: The revised method of **Tang** and **Gal** with $P(t)$

Before talking about the implementation of $cr_{\log}$, we will first see the used algorithms.

The algorithms of the chapter 4 and chapter 5 are taken from [4] and [12].

Chapter 3

Operators on Double-Double numbers

3.1 Addition Operators

3.1.1 Add112

See algorithm 1

Lemma 1 (Add112) *Let a and b floating point numbers, with $|a| \geq |b|$, s and t result of $Add112(a, b)$ for the 4 modes of rounding, considering that there is no* ***overflow*** *so :*

- *(1) $s + t$ is exactly equal to $a + b$.*
- *(2) $|t| \leq 2^{-53}|s|$ for $RNDN$ and $|t| \leq 2^{-52}|s|$ for direct rounding modes.*

PROOF (1):
According to the calculation technique of **Fast2Sum algorithm** proof's ([19]), this proof shown that for the closest rounding mode ($RNDN$).
We suppose that $|a| \geq |b|$ and there is no **overflow**.
We suppose $a > 0$ and $b > 0$ (respectively $a < 0$ and $b < 0$).
We take $b = b_h + b_\ell$ with b_h a multiple of $ulp(a)$,$|b_\ell| < ulp(a)$ and b_ℓ is of the same sign that b (and b_h).

Either we have $0 \leq |b_\ell| < 2^{-53}.|a|$ or $2^{-53}.|a| \leq |b_\ell| \leq 2^{-52}.|a|$.
If $0 \leq |b_\ell| < 2^{-53}.|a|$ for the modes of Rounding $RNDZ$, $RNDN$ and $RNDD$ (respectively $RNDZ$, $RNDN$ and $RNDU$) then b_ℓ will be ignored otherwise $RNDU$(respectively $RNDD$) b_ℓ is not ignore, so $\circ(b) =$

$b_h + ulp(a)$;
If $2^{-53}.|a| \leq |b_\ell| \leq 2^{-52}.|a|$ for the modes of Rounding $RNDZ$ and $RNDD$ (respectively $RNDZ$ and $RNDU$) then b_ℓ will be also ignored for $RNDN$(closest round) and $RNDU$(respectively $RNDN$ and $RNDD$).
The first case when b_ℓ is ignored so we have $\circ(b) = b_h$. $\Rightarrow$:

- $s = \circ(a + b) \Rightarrow s = a + b_h$
- $z = \circ(s - a) \Rightarrow z = a + b_h - a \Rightarrow z = b_h$
- $t = \circ(b - z) \Rightarrow t = b - b_h$ we have that $b = b_h + b_\ell \Rightarrow t = b_\ell$

So we have that $s + t = a + b_h + b_\ell \Rightarrow s + t$ is exactly equal to $a + b$.

The second case when b_ℓ is not ignored so we have $\circ(b) = b_h + ulp(a)$. $\Rightarrow$:

- $s = \circ(a + b) \Rightarrow s = a + b_h + ulp(a)$
- $z = \circ(s - a) \Rightarrow z = a + b_h + ulp(a) - a \Rightarrow z = b_h + ulp(a)$
- $t = \circ(b - z) \Rightarrow t = b - (b_h + ulp(a))$ we have that $b = b_h + b_\ell \Rightarrow t = b_\ell - ulp(a)$

So we have that $s + t = a + b_h + ulp(a) + b_\ell - ulp(a) \Rightarrow s + t$ is exactly equal to $a + b$.

We suppose $a > 0$ and $b > 0$ (respectively $a < 0$ and $b < 0$). We suppose that $|b| < ulp(a)$. Either we have $0 < |b| < 2^{-53}.|a|$ or $2^{-53}|a| \leq |b| \leq 2^{-52}.|a|$.
If $0 < |b| < 2^{-53}.|a|$ for the modes of Rounding $RNDZ$, $RNDN$ and $RNDD$ (respectively $RNDZ$, $RNDN$ and $RNDU$) then b will be ignored otherwise $RNDU$(respectively $RNDD$) b is not ignore, so we have $\circ(b) = ulp(a)$;
If $2^{-53}|a| \leq |b| \leq 2^{-52}.|a|$ for the modes of Rounding $RNDZ$ and $RNDD$ (respectively $RNDZ$ and $RNDU$) then b_ℓ will be also ignored for $RNDN$(closest round) and $RNDU$(respectively $RNDN$ and $RNDD$).

The first case when b is ignored so we have :

- $s = \circ(a + b) \Rightarrow s = a$
- $z = \circ(s - a) \Rightarrow z = a - a \Rightarrow z = 0$
- $t = \circ(b - z) \Rightarrow t = b - 0 \Rightarrow t = b$

So we have that $s + t = a + b \Rightarrow s + t$ is exactly equal to $a + b$.

The second case when b is not ignored so we have $\circ(b) = ulp(a)$.

- $s = \circ(a + b) \Rightarrow s = a + ulp(a)$
- $z = \circ(s - a) \Rightarrow z = a + ulp(a) - a \Rightarrow z = ulp(a)$
- $t = \circ(b - z) \Rightarrow t = b - ulp(a)$

So we have that $s+t = a+ulp(a)+b-ulp(a) \Rightarrow s+t$ is exactly equal to $a+b$.

We suppose that $a > 0$ and $b < 0$ (respectively $a < 0$ and $b > 0$):
If $|b| \geq |\frac{a}{2}|$.
So we have:

- $s = \circ(a + b)$, we have $\frac{-b}{2} \leq a \leq -2b$ after the **Sterbenz**'s lemma (lemma 1) , we have s is exactly equal to $a + b$
- $z = \circ(s - a) \Rightarrow z = \circ((a + b) - a) \Rightarrow z = \circ(b) = b \Rightarrow z = b$
- $t = \circ(b - z) \Rightarrow t = \circ(b - b) \Rightarrow t = 0$

So we have, $s + t$ is exactly equal to $a + b$.

If $|b| < |\frac{a}{2}|$:

- $s = \circ(a + b) \Rightarrow \frac{a}{2} < s \leq \ a \Rightarrow \frac{a}{2} \leq s \leq \ 2a$
- $z = \circ(s - a) \Rightarrow$ and $\frac{a}{2} \leq s \leq \ 2a$ after the **Sterbenz**'s lemma (lemma 1) , we have z is exactly equal to $s - a$.
- $t = \circ(b - z) \Rightarrow t = \circ(b - (s - a) \Rightarrow t = \circ(a + b - s) \Rightarrow t = \circ(a + (b - s))$
  or $-\frac{a}{2} < b < 0$
  and $\frac{a}{2} \leq s \leq \ 2a \Rightarrow \frac{a}{2} - 0 < s - b < 2a - (-\frac{a}{2}) \Rightarrow \frac{a}{2} < s - b < \frac{3a}{2} \Rightarrow \frac{a}{2} \leq s - b \leq 2a$ after the **Sterbenz**'s lemma (lemma 1), we have t is exactly equal to $a + b - s$.

We have $s + t = a + b - s + s \Rightarrow s + t = a + b$.
(By symetric, we have the same results for $a < 0$ and $b > 0$.)
The 4 modes of rounding for $Add112$ have an exact equality between $s + t$ and $a + b$.

(2):
According to Proof(1) , We have 6 possibilities for the result of s and t:

- If $s = a + b$ and $t = 0$ so $|t| \leq 2^{-53}|s|$ for $RNDN$ and $|t| \leq 2^{-52}|s|$ for the direct rounding modes.

- If $s = a$ and $t = b$ (a and b have the same sign) with $|b| < 2^{-53}.|a| \Rightarrow |t| \leq 2^{-53}|s|$ for $RNDN$ a fortiori $|t| \leq 2^{-52}|s|$ for the direct rounding modes.

- If $s = a + b_h$ and $t = b_l$ (a, b_h and b_ℓ are the same sign). We suppose that $a > 0$, $b_h > 0$ and $b_\ell > 0$. We know that $0 \leq b_\ell < 2^{-53}.a \Rightarrow a > 2^{53}.b_\ell$ and that b_h is a multiple of $ulp(a) \Rightarrow b_h > 2.b_\ell$. So we have $a + b_h > 2^{53}b_\ell + 2b_\ell \Rightarrow a + b_h > (2^{53} + 2).b_\ell$ or $2^{53} + 2 > 2^{53}$ $\Rightarrow a + b_h > 2^{53}b_\ell \Rightarrow s > 2^{53}.t \Rightarrow t \leq 2^{-53}.s \Rightarrow |t| \leq 2^{-53}.|s|$ (and similarly if $a < 0$, $b_h < 0$ and $b_\ell < 0$).

- If $s = a + ulp(a)$ and $t = b - ulp(a)$:
  We know according to proof(1) that $|b| \leq 2^{-53}|a|$.
  We suppose that $a > 0$ and $b > 0$, so we have:
  $t \leq t + ulp(a) \Rightarrow t \leq b - ulp(a) + ulp(a) \Rightarrow t \leq b \Rightarrow |t| \leq 2^{-53}|a|$
  We search the upper bound of $|a|$ as a function of $|s|$:
  $s = a + ulp(a) \Rightarrow s \geq a \Rightarrow |s| \geq |a|$ so:
  $|t| \leq 2^{-53}|s|$
  (and similarly if $a < 0$ and $b < 0$).

- If $s = a + b_h + ulp(a)$ and $t = b_\ell - ulp(a)$
  We know according to proof(1) that $|b_\ell| \leq 2^{-53}|b_h|$.
  We suppose that $a > 0$ and $b_h > 0$, so we have:
  $t \leq t + ulp(a) \Rightarrow t \leq b_\ell - ulp(a) + ulp(a) \Rightarrow t \leq b_\ell \Rightarrow |t| \leq 2^{-53}|b_h|$
  $\Rightarrow |t| \leq 2^{-53}|a + b_h|$
  We search the upper bound of $|a + b_h|$ as a function of $|s|$:
  $s = a + b_h + ulp(a) \Rightarrow s \geq a + b_h \Rightarrow |s| \geq |a + b_h|$ so:
  $|t| \leq 2^{-53}|s|$
  (and similarly if $a < 0$ and $b < 0$).

- If $t = a + b - s$, so $|t| = |a + b - s| = |s - (a + b)|$ according to collary 6 $\Rightarrow$:

  $$|s - (a + b)| \leq u.|s|$$

  with u is the **unit Roundoff**(see 1.2.2).

For all cases, we have $|t| \leq u.|s|$ so for $RNDN$: $|t| \leq 2^{-53}.|s|$ and for the others rounding modes: $|t| \leq 2^{-52}.|s|$.

3.1.2 Add122

See algorithm 2

Lemma 2 (Add122) *Let a a* ***Double*** *number and* (b_h, b_ℓ) *a* ***Double-Double*** *number, with* $|a| \geq |b_h|$*, s and t result of* $Add122(a, b_h, b_\ell)$ *for the 4 modes of rounding, considering that there is no* ***overflow*** *so:*

- (1) $s + t = a + b_h + b_\ell + \theta$ *with* $|\theta| \leq u.|t|$
- (2) $|t| \leq 2.u.|s|$

$2.u = 2^{-52}$ for $RNDN$ and $2.u = 2^{-51}$ for the other rounding modes.

PROOF (1) We suppose $|a| \geq |b_h|$ and there is no **overflow**.
We have:

- $s + \ell = Add112(a, b_h)$ according to Lemma 1 $\Rightarrow s + \ell$ is exactly equal to $a + b_h$.
- $t = \circ(\ell + b_\ell) \Rightarrow t = \ell + b_\ell + \theta$ according to the collary 6 we have $|\theta| \leq u.|t|$

So, we have $s + t = a + b_h - \ell + l + b_\ell + \theta_1 \Rightarrow s + t = a + b_h + b_\ell + \theta_1 \Rightarrow s + t = a + b + \theta$ with $|\theta| \leq u.|t|$.

(2) We suppose $|a| \geq |b_h|$ and there is no **overflow**.
We have:

- $t = \circ(\ell + b_\ell) \Rightarrow t = \ell + b_\ell + \epsilon$ with $|\epsilon| \leq u.|t|$

- $s+\ell = Add112(a, b_h)$ according to Lemma 1 $\Rightarrow$ $s+\ell = a+b_h$ and $|\ell| \leq u.|s| \Rightarrow$

$$|s+\ell| \geq |s| - |\ell|$$
$$|s+\ell| \geq |s| - u.|s|$$
$$|s+\ell| \geq (1-u).|s|$$

$\Rightarrow$

$$(1-u).|s| \leq |a+b_h|$$
$$|s| \leq \frac{1}{1-u}.|a+b_h|$$

As $|\ell| \leq u.|s| \Rightarrow$

$$|\ell| \leq \frac{u}{1-u}.|a+b_h|$$

As $|b_\ell| \leq u.|b_h|$ and $|a| \geq |b_h| \Rightarrow |b_\ell| \leq u.|a|$ and that $|a| \leq 2.|a+b_h| \Rightarrow$

$$|b_\ell| \leq 2.u.|a+b_h|$$
$$|t| \leq \frac{u}{1-u}.|a+b_h| + 2.u.|a+b_h| + u.|t|$$
$$(1-u).|t| \leq (\frac{u}{1-u} + 2.u).|a+b_h|$$
$$|t| \leq \frac{1}{1-u}.(\frac{u}{1-u} + 2.u).|a+b_h|$$

We seek the upper bound of $|a+b_h|$ in function of $|s|$:

$$|a+b_h| \leq (1+u).|s|$$

$\Rightarrow$

$$|t| \leq \frac{1}{1-u}.(\frac{u}{1-u} + 2.u).(1+u).|s|$$
$$|t| \leq \frac{(-2.u^2 + 2.u).(u+1)}{(1-u)^2}.|s|$$

As $\frac{(-2.u^2+2.u).(u+1)}{(1-u)^2} \leq 2.u$ We have:

$$|t| \leq 2.u.|s|$$

Theorem 1 (Relative error algorithm *Add*122 without occurring of cancellation) *Let a a **double** number and (b_h, b_ℓ) a **double-double** number are the arguments of the function Add122:*
If a and b_h are the same sign, so:

$$s + t = (a + b_h + b_\ell)(1 + \epsilon)$$

with $|\epsilon| \leq 2.u^2$.

$2.u^2 = 2^{-105}$ for $RNDN$ and $2.u^2 = 2^{-103}$ for the other rounding modes.

Proof According to the calculation technique of the theorem 4.2 proof's([12]).
We have $|a| \geq |b_h|$, either $a > 0$ and $b_h > 0$ or $a < 0$ and $b_h < 0$. As they're symmetric, we only use $a > 0$ and $b_h > 0$.
Based on the algorithm ($Add122$), we have:
$t = \circ(l + b_\ell)$ according to collary 5 $\Rightarrow t = (l + b_\ell).(1 + \epsilon)$ with $|\epsilon| \leq u$.

$$t = \ell + b_\ell + \delta$$

with $\delta = (\ell + b_\ell).\epsilon$ We calculate $|\delta|$, so we have:

$$|\delta| = |(\ell + b_\ell).\epsilon$$

according to the triangle inequality, we have:

$$|\delta| \leq |\ell.\epsilon| + |b_\ell.\epsilon|$$

$$|\delta| \leq |\epsilon|.(|\ell| + |b_\ell|)$$

Based on Lemma 1, we have $|l| \leq u.|s| \Rightarrow |s + \ell| \geq (1 - u).|s| \Rightarrow (1 - u).|s| \leq |a + b_h| \Rightarrow |s| \leq \frac{1}{1-u}.|a + b_h| \Rightarrow |\ell| \leq \frac{u}{1-u}.|a + b_h|$

So we have:

$$|\delta| \leq |\epsilon|.(\frac{u}{1 - u}.|a + b_h| + |b_\ell|)$$

we have $|b_\ell| \leq u.|b_h| \leq u.|a| \leq u.|a + b_h| \Rightarrow$

$$|\delta| \leq |\epsilon|.(\frac{u}{1 - u}.|a + b_h| + u.|a + b_h|)$$

$$|\delta| \leq |\epsilon|.\frac{-u^2 + 2.u}{1 - u}.|a + b_h|$$

As $\frac{-u^2+2.u}{1-u} \leq 2.u$ and $|\epsilon| \leq u \Rightarrow$

$$|\delta| \leq 2.u^2.|a + b_h|$$

$$|b_\ell| \leq u.|a|$$

$$|b_\ell| \leq u.|a + b_h|$$

Now we search a lower bound for $|a + b_h + b_\ell|$ in function $|a + b_h|$

$$|a + b_h + b_\ell| \leq |a + b_h| + |b_\ell|$$

So we have:

$$|a + b_h + b_\ell| \geq (1 - u)|a + b_h|$$

and we knows that $|\delta| \leq |a + b_h| \times 2.u^2 \Rightarrow$

$$|\delta| \leq |a + b_h + b_\ell|(\frac{1}{1-u}).2.u^2$$

After the calculations, we have :

$$|\delta| \leq |a + b_h + b_\ell|.|\epsilon|$$

with $|\epsilon| \leq \frac{2.u^2}{1-u} \leq 2.u^2$

Theorem 2 (Relative error algorithm *Add*122 with a bounded cancellation) *Let a a **Double** number and $(b_h,\ b_\ell)$ a **double-double** number . We have for the algorithm Add122 with a and $(b_h,\ b_\ell)$ it's arguments.*
If a and b_h are different sign and we suppose $|b_h| \leq 2^{-\mu}|a|$ with $\mu \geq 1$.
So :

$$s + t = (a + (b_h + b_\ell))(1 + \epsilon)$$

with

$$|\epsilon| \leq 2.u^2.\frac{2^{-53-\mu}}{1 - 2^{-\mu} - u} \leq 2.(2.u^2) = 4.u^2$$

Proof According to the calculation technique of the theorem 4.3 proof's([12]). We suppose $|b_h| \leq 2^{-\mu}|a|$ with $\mu \geq 1$. First we look for the upper bound, using the results of the proof of Theorem 1. We have:

$$|b_\ell| \leq u.|b_h| \leq u.2^{-\mu}|a|$$

then we search the lower bound for $|a + b_h|$ in function of $|a|$:

$$|a + b_h| \geq (1 - 2^{-\mu}).|a|$$

So we have:

$$|b_\ell| \leq \frac{u.2^{-\mu}}{1-2^{-\mu}}|a+b_h|$$

So:

$$|\delta| \leq |a_h+b_h|.2.u^2.\frac{u.2^{-\mu}}{1-2^{-\mu}}$$

Now, we search the lower bound for $|a+b_h+b_\ell|$ depending on $|a+b_h|$.
We have :

$$|b_\ell| \leq \frac{u.2^{-\mu}}{1-2^{-\mu}}|a+b_h|$$

$$|a+b_h+b_\ell| \geq |a+b_h|\frac{1-2^{-\mu}-u}{1-2^{-\mu}}$$

So we have for $|\delta|$:

$$|\delta| \leq |a+b_h+b_\ell|.\frac{1-2^{-\mu}}{1-2^{-\mu}-u}.2.u^2.\frac{u.2^{-\mu}}{1-2^{-\mu}}$$

$$|\delta| \leq |a+b_h+b_\ell|.2.u^2.\frac{u.2^{-\mu}}{1-2^{-\mu}-u}$$

So:

$$|\epsilon| \leq 2.u^2.\frac{2^{-53-\mu}}{1-2^{-\mu}-u}$$

As $\mu \geq 1$, we want the upper bound of $\frac{u.2^{-\mu}}{1-2^{-\mu}-u} \Rightarrow$

$$\frac{2^{-54}}{1/2-2^{-52}} \leq 2$$

because $2^{-53} \leq u \leq 2^{-52}$ So :

$$|\epsilon| \leq 2.2.u^2 = 4.u^2$$

3.1.3 Add222

See algorithm 3

Lemma 3 (Add222) *Let* (a_h, a_ℓ) *and* (b_h, b_ℓ) ***Double-Double*** *numbers, with* $|a_h| \geq |b_h|$, s *and* t *result of* $Add222(a_h, a_\ell, b_h, b_\ell)$ *for the 4 modes of rounding, considering that there is no* ***overflow*** *so:*

$$|t| \leq 6.u.|s|$$

$6.u = 2^{-50.4}$ for $RNDN$ and $6.u = 2^{-49.4}$ for the other rounding modes.

PROOF According to *Add*222 $\Rightarrow$: $t = \circ(m + b_\ell)$ (corollary 5) $\Rightarrow t \leq (m + b_\ell)(1 + \epsilon_1)$ with $|\epsilon_1| \leq u$.

$$|t| \leq (|m| + |b_\ell|).|1 + \epsilon_1|$$

We have : $m = \circ(\ell + a_\ell) \Rightarrow m = (\ell + a_\ell).(1 + \epsilon_2)$ with $|\epsilon_2| \leq u \Rightarrow$.

$$|t| \leq (|(\ell + a_\ell)(1 + \epsilon_2)| + |b_\ell|).|1 + \epsilon_1|$$

After calculate, we have:

$$|t| \leq |\ell|.|1 + \epsilon_1 + \epsilon_2 + \epsilon_1.\epsilon_2| + |a_\ell|.|1 + \epsilon_1 + \epsilon_2 + \epsilon_1.\epsilon_2| + |b_\ell|.|1 + \epsilon_1|$$

As $|\epsilon_1| \leq u$ and $|\epsilon_2| \leq u \Rightarrow$

$$|t| \leq |\ell|.(1 + 2.u + u^2) + |a_\ell|.(1 + 2.u + u^2) + |b_\ell|.(1 + u)$$

As :

$$|a_\ell| \leq u.|a_h| \leq 2.u.|a_h + b_h|$$
$$|b_\ell| \leq u.|b_h| \leq u.|a_h| \leq 2.u.|a_h + b_h|$$

and that $a_h + b_h = s + \ell \Rightarrow$

$$|t| \leq |\ell|.(1 + 2.u + u^2) + 2.u.(1 + 2.u + u^2).|s + \ell| + 2.u.(1 + u).|s + \ell|$$

We know that $|\ell| \leq u.|s|$, after calculate, we have:

$$|t| \leq (u^4 + 5.u^3 + 7.u^2 + 3.u).|s|$$

As $(2.u^4 + 9.u^3 + 12.u^2 + 5.u) \leq 6.u$

$$|t| \leq 6.u.|s|$$

Theorem 3 (Relative error algorithm *Add*222 without occurring of cancellation) *Let (a_h, a_ℓ) and (b_h, b_ℓ) the **double-double** . We have for the algorithm Add222 with (a_h, a_ℓ) and (b_h, b_ℓ) it's arguments.*
If a_h and b_h have the same sign, so:

$$s + t = ((a_h + a_\ell) + (b_h + b_\ell))(1 + \epsilon)$$

with $|\epsilon| \leq 3.u^2$

$3.u^2 = 2^{-104.4}$ for $RNDN$ and $3.u^2 = 2^{-102.4}$ for the other rounding modes.

PROOF According to the calculation technique of the theorem 4.2 proof's([12]). We have $|a_h| \geq |b_h|$, either $a_h > 0$ and $b_h > 0$ or $a_h < 0$ and $b_h < 0$. As they're symmetric, we only use $a_h > 0$ and $b_h > 0$.
Based on the algorithm $(Add222)$, we have:
$t = \circ(m + b_\ell)$ according to corollary 5 $\Rightarrow t = (m + b_\ell)(1 + \epsilon_1)$ with $|\epsilon_1| \leq u$.

$$t = (\circ(\ell + a_\ell) + b_\ell)(1 + \epsilon_1)$$

$$t = ((\ell + a_\ell)(1 + \epsilon_2) + b_\ell)(1 + \epsilon_1)$$

with $|\epsilon_2| \leq u$.

$$t = \ell + a_\ell + b_\ell + \delta$$

with $\delta = (\ell + a_\ell + b_\ell)\epsilon_1 + (\ell + a_\ell)\epsilon_1\epsilon_2$.
We calculate $|\delta|$, so we have:

$$|\delta| = |(\ell + a_\ell + b_\ell)\epsilon_1 + (\ell + a_\ell)\epsilon_1\epsilon_2|$$

according to the triangle inequality, we have:

$$|\delta| \leq |(\ell + a_\ell + b_\ell)\epsilon_1| + |(\ell + a_\ell)\epsilon_1\epsilon_2|$$

$$|\delta| \leq |\ell\epsilon_1| + |\ell\epsilon_1\epsilon_2| + |(a_\ell + b_\ell)\epsilon_1| + |a_\ell\epsilon_1\epsilon_2|$$

$$|\delta| \leq |\ell|(|\epsilon_1| + |\epsilon_1\epsilon_2|) + |(a_\ell + b_\ell)\epsilon_1| + |a_\ell\epsilon_1\epsilon_2|$$

$$|\delta| \leq |\ell|.(|\epsilon_1| + |\epsilon_1\epsilon_2|) + |(a_\ell + b_\ell)|.|\epsilon_1| + |a_\ell|.|\epsilon_1\epsilon_2|$$

$$|\delta| \leq |\ell|.(|\epsilon_1| + |\epsilon_1\epsilon_2|) + |(a_\ell + b_\ell)|.|\epsilon_1| + |(a_\ell + b_\ell)|.|\epsilon_1\epsilon_2|$$

$$|\delta| \leq (|\ell| + |(a_\ell + b_\ell)|).|\epsilon_1 + \epsilon_1\epsilon_2|$$

Based on Lemma 1, we have $|l| \leq u.|s|$ and as $|s| \leq |a_h + b_h| \Rightarrow |l| \leq u.|a_h + b_h|$.

$$|\delta| \leq (u.|a_h + b_h| + |a_\ell| + |b_\ell|).|\epsilon_1 + \epsilon_1\epsilon_2|$$

we have $|a_\ell| \leq u.|a_h| \leq u.|a_h + b_h|$ and $|b_\ell \leq u.|b_h| \leq u.|a_h + b_h| \Rightarrow$

$$|\delta| \leq (u.|a_h + b_h| + u.|a_h + b_h| + u.|a_h + b_h|).|\epsilon_1 + \epsilon_1\epsilon_2|$$

$$|\delta| \leq |a_h + b_h| \times 3.u.|\epsilon_1 + \epsilon_1\epsilon_2|$$

$$|\delta| \leq |a_h + b_h| \times 3.u.(u + u^2)$$

$$|\delta| \leq |a_h + b_h| \times 3.(u^2 + u^3)$$

We seek the upper bound of $|a_h + b_h|$ in function of $|a_h + b_h + a_\ell + b_\ell|$:

$$|a_\ell + b_\ell| \leq |a_\ell| + |b_\ell|$$
$$|a_\ell + b_\ell| \leq u.|a_h| + u.|b_h|$$
$$|a_\ell + b_\ell| \leq 2.u.|a_h|$$
$$|a_\ell + b_\ell| \leq 2.u.|a_h + b_h|$$

$\Rightarrow$

$$|a_h + b_h + a_\ell + b_\ell| \geq (1 - 2.u)|a_h + b_h|$$

and we know that $|\delta| \leq |a_h + b_h| \times 3.(u^2 + u^3) \Rightarrow$

$$|\delta| \leq |a_h + a_l + b_h + b_\ell|(\frac{1}{1 - 2.u})3.(u^2 + u^3)$$

$$|\delta| \leq |a_h + a_l + b_h + b_\ell|(\frac{3.u^2 + 3.u^3}{1 - 2.u})$$

but $\frac{3.u^2+3.u^3}{1-2.u} \leq 3.u^2$ we have :

$$|\epsilon| \leq 3.u^2$$

Theorem 4 (Relative error algorithm *Add*222 with a bounded cancellation) *Let $(a_h,\, a_\ell)$ and $(b_h,\, b_\ell)$ are the **double-double** number . We have for the algorithm Add222 with $(a_h,\, a_\ell)$ and $(b_h,\, b_\ell)$ it's arguments.*
If a_h and b_h are different sign and we suppose $|b_h| \leq 2^{-\mu}|a_h|$ with $\mu \geq 1$.
So :

$$s + t = ((a_h + a_\ell) + (b_h + b_\ell))(1 + \epsilon)$$

with

$$|\epsilon| \leq 3.u^2.\frac{1 - 2^{-\mu-1}}{1 - 2^{-\mu} - 2.u} \leq 2.3.u^2 = 6.u^2$$

Proof According to the calculation technique of the theorem 4.3 proof's([12]). We suppose $|b_h| \leq 2^{-\mu}|a_h|$ with $\mu \geq 1$. First we look for the upper bound, using the results of the proof of Theorem 4.1.1. We have:

$$|b_\ell| \leq u.|b_h| \leq u.2^{-\mu}|a_h|$$

then we search the lower bound:

$$|a_h + b_h| \geq (1 - 2^{-\mu}).|a_h|$$

As $|a_\ell| \leq u.|a_h|$ and $|b_\ell| \leq u.|b_h| \leq u.2^{-\mu}.|a_h|$.

$$|a_\ell| \leq \frac{u}{1-2^{-\mu}}|a_h + b_h|$$

and

$$|b_\ell| \leq \frac{u.2^{-\mu}}{1-2^{-\mu}}|a_h + b_h|$$

$\Rightarrow$

$$u + u.2^{-\mu} \leq 1 - 2^{-\mu-1}$$

because of $\mu \geq 1$
So:

$$|\delta| \leq |a_h + b_h|.3.u^2.\frac{1-2^{-\mu-1}}{1-2^{-\mu}}$$

Now, we search the lower bound for $|a_h + a_\ell + b_h + b_\ell|$ depending on $|a_h + b_h|$.
We have:

$$|a_\ell + b_\ell| \leq |a_\ell| + |b_\ell|$$

$$|a_\ell + b_\ell| \leq 2.u.|a_h|$$

$$|a_\ell + b_\ell| \leq \frac{2.u}{1-2^{-\mu}}|a_h + b_h|$$

$$|a_h + a_\ell + b_h + b_\ell| \geq |a_h + b_h|\frac{1-2^{-\mu}-2.u}{1-2^{-\mu}}$$

So we have for $|\delta|$:

$$|\delta| \leq |a_h + a_\ell + b_h + b_\ell|.\frac{1-2^{-\mu}}{1-2^{-\mu}-2.u}.3.u^2.\frac{1-2^{-\mu-1}}{1-2^{-\mu}}$$

$$|\delta| \leq |a_h + a_\ell + b_h + b_\ell|.3.u^2.\frac{1-2^{-\mu-1}}{1-2^{-\mu}-2.u}$$

So:

$$|\epsilon| \leq 3.u^2.\frac{1-2^{-\mu-1}}{1-2^{-\mu}-2.u}$$

As $\mu \geq 1$, we want the upper bound of $\frac{1-2^{-\mu-1}}{1-2^{-\mu}-2.u} \Rightarrow$

$$\frac{3/4}{1/2-2.u} \leq 2$$

So :

$$|\epsilon| \leq 2.3.u^2 = 6.u^2$$

3.2 Multiplication Operators

3.2.1 Mul112

See algorithm 4

Lemma 4 (Mul112) *Let a and b floating point numbers, s and t result of $Mul112(a, b)$ for the 4 modes of rounding, considering that there is no **overflow** so :*

- *(1) $r_1 + r_2$ is exactly equal to $a.b$.*
- *(2) $|r_2| \leq u.|r_1|$*

PROOF (1) According to [19], $FMA(a, b, -c)$ is exacty equal to $a + b - c$ for the 4 modes of rounding .
Based on Algorithm $Mul112$, we have :
$r_2 = FMA(a, b, -r1)$ that supposed $\Rightarrow r_2 = a * b - r_1$
So

$$r_1 + r_2 = a * b - r_1 + r_1$$

$$r_1 + r_2 = a * b$$

(2) $r1 = \circ(a * b)$ according to the collary 6 $\Rightarrow r1 = a * b + \epsilon$ with $|\epsilon| \leq u.|r_1|$ We know that $r2$ is exactly equal to $a * b - r_1$.
$\Rightarrow r_2 = a * b - r1 \Rightarrow$

$$r_2 = a * b - (a * b + \epsilon)$$

$$r_2 = -\epsilon$$

That $|\epsilon| \leq u.|r_1| \Rightarrow |r_2| \leq u.|r_1|$.

3.2.2 Mul122

See algorithm 5

Lemma 5 (Mul122) *Let a a **Double** number and (b_h, b_ℓ) a **Double-Double** number, r_1 and r_2 result of $Mul122(a, b_h, b_\ell)$ for the 4 modes of rounding, considering that there is no **overflow** so $|r_2| \leq u.|r_1|$.*

PROOF We know $|b_\ell| \leq u.|b_h|$

In the algorithme $Mul122$, we see $r_1, r_2 = Add112(t_1, t_6)$, bases on the lemma 1, we have $|r_2| \leq u.|r_1|$.

Theorem 5 (Relative error algorithm $Mul122$) *Let a a* ***double*** *number and (b_h, b_ℓ) a* ***double-double*** *number are the arguments of the function $Mul122$:*
So:

$$r_1 + r_2 = (a.(b_h + b_\ell)).(1 + \epsilon)$$

with $|\epsilon| \leq 3.u^2$.

$3.u^2 = 2^{-104.41}$ for $RNDN$ and $3.u^2 = 2^{-102.41}$ for the other rounding modes.

Proof According to the calculation technique of the theorem 4.7 proof's([12]).
We have from Algorithm 5:
$r_1, r_2 = Add112(t_1, t_4)$ according to $Add112 \Rightarrow r_1 + r_2 = t_1 + t_4$
$t_4 = \circ(t_2 + t_3) \Rightarrow t_4 = (t_2 + t_3)(1 + \epsilon_1)$ with $|\epsilon_1| \leq u$
$t_3 = \circ(a.b_\ell) \Rightarrow t_3 = (a.b_\ell)(1 + \epsilon_2)$ with $|\epsilon_2| \leq u$
$t_1, t_2 = Mul112(a, b_h)$ according to $Mul112 \Rightarrow |t_2| \leq u.|t_1|$ and $t_1 + t_2 = a \times b_h$
So we have :

$$t_4 = (t_2 + a.b_\ell.(1 + \epsilon_2)).(1 + \epsilon_1)$$

$$t_4 = t_2 + a.b_\ell + a.b_\ell.\epsilon_2 + t_2.\epsilon_1 + a.b_\ell.\epsilon_1 + a.b_\ell.\epsilon_2.\epsilon_1$$

as $r_1 + r_2 = t_1 + t_4 \Rightarrow$

$$r_1 + r_2 = t_1 + (t_2 + a.b_\ell + a.b_\ell.\epsilon_2 + t_2.\epsilon_1 + a.b_\ell.\epsilon_1 + a.b_\ell.\epsilon_2.\epsilon_1)$$

but $t_1 + t_2 = a.b_h \Rightarrow$

$$r_1 + r_2 = a.b_h + a.b_\ell + a.b_\ell.\epsilon_2 + t_2.\epsilon_1 + a.b_\ell.\epsilon_1 + a.b_\ell.\epsilon_2.\epsilon_1$$

$$r_1 + r_2 = a.b_h + a.b_\ell + \delta$$

with

$$\delta = t_2.\epsilon_1 + a.b_\ell.\epsilon_2 + a.b_\ell.\epsilon_1 + a.b_\ell.\epsilon_2.\epsilon_1$$

So we have:

$$|\delta| \leq |t_2.\epsilon_1| + |a.b_l.(\epsilon_1 + \epsilon_2 + \epsilon_2.\epsilon_1)|$$

as we know that $|\epsilon_1| \leq u$ and $|\epsilon_2| \leq u \Rightarrow$

$$|\delta| \leq |t_2|.u + |a.b_\ell|.(u^2 + 2.u)$$

According to the conditions of the algorithm :
$|b_\ell| \leq u.|b_h| \Rightarrow |a \times b_\ell| \leq u.|a.b_h|$. So:

$$|\delta| \leq |t_2|.u + u.|a.b_h|.(u^2 + 2.u)$$

$$|\delta| \leq |t_2|.u + |a.b_h|.(u^3 + 2.u^2)$$

As $a.b_h = t_1 + t_2$ and that $|t_2| \leq u.|t_1|$, we have $|a.b_h| \geq |(1 + \frac{1}{u}).t_2|$ but $1 + \frac{1}{u} > 0 \Rightarrow$

$$(1 + \frac{1}{u}).|t_1| \leq |a.b_h|$$

$$\frac{1+u}{u}.|t_2| \leq |a.b_h|$$

$$|t_2| \leq \frac{u}{1+u}.|a.b_h|$$

So :

$$|\delta| \leq \frac{u^2}{1+u}.|a.b_h| + |a.b_h|.(u^3 + 2.u^2)$$

After calculate, $\Rightarrow$

$$|\delta| \leq \frac{u^4 + 3.u^3 + 3.u^2}{1+u}.|a.b_h|$$

Now , we search a lower bound for $|a_h.(b_h + b_\ell)|$ in fuction of $|a_h.b_h|$.
For to calculate this lower bound, we need to search the upper bound of $|a.b_\ell|$. So we have:

$$|a.b_\ell| \leq u.|a.b_h|$$

So we have:

$$|a.(b_h + b_\ell)| \geq (1 - u).|a_h.b_h|$$

$$\frac{1}{1-u}.|a.(b_h + b_\ell)| \geq |a_h.b_h|$$

For the calculation of relative error, we have:

$$|\delta| \leq \frac{u^4 + 3.u^3 + 3.u^2}{1-u^2}.|a.b_h|$$

As $\frac{u^4+3.u^3+3.u^2}{1-u^2} \leq 3.u^2 \Rightarrow |\delta| \leq 3.u^2.|a.(b_h + b_\ell)|$.
For $r_1 + r_2 = (a.(b_h + b_\ell)).(1 + \epsilon)$ We have that $|\epsilon| \leq 3.u^2$

3.2.3 Mul222

See algorithm 6

Lemma 6 (Mul222) *Let* (a_h, a_ℓ) *and* (b_h, b_ℓ) *are* ***Double-Double*** *numbers,* r_1 *and* r_2 *result of* $Mul222(a_h, a_\ell, b_h, b_\ell)$ *for the* 4 *modes of rounding, considering that there is no* ***overflow*** *so* $|r_1| \leq u.|r_2|$.

PROOF We know $|b_\ell| \le u.|b_h|$
In the algorithme $Mul222$, we see $r_1, r_2 = Add112(t_1, t_6)$, based on the lemma 1, we have $|r_2| \le u.|r_1|$.

Theorem 6 (Relative error algorithm *Mul222*) *Let (a_h, a_ℓ) and (b_h, b_ℓ) are **double-double** numbers and are the arguments of the function $Mul222$: so:*

$$r_1 + r_2 = ((a_h + a_\ell) \times (b_h + b_\ell))(1 + \epsilon)$$

with $|\epsilon| \le 7.u^2$.

$7.u^2 = 2^{-103.19}$ for $RNDN$ and $7.u^2 = 2^{-101.19}$ for the other rounding modes.

PROOF According to the calculation technique of the theorem 4.7 proof's([12]).
We have from Algorithm 7:
$r_1, r_2 = Add112(t_1, t_6)$ according to $Add112 \Rightarrow r_1 + r_2 = t_1 + t_6$
$t_6 = \circ(t_2 + t_5) \Rightarrow t_6 = (t_2 + t_5)(1 + \epsilon_1)$ with $|\epsilon_1| \le u$
$t_5 = \circ(t_3 + t_4) \Rightarrow t_5 = (t_3 + t_4)(1 + \epsilon_2)$ with $|\epsilon_2| \le u$
$t_4 = \circ(b_h \times a_\ell) \Rightarrow t_4 = (b_h.a_\ell).(1 + \epsilon_3)$ with $|\epsilon_3| \le u$
$t_3 = \circ(a_h \times b_\ell) \Rightarrow t_3 = (a_h.b_l).(1 + \epsilon_4)$ with $|\epsilon_4| \le u$
$t_1, t_2 = Mul112(a_h, b_h)$ according to $Mul112 \Rightarrow |t_2| \le u.|t_1|$ and $t_1 + t_2 = a_h.b_h$
So we have :

$$t_6 = (t_2 + ((a_h.b_\ell).(1 + \epsilon_4) + (b_h.a_\ell).(1 + \epsilon_3)).(1 + \epsilon_2)).(1 + \epsilon_1)$$

As $r_1 + r_2 = t_1 + t_6 \Rightarrow$

$$r_1 + r_2 = t_1 + (t_2 + ((a_h.b_\ell).(1 + \epsilon_4) + (b_h.a_\ell).(1 + \epsilon_3)).(1 + \epsilon_2)).(1 + \epsilon_1)$$

$$r_1 + r_2 = t_1 + t_2 + a_h.b_\ell + b_h.a_\ell + \delta$$

with

$$\delta = t_2.\epsilon_1 + a_h.b_l.(\epsilon_4 + \epsilon_2 + \epsilon_4.\epsilon_2 + \epsilon_1 + \epsilon_4.\epsilon_1 + \epsilon_2.\epsilon_1 + \epsilon_4.\epsilon_2.\epsilon_1) + a_l.b_h.(\epsilon_3 + \epsilon_2 + \epsilon_3.\epsilon_2 + \epsilon_1 + \epsilon_3.\epsilon_1 + \epsilon_2.\epsilon_1 + \epsilon_3.\epsilon_2.\epsilon_1)$$

So we have:

$$|\delta| \le |t_2.\epsilon_1| + |a_h.b_l.(\epsilon_4 + \epsilon_2 + \epsilon_4.\epsilon_2 + \epsilon_1 + \epsilon_4.\epsilon_1 + \epsilon_2.\epsilon_1 + \epsilon_4.\epsilon_2.\epsilon_1)| + |a_l.b_h.(\epsilon_3 + \epsilon_2 + \epsilon_3.\epsilon_2 + \epsilon_1 + \epsilon_3.\epsilon_1 + \epsilon_2.\epsilon_1 + \epsilon_3.\epsilon_2.$$

$$|\delta| \le |t_2.\epsilon_1| + |a_h.b_l|.|\epsilon_4 + \epsilon_2 + \epsilon_4.\epsilon_2 + \epsilon_1 + \epsilon_4.\epsilon_1 + \epsilon_2.\epsilon_1 + \epsilon_4.\epsilon_2.\epsilon_1| + |a_l.b_h|.|\epsilon_3 + \epsilon_2 + \epsilon_3.\epsilon_2 + \epsilon_1 + \epsilon_3.\epsilon_1 + \epsilon_2.\epsilon_1 + \epsilon_3.\epsilon_2.$$

as we know that $|\epsilon_i| \leq u$ with $1 \leq i \leq 4 \Rightarrow$

$|\delta| \leq |t_2.\epsilon_1| + (|a_h.b_l| + |a_l.b_h|).|\epsilon_4 + \epsilon_2 + \epsilon_4.\epsilon_2 + \epsilon_1 + \epsilon_4.\epsilon_1 + \epsilon_2.\epsilon_1 + \epsilon_4.\epsilon_2.\epsilon_1|$

According to the conditions of the algorithm:
$|a_\ell| \leq u.|a_h|$ and $|b_\ell| \leq u.|b_h| \Rightarrow |a_\ell \times b_h| \leq u.|a_h \times b_h|$ and $|a_h \times b_\ell| \leq u.|a_h \times b_h|$. So:

$|\delta| \leq |t_2.\epsilon_1| + (u.|a_h.b_h| + u|a_h.b_h|).|\epsilon_4 + \epsilon_2 + \epsilon_4.\epsilon_2 + \epsilon_1 + \epsilon_4.\epsilon_1 + \epsilon_2.\epsilon_1 + \epsilon_4.\epsilon_2.\epsilon_1|$

We search the upper bound of $|t_2|$ in function of $|a_h.b_h|$, we begin with the upper bound of $|t_1|$, as $t_1 + t_2 = a_h.b_h$ and $|t_2| \leq u.|t_1|$, we have that:

$$|t_1 + t_2| \geq (1 - u).|t_1|$$

$$|t_1| \leq \frac{1}{1-u}.|a_h.b_h|$$

and so:

$$|t_2| \leq \frac{u}{1-u}.|a_h.b_h|$$

$$|\delta| \leq \frac{u}{1-u}.|a_h.b_h|.|\epsilon_1| + 2.u.|a_h.b_h|.|\epsilon_4 + \epsilon_2 + \epsilon_4.\epsilon_2 + \epsilon_1 + \epsilon_4.\epsilon_1 + \epsilon_2.\epsilon_1 + \epsilon_4.\epsilon_2.\epsilon_1|$$

$$|\delta| \leq |a_h.b_h|.(\frac{u}{1-u}.|\epsilon_1| + 2.u.|\epsilon_4 + \epsilon_2 + \epsilon_4.\epsilon_2 + \epsilon_1 + \epsilon_4.\epsilon_1 + \epsilon_2.\epsilon_1 + \epsilon_4.\epsilon_2.\epsilon_1|)$$

After calculation, $\Rightarrow$

$$|\delta| \leq |a_h.b_h|.\frac{-2.u^5 - 4.u^4 + 7.u^2}{1-u}$$

Now , we search a lower bound for $|(a_h + a_\ell).(b_h + b_\ell)|$ in function of $|a_h.b_h|$. To calculate this lower bound, we need to search the upper bound of $|a_h.b_\ell + a_\ell.b_h + a_\ell.b_\ell|$. So we have:

$$|a_h.b_\ell + a_\ell.b_h + a_\ell.b_\ell| \leq |a_h.b_\ell| + |a_\ell.b_h| + |a_\ell.b_\ell|$$

$$|a_h.b_\ell + a_\ell.b_h + a_\ell.b_\ell| \leq u.|a_h.b_h| + u.|a_h.b_h| + u^2.|a_h.b_h|$$

$$|a_h.b_\ell + a_\ell.b_h + a_\ell.b_\ell| \leq (u^2 + 2.u).|a_h.b_h|$$

So we have:

$$|(a_h + a_\ell).(b_h + b_\ell)| \geq (1 - (u^2 + 2.u)).|a_h.b_h|$$

For the calculation of relative error, we have:

$$|\delta| \leq \frac{-2.u^5 - 4.u^4 + 7.u^2}{(1-u).(1-(u^2+2.u))}.|(a_h + a_\ell).(b_h + b_\ell)|$$

but $\frac{-2.u^5 - 4.u^4 + 7.u^2}{(1-u).(1-(u^2+2.u))} \leq 7.u^2 \Rightarrow |\delta| \leq 7.u^2.|(a_h + a_\ell).(b_h + b_\ell)|$.
For $r_1 + r_2 = ((a_h + a_\ell).(b_h + b_\ell)).(1 + \epsilon)$ so $|\epsilon| \leq 7.u^2$

Chapter 4

Operators on Triple-Double numbers

4.1 Addition Operators

4.1.1 Add133

See algorithm 7

Lemma 7 (Add133) *Let a a **double** number and (b_h, b_m, b_ℓ) a **triple-Double** number, r_h, r_m and r_ℓ result of $Add133(a, b_h, b_m, b_\ell)$ for the 4 modes of rounding, considering that there is no **overflow** so $|r_\ell| \le u.|r_m|$, $|r_m| \le u.|r_h|$ and $|r_\ell| \le u^2.|r_h|$*

PROOF We suppose that $|b_m| \le u.|b_h|$, $|b_\ell| \le u.|b_m|$, according to $Add133$, so we have:

- $r_m + r_\ell = Add112(t_2, t_4)$ thanks to $Add112 \Rightarrow r_m + r_\ell = t_2 + t_4$ and $|r_\ell| \le u.|r_m|$.
- $t_4 = \circ(t_3 + b_\ell) \Rightarrow t_4 = (t_3 + b_\ell).(1 + \epsilon)$ with $|\epsilon| \le u$.
- $t_2 + t_3 = Add112(t_1, b_m)$ thanks to $Add112 \Rightarrow t_2 + t_3 = t_1 + b_m$ and $|t_3| \le u.|t_2|$.
- $r_h + t_1 = Add112(a, b_h)$ thanks to $Add112 \Rightarrow r_h + t_1 = a + b_h$ and $|t_1| \le u.|r_h| \Rightarrow$

$$r_m + r_\ell = t_2 + t_4$$

$$r_m + r_\ell = t_2 + (t_3 + b_\ell).(1 + \epsilon)$$

$$r_m + r_\ell = t_2 + t_3.(1+\epsilon) + b_\ell.(1+\epsilon)$$
$$r_m + r_\ell = t_2 + t_3 + t_3.\epsilon + b_\ell.(1+\epsilon)$$

As $t_2 + t_3 = t_1 + b_m \Rightarrow$

$$r_m + r_\ell = t_1 + b_m + t_3.\epsilon + b_\ell.(1+\epsilon)$$
$$|r_m + r_\ell| = |t_1 + b_m + t_3.\epsilon + b_\ell.(1+\epsilon)|$$
$$|r_m + r_\ell| \leq |t_1 + b_m| + |t_3.\epsilon| + |b_\ell.(1+\epsilon)|$$

We search the lower bound of $|t_3|$ in function of $|t_1 + b_m|$:
$|t_1 + b_m| = |t_2 + t_3| \Rightarrow |t_1 + b_m| \geq (1 + \frac{1}{u}).|t_3| \Rightarrow |t_3| \leq \frac{u}{u+1}.|t_1 + b_m|$

$$|r_m + r_\ell| \leq |t_1 + b_m| + \frac{u}{u+1}.|(t_1 + b_m).\epsilon| + |b_\ell.(1+\epsilon)|$$
$$|r_m + r_\ell| \leq |t_1 + b_m| + \frac{u^2}{u+1}.|(t_1 + b_m)| + |b_\ell.(1+u)|$$
$$|r_m + r_\ell| \leq +\frac{u^2+u+1}{u+1}.|(t_1 + b_m)| + |b_\ell.(1+u)|$$

According to the conditions of $Add112$, we have $|t_1| \geq |b_m|$ and also that $|b_\ell| \leq u.|b_m| \Rightarrow$

$$|r_m + r_\ell| \leq \frac{2.(u^2+u+1) + u(u+1)^2}{u+1}.|t_1|$$
$$|r_m + r_\ell| \leq \frac{u^3 + 4.u^2 + 3.u + 1}{u+1}.|t_1|$$

As $|t_1| \leq u.|r_h| \Rightarrow$

$$|r_m + r_\ell| \leq u.\frac{u^3 + 4.u^2 + 3.u + 1}{u+1}.|r_h|$$

We have: $\frac{u^4+4.u^3+3.u^2+u}{u+1} \leq u \Rightarrow$

$$|r_m + r_\ell| \leq u.|r_h|$$

We know that $|r_\ell| \leq u.|r_m| \Rightarrow (1-u).|r_m| \leq |r_m + r_\ell|$.
We have :

$$(1-u).|r_m| \leq u.|r_h|$$
$$|r_m| \leq \frac{u}{1-u}.|r_h|$$

But $\frac{u}{1-u} \leq u \Rightarrow$

$$|r_m| \leq u.|r_h|$$

and

$$|r_\ell| \leq u^2.|r_h|$$

Theorem 7 (Relative error algorithm *Add*133 without occuring of cancellation) *Let a a **double** number and (b_h, b_m, b_ℓ) a **triple-double** number are the arguments of the function $Add133$.*
So:

$$r_h + r_m + r_\ell = (a + (b_h + b_m + b_\ell)).(1 + \epsilon)$$

with $|\epsilon| \le 3.u^3$.

$3.u^3 = 2^{-157.41}$ for $RNDN$ and $3.u^3 = 2^{-154.41}$ for the other rounding modes.

Proof According to the calculation technique of the theorem 4.2 proof's([12]).
We have $|a| \ge |b_h|$, either $a > 0$ and $b_h > 0$ or $a < 0$ and $b_h < 0$. As they're symmetric, we only use $a > 0$ and $b_h > 0$.
We have from Algorithme 7:
$r_m + r_\ell = Add112(t_2, t_4)$ based on $Add112 \Rightarrow r_m + r_\ell = t_2 + t_4$
$t_4 = \circ(t_3 + b_\ell) \Rightarrow t_4 = (t_3 + b_\ell)(1 + \epsilon_1)$ with $|\epsilon_1| \le u$
$t_2 + t_3 = Add112(t_1, b_m)$ based on $Add112 \Rightarrow t_2 + t_3 = t_1 + b_m$
$r_h + t_1 = Add112(a, b_h)$ based on $Add112 \Rightarrow r_h + t_1 = a + b_h$
So we have:

$$r_m + r_\ell = t_2 + (t_3 + b_\ell)(1 + \epsilon_1)$$

$$r_m + r_\ell = t_2 + t_3 + t_3.\epsilon + b_\ell.(1 + \epsilon_1)$$

As $t_2 + t_3 = t_1 + b_m \Rightarrow$

$$r_m + r_\ell = t_1 + b_m + t_3.\epsilon_1 + b_\ell.(1 + \epsilon_1)$$

Then we have:

$$r_h + r_m + r_\ell = r_h + t_1 + b_m + t_3.\epsilon_1 + b_\ell.(1 + \epsilon_1)$$

As $r_h + t_1 = a + b_h$

$$r_h + r_m + r_\ell = a + b_h + b_m + t_3.\epsilon_1 + b_\ell.(1 + \epsilon_1)$$

$$r_h + r_m + r_\ell = a + (b_h + b_m + b_\ell + t_3.\epsilon_1 + b_\ell.\epsilon_1$$

$$r_h + r_m + r_\ell = a + (b_h + b_m + b_\ell) + \delta$$

with $\delta = t_3.\epsilon_1 + b_\ell.\epsilon_1$
So:

$$|\delta| \le |t_3.\epsilon_1| + |b_\ell.\epsilon_1|$$

We know that $t_1 + b_m = t_2 + t_3$ and that $|t_3| \leq u.|t_2| \Rightarrow$

$$|t_1 + b_m| \geq (\frac{1}{u} + 1).|t_3|$$

$$|t_3| \leq \frac{u}{u+1}.|t_1 + b_m|$$

$$|t_3| \leq \frac{u}{u+1}.(|t_1| + |b_m|)$$

As $a + b_h = r_h + t_1$ and that $|t_1| \leq u.|r_h| \Rightarrow$

$$|a + b_h| \geq (\frac{1}{u} + 1).|t_1|$$

$$|t_1| \leq \frac{u}{u+1}.|a + b_h|$$

$\Rightarrow$

$$|\delta| \leq \frac{u}{u+1}.(\frac{u}{u+1}.|a + b_h| + |b_m|).\epsilon_1| + |b_\ell.\epsilon_1|$$

As $|\epsilon_1| \leq u$, $|b_m| \leq u.|b_h| \leq u.|a + b_h|$ and $|b_\ell| \leq u.|b_m| \Rightarrow$

$$|\delta| \leq \frac{u}{u+1}.(\frac{u}{u+1}.|a + b_h| + u.|a + b_h|).u + u.u^2.|a + b_h|$$

After calculating, we find:

$$|\delta| \leq \frac{u^5 + 3.u^4 + 3.u^3}{(u+1)^2}.|a + b_h|$$

We search the upper bound of $|a + b_h|$ in function of $|a + b_h + b_m + b_\ell|$: first, we calculate the upper bound of $|b_m + b_\ell|$ in function of $|a + b_h|$:

$$|b_m + b_\ell| \leq |b_m| + |b_\ell|$$

$$|b_m + b_\ell| \leq u.|a + b_h| + u^2.|a + b_h|$$

We deduce:

$$|a + b_h + b_m + b_\ell| \geq (1 - (u + u^2)).|a + b_h|$$

$$|a + b_h + b_m + b_\ell| \geq |(1 - u - u^2)|.|a + b_h|$$

$$\frac{1}{(1 - u - u^2)}.|a + b_h + b_m + b_\ell| \geq |a|$$

$\Rightarrow$

$$|\delta| \leq \frac{u^5 + 3.u^4 + 3.u^3}{(u+1)^2.(1 - u - u^2)}.|a + b_h + b_m + b_\ell|$$

$$|\delta| \leq |\epsilon|.|a + b_h + b_m + b_\ell|$$

We have $\frac{u^5+3.u^4+3.u^3}{(u+1)^2.(1-u-u^2)} \leq 3.u^3 \Rightarrow |\epsilon| \leq 3.u^3$

Theorem 8 (Relative error algorithm *Add*133 with a bounded cancellation) *Let a a **double** number and (b_h, b_m, b_ℓ) a **triple-double** number . We have for the algorithm Add133 with a and (b_h, b_m, b_ℓ) it's arguments.*
If a and b_h are different sign and we suppose $|b_h| \leq 2^{-\mu}|a|$ with $\mu \geq 1$.
So :

$$r_h + r_m + r_\ell = (a + (b_h + b_m + b_\ell)).(1 + \epsilon)$$

with

$$|\epsilon| \leq \frac{u^5 + 3.u^4 + 3.u^3}{(u+1)^2}.\frac{1 - 2^{-\mu}}{1 - (1 + u + u^2).2^{-\mu}} \leq 6.u^3$$

Proof According to the calculation technique of the theorem 4.3 proof's([12]).
We suppose $|b_h| \leq 2^{-\mu}|a|$ with $\mu \geq 1$. First we look for the upper bound, using the results of the proof of Theorem 7. We have:

$$|b_\ell| \leq u.|b_m| \leq u^2.|b_h| \leq u^2.2^{-\mu}|a|$$

then we search the lower bound of $|a + b_h|$ in function of $|a|$:

$$|a + b_h| \geq (1 - 2^{-\mu}).|a|$$

Now, we search the upper bound of $|b_m|$ and $|b_h|$ in function of $|a + b_h|$.

$$|b_m| \leq \frac{u.2^{-\mu}}{1 - 2^{-\mu}}|a + b_h|$$

and

$$|b_\ell| \leq \frac{u^2.2^{-\mu}}{1 - 2^{-\mu}}|a + b_h|$$

As :

$$|\delta| \leq \frac{u^5 + 3.u^4 + 3.u^3}{(u+1)^2}.|a + b_h|$$

Now, we search the lower bound for $|a + b_h + b_m + b_\ell|$ depending on $|a + b_h|$.
We have:

$$|b_m + b_\ell| \leq |b_m| + |b_\ell|$$

$$|b_m + b_\ell| \leq (u + u^2).2^{-\mu}.|a|$$

$$|b_m + b_\ell| \leq \frac{(u + u^2).2^{-\mu}}{1 - 2^{-\mu}}|a + b_h|$$

$$|a + b_h + b_m + b_\ell \geq |a_h + b_h|\frac{1 - 2^{-\mu} - (u + u^2).2^{-\mu}}{1 - 2^{-\mu}}$$

So we have for $|\delta|$:

$$|\delta| \leq |a + b_h + b_m + b_\ell|.\frac{u^5 + 3.u^4 + 3.u^3}{(u+1)^2}.\frac{1 - 2^{-\mu}}{1 - (1 + u + u^2).2^{-\mu}}$$

So:

$$|\epsilon| \leq \frac{u^5 + 3.u^4 + 3.u^3}{(u+1)^2}.\frac{1 - 2^{-\mu}}{1 - (1 + u + u^2).2^{-\mu}}$$

As $\mu \geq 1$, we want the upper bound of $\frac{1-2^{-\mu}}{1-(1+u+u^2).2^{-\mu}} \Rightarrow$

$$\frac{1}{1/2} \leq 2$$

So :

$$|\epsilon| \leq 6.u^3$$

4.1.2 Add333

See algorithm 8

Lemma 8 (Add333) *Let (a_h, a_m, a_ℓ) and (b_h, b_m, b_ℓ) are **triple-Double** number, r_h, r_m and r_ℓ result of $Add333(a_h, a_m, a_\ell, b_h, b_m, b_\ell)$ for the 4 modes of rounding, considering that there is no **overflow** so: $|r_\ell| \leq u.|r_m|$, $|r_m| \leq 2^{-2}.|r_h|$ and $|r_\ell| \leq 2^{-2}.u.|r_h|$*

Proof

- $r_h + r_\ell = Add112(t_7, t_8)$ According to $Add112 \Rightarrow |r_\ell| \leq u.|r_h|$.
- $t_8 = \circ(t_5 + t_6)$ according to collary 5: $t_8 = (t_5 + t_6).(1 + \epsilon_1)$ with $|\epsilon_1| \leq u$
- $t_5 = \circ(t_3 + t_4)$ according to collary 5: $t_5 = (t_3 + t_4).(1 + \epsilon_2)$ with $|\epsilon_2| \leq u$
- $t_6 = \circ(a_\ell + b_\ell)$ according to collary 5: $t_6 = (a_\ell + b_\ell).(1 + \epsilon_3)$ with $|\epsilon_3| \leq u$.
- $t_7 + t_4 = Add112(t_1, t_2)$ according to $Add112$: $|t_4| \leq u.|t_7| \Rightarrow |t_7 + t_4| \geq (1 - u).|t_7| \Rightarrow (1 - u).|t_7| \leq |t_1 + t_2| \Rightarrow |t_7| \leq \frac{1}{1-u}.|t_1 + t_2| \Rightarrow |t_4| \leq \frac{u}{1-u}.|t_1 + t_2|$
- $t_2 + t_3 = Add112Cond(a_m, b_m)$ according to $Add112$: $|t_3| \leq u.|t_2| \Rightarrow |t_3| \leq \frac{u}{1-u}.|a_m + b_m|$

- $|t_5| \leq (|t_3| + |t_4|)(1+u) \Rightarrow |t_5| \leq (\frac{u}{1-u}.|a_m + b_m| + \frac{u}{1-u}.|t_1 + t_2|)(1+u)$ thanks to the condition $Add112 \Rightarrow |t_1| \geq |t_2| \Rightarrow$

$$|t_5| \leq (\frac{u}{1-u}.|a_m + b_m| + \frac{2.u}{1-u}.|t_1|)(1+u)$$

$$|t_5| \leq (u.|a_m + b_m| + 2.u.|t_1|)$$

As $|t_1| \leq u.|r_h| \Rightarrow$

$$|t_5| \leq (u.|a_m + b_m| + 2.u^2.|r_h|)$$

As $|a_m| \leq 2^{-\alpha_o}.|a_h|$ and that $|b_m| \leq 2^{-\beta_o}.|b_h| \leq 2^{-\beta_o}.\frac{3}{4}.|a_h|$, so we have:

$$|a_m + b_m| \leq (2^{-\alpha_o} + 2^{-\beta_o}.\frac{3}{4}).|a_h|$$

We search the lower bound of $|a_h|$ in function of $|a_h + b_h|$ and as $b_h \leq \frac{3}{4}).|a_h|$

$$|a_h + b_h| \geq (1 - \frac{3}{4}).|a_h|$$

$$|a_h| \leq 4.|a_h + b_h|$$

$\Rightarrow$

$$|a_m + b_m| \leq (2^{-\alpha_o} + 2^{-\beta_o}.\frac{3}{4}).4.|a_h + b_h|$$

$$|a_m + b_m| \leq (2^{-\alpha_o+2} + 2^{-\beta_o}.3).|a_h + b_h|$$

But we know that $|a_h + b_h| \leq (1+u).|r_h|$

$$|a_m + b_m| \leq (2^{-\alpha_o+2} + 2^{-\beta_o}.3).(1+u).|r_h|$$

but $\alpha_o \geq 4$ and $\beta_o \geq 4$

$$|a_m + b_m| \leq 7.2^{-4}.(1+u).|r_h|$$

$$|t_5| \leq (u.7.2^{-4}.(1+u).|r_h| + 2.u^2.|r_h|)$$

After calculating, we have

$$|t_5| \leq (2 + 7.2^{-4}).u^2 + 7.2^{-4}.u).|r_h|$$

We know that $t_6 = (a_\ell + b_\ell)(1 + \epsilon_3) \Rightarrow$

$$|t_6| \leq |a_\ell + b_\ell|.(1+u)$$

As $|a_\ell| \leq 2^{-\alpha_u}.|a_m| \leq 2^{-\alpha_u-\alpha_o}.|a_h|$ and that $|b_\ell| \leq 2^{-\beta_u}.|b_m| \leq 2^{-\beta_u-\beta_o}.|b_h| \leq 2^{-\beta_u-\beta_o}.\frac{3}{4}.|a_h|$, so we have:

$$|a_\ell + b_\ell| \leq (2^{-\alpha_u-\alpha_o} + 2^{-\beta_u-\beta_o}.\frac{3}{4}).|a_h|$$

$$|a_\ell + b_\ell| \leq (2^{-\alpha_u-\alpha_o+2} + 2^{-\beta_u-\beta_o}.3).|a_h + b_h|$$

As $\alpha_o \geq 4$, $\alpha_u \geq 1$, $\beta_o \geq 4$ and $\beta_u \geq 1$, $\Rightarrow$

$$|a_\ell + b_\ell| \leq (2^{-3} + 2^{-5}.3).|a_h + b_h|$$

$$|a_\ell + b_\ell| \leq 7.2^{-5}.|a_h + b_h|$$

$\Rightarrow$

$$|t_6| \leq 7.2^{-5}.(1+u).|r_h|.(1+u)$$
$$|t_6| \leq (7.2^{-5} + 7.2^{-4}.u + 7.2^{-5}.u^2).|r_h|$$

$\Rightarrow$

$$|t_8| \leq (|t_5| + |t_6|).(1+u)$$
$$|t_8| \leq ((2+7.2^{-4}).u^2+7.2^{-4}.u).|r_h|+(7.2^{-5}+7.2^{-4}.u+7.2^{-5}.u^2).|r_h|).(1+u)$$

After the calculations, we have :

$$|t_8| \leq ((2+7.2^{-4}+7.2^{-5}).u^3+(2+7.2^{-4}+7.2^{-5}+7.2^{-3}).u^2+(7.2^{-5}+7.2^{-3}).u+7.2^{-5}).|r_h|$$

Now we look at the calculation of upper bound of $|t_7|$ in function of $|r_h|$. we have $|t_7| \leq \frac{1}{1-u}.|t_1+t_2|$ thanks to the condition of $Add112$, we have:

$$|t_7| \leq frac21 - u.|t_1|$$

As $|t_1| \leq u.|r_h| \Rightarrow$

$$|t_7| \leq \frac{2.u}{1-u}.|r_h|$$

We have :

$$|r_m + r_\ell| \leq |t_7| + |t_8|$$

$$|r_m+r_\ell| \leq \frac{2.u}{1-u}.|r_h|+((2+7.2^{-4}+7.2^{-5}).u^3+(2+7.2^{-4}+7.2^{-5}+7.2^{-3}).u^2+(7.2^{-5}+7.2^{-3}).u+7$$

We have $\frac{2.u}{1-u} + ((2+7.2^{-4}+7.2^{-5}).u^3 + (2+7.2^{-4}+7.2^{-5}+7.2^{-3}).u^2 + (7.2^{-5}+7.2^{-3}).u + 7.2^{-5}) \leq 2^{-2} \Rightarrow$

$$|r_m + r_\ell| \leq 2^{-2}.|r_h|$$

As $|r_h| \leq u.|r_\ell| \Rightarrow (1-u).|r_m| \leq |r_m + r_\ell| \Rightarrow$

$$|r_m| \leq \frac{1}{1-u}.2^{-2}.|r_h|$$

$$|r_m| \leq 2^{-2}.|r_h|$$

and

$$|r_\ell| \leq 2^{-2}.u.|r_h|$$

Theorem 9 (Relative errror algorithm *Add*333) *Let* (a_h, a_m, a_ℓ) *and* (b_h, b_m, b_ℓ) *are* ***triple-double*** *numbers and arguments of the function* $Add333$*. So:*

$$r_h + r_m + r_\ell = ((a_h + a_m + a_\ell) + (b_h + b_m + b_\ell)).(1+\epsilon)$$

with $|\epsilon| \leq (4.u^4 + 5.u^3 + u^2).2^{-min(\alpha_o,\beta_o)+5} + (u^2 + 2.u).2^{-min(\alpha_u+\alpha_o,\beta_u+\beta_o)+5}$

Proof According to the calculation technique of the theorem 5.2 prof's([12]). According to $Add333$ and thanks to $Add112$, we have these results:

$$|r_\ell| \leq u.|r_\ell|$$

$$|t_4| \leq u.|t_7|$$

$$|t_3| \leq u.|t_2|$$

$$|t_1| \leq u.|r_h|$$

We search the upper bound of $|r_h|$ in function of $|a_h|$.

$$|t_1 + r_h| = |a_h + b_h|$$

As $|b_h| \leq \frac{3}{4}.|a_h|$

$$|t_1 + r_h| \leq (1 + \frac{3}{4}).|a_h|$$

$$|t_1 + r_h| \leq \frac{7}{4}.|a_h| \leq 2.|a_h|$$

As $|t_1| \leq u.|r_h|$

$$|t_1 + r_h| \geq (1-u).|r_h|$$

So we have:

$$|r_h| \leq \frac{2}{1-u}.|a_h|$$

As $|t_1| \leq u.|r_h|$

$$|t_1| \leq \frac{2.u}{1-u}.|a_h|$$

Now, we calculate for all $|t_i|$ with $2 \leq i \leq 8$ their upper bounds in function of $|a_h|$. As calculated previously with r_h , we have:

$$|t_2| \leq \frac{1}{1-u}.|a_m + b_m|$$

As $|a_m| \leq 2^{-\alpha_o}.|a_h|$ and that $|b_m| \leq 2^{-\beta_o}.|b_h| \leq 2^{-\beta_o}.\frac{3}{4}.|a_h| \Rightarrow$

$$|t_2| \leq \frac{1}{1-u}.(2^{-\alpha_o} + 2^{-\beta_o}.\frac{3}{4}).|a_h|$$

As $\frac{3}{4} \leq 1 \Rightarrow$

$$|t_2| \leq \frac{1}{1-u}.(2^{-\alpha_o} + 2^{-\beta_o}).|a_h|$$

$$|t_2| \leq \frac{1}{1-u}.2^{-min(\alpha_o,\beta_o)+1}.|a_h|$$

As $|t_3| \leq u.|t_2| \Rightarrow$

$$|t_3| \leq \frac{u}{1-u}.2^{-min(\alpha_o,\beta_o)+1}.|a_h|$$

Same for $|t_7|$, we have:

$$|t_7| \leq \frac{1}{1-u}.|t_1 + t_2|$$

$$|t_7| \leq \frac{1}{1-u}.(\frac{2.u}{1-u}.|a_h| + \frac{1}{1-u}.2^{-min(\alpha_o,\beta_o)+1}.|a_h|)$$

$$|t_7| \leq \frac{1}{(1-u)^2}.(2.u + 2^{-min(\alpha_o,\beta_o)+1}).|a_h|$$

In case where $2.u = 2^{-min(\alpha_o,\beta_o)+1}$, we have:

$$|t_7| \leq \frac{1}{(1-u)^2}.2^{-min(\alpha_o,\beta_o)+2}.|a_h|$$

As $|t_4| \leq u.|t_7| \Rightarrow$

$$|t_4| \leq \frac{u}{(1-u)^2}.2^{-min(\alpha_o,\beta_o)+2}.|a_h|$$

For t_6:

$$|t_6| \leq (|a_\ell| + |b_\ell|).(1+u)$$

As $|a_\ell| \leq 2^{-\alpha_u}.|a_m| \leq 2^{-\alpha_u-\alpha_o}.|a_h|$ and $|b_\ell| \leq 2^{-\beta_u}.|b_m| \leq 2^{-\beta_u-\beta_o}.|b_h| \leq 2^{-\beta_u-\beta_o}.\frac{3}{4}.|a_h| \Rightarrow$

$$|t_6| \leq (2^{-\alpha_u-\alpha_o}.|a_h| + 2^{-\beta_u-\beta_o}.\frac{3}{4}.|a_h|).(1+u)$$

$$|t_6| \leq (1+u).2^{-min(\alpha_u+\alpha_o,\beta_u+\beta_o)+1}.|a_h|$$

And t_5, we have:

$$|t_5| \leq (|t_3| + |t_4|).(1+u)$$

$$|t_5| \leq (1+u).(\frac{u}{1-u}.2^{-min(\alpha_o,\beta_o)+1}.|a_h| + \frac{u}{(1-u)^2}.2^{-min(\alpha_o,\beta_o)+2}.|a_h|)$$

$$|t_5| \leq (1+u).\frac{u}{(1-u)^2}((1-u).2^{-min(\alpha_o,\beta_o)+1}. + u.2^{-min(\alpha_o,\beta_o)+2}).|a_h|$$

$$|t_5| \leq (1+u).\frac{u^2}{(1-u)^2}.(1+u).2^{-min(\alpha_o,\beta_o)+1}.|a_h|$$

After the calculation:

$$|t_5| \leq \frac{u^3+2.u^2+u}{(1-u)^2}.2^{-min(\alpha_o,\beta_o)+1}.|a_h|$$

Simplifying, we have:
As $\frac{u^3+2.u^2+u}{(1-u)^2} \leq u \Rightarrow$

$$|t_5| \leq u.2^{-min(\alpha_o,\beta_o)+1}.|a_h|$$

and similarly for the others:

$$|r_h| \leq 2.|a_h|$$

$$|t_1| \leq 2.u.|a_h|$$

$$|t_2| \leq 2^{-min(\alpha_o,\beta_o)+1}.|a_h|$$

$$|t_3| \leq u.2^{-min(\alpha_o,\beta_o)+1}.|a_h|$$

$$|t_4| \leq u^2.2^{-min(\alpha_o,\beta_o)+2}.|a_h|$$

$$|t_6| \leq 2^{-min(\alpha_u+\alpha_o,\beta_u+\beta_o)+1}.|a_h|$$

$$|t_7| \leq u.2^{-min(\alpha_o,\beta_o)+2}.|a_h|$$

As $r_m + r_\ell = t_7 + t_8$, we calculated t_8:

$$t_8 = (t_5 + t_6).(1+\epsilon_1)$$

with $|\epsilon_1| \leq u$

$$t_8 = ((t_3 + t_4).(1 + \epsilon_2) + (a_\ell + b_\ell).(1 + \epsilon_3)).(1 + \epsilon_1)$$

with $|\epsilon_2| \leq u$ and $|\epsilon_3| \leq u$ After the calculations:

$$t_8 = t_3 + t_4 + a_\ell + b_\ell + \delta$$

with $\delta = t_3.(\epsilon_2 + \epsilon_1 + \epsilon_2.\epsilon_1) + t_4.(\epsilon_2 + \epsilon_1 + \epsilon_2.\epsilon_1) + a_\ell.(\epsilon_3 + \epsilon_1 + \epsilon_3.\epsilon_1) + b_\ell.(\epsilon_3 + \epsilon_1 + \epsilon_3.\epsilon_1)$
We seek the upper bound of $|\delta|$ in function of $|a_h|$:

$|\delta| \leq |t_3.(\epsilon_2+\epsilon_1+\epsilon_2.\epsilon_1)|+|t_4.(\epsilon_2+\epsilon_1+\epsilon_2.\epsilon_1)|+|a_\ell.(\epsilon_3+\epsilon_1+\epsilon_3.\epsilon_1)|+|b_\ell.(\epsilon_3+\epsilon_1+\epsilon_3.\epsilon_1)|$

As $|\epsilon_i| \leq u$ with $1 \leq i \leq 3 \Rightarrow$

$$|\delta| \leq (2.u + u^2).(|t_3| + |t_4| + |a_\ell| + |b_\ell|)$$

$|\delta| \leq (2.u+u^2).(u.2^{-min(\alpha_o,\beta_o)+1}.|a_h|+u^2.2^{-min(\alpha_o,\beta_o)+2}.|a_h|+2^{-\alpha_u-\alpha_o}.|a_h|+2^{-\beta_u-\beta_o}.\frac{3}{4}.|a_h|)$

$|\delta| \leq (2.u+u^2).(u.2^{-min(\alpha_o,\beta_o)+1}.|a_h|+u^2.2^{-min(\alpha_o,\beta_o)+2}.|a_h|+2^{-\alpha_u-\alpha_o}.|a_h|+2^{-\beta_u-\beta_o}|a_h|)$
$|\delta| \leq (2.u+u^2).(u.2^{-min(\alpha_o,\beta_o)+1}.|a_h|+u^2.2^{-min(\alpha_o,\beta_o)+2}.|a_h|+2^{-min(\alpha_u+\alpha_o,\beta_u+\beta_o)+1}.|a_h|)$
$|\delta| \leq ((4.u^4+5.u^3+u^2).2^{-min(\alpha_o,\beta_o)+1}.+(u^2+2.u).2^{-min(\alpha_u+\alpha_o,\beta_u+\beta_o)+1}).|a_h|$
We look the lower bound of $|a_h|$ in function of $|a_h + b_h + a_m + b_m + a_\ell + b_\ell|$:

$$|b_h + a_m + b_m + a_\ell + b_\ell| \leq |b_h| + |a_m| + |b_m| + |a_\ell| + |b_\ell|$$

$|b_h+a_m+b_m+a_\ell+b_\ell| \leq \frac{3}{4}.|a_h|+2^{-\alpha_o}.|a_h|+\frac{3}{4}.2^{-\beta_o}.|a_h|+2^{-\alpha_o-\alpha_u}|a_h|+\frac{3}{4}.2^{-\beta_o-\beta_u}.|a_h|$
with the condition of this algorithm, after calculate:

$$|b_h + a_m + b_m + a_\ell + b_\ell| \leq \frac{117}{128}.|a_h|$$

As $\frac{117}{128} < \frac{15}{16} \Rightarrow$

$$|b_h + a_m + b_m + a_\ell + b_\ell| \leq \frac{15}{16}.|a_h|$$

We can say that:

$$|a_h + b_h + a_m + b_m + a_\ell + b_\ell| \geq \frac{1}{16}.|a_h|$$

We have :

$|\delta| \leq 16.((4.u^4+5.u^3+u^2).2^{-min(\alpha_o,\beta_o)+1}+(u^2+2.u).2^{-min(\alpha_u+\alpha_o,\beta_u+\beta_o)+1}).|a_h+b_h+a_m+b_m+a_\ell+$
$|\delta| \leq ((4.u^4+5.u^3+u^2).2^{-min(\alpha_o,\beta_o)+5}+(u^2+2.u).2^{-min(\alpha_u+\alpha_o,\beta_u+\beta_o)+5}).|a_h+b_h+a_m+b_m+a_\ell+b_\ell|$
So we have $|\epsilon| \leq (4.u^4+5.u^3+u^2).2^{-min(\alpha_o,\beta_o)+5}+(u^2+2.u).2^{-min(\alpha_u+\alpha_o,\beta_u+\beta_o)+5}$

4.2 Multiplication Operators

4.2.1 Mul133

See algorithm 9

Lemma 9 (Mul133) *Let a a* ***double*** *number and* (b_h, b_m, b_ℓ) *a* ***triple-Double*** *number,* r_h*,* r_m *and* r_ℓ *result of* $Mul133(a, b_h, b_m, b_\ell)$ *for the* 4 *modes of rounding, considering that there is no* ***overflow*** *so:* $|r_\ell| \leq u.|r_m|$ *,* $|r_m| \leq (u + 2^{-\beta_o+1} + 2^{-\beta_o-\beta_u+1}).|r_h|$ *and* $|r_\ell| \leq u.(u^2 + 2^{-\beta_o+1} + 2^{-\beta_o-\beta_u+1}).|r_h|$*.*

Proof According to $Mul133$, we have :
$r_m + r_\ell = Add112(t_9, t_{10})$ According to $Add112 \Rightarrow |r_\ell| \leq u.|r_m|$.
$t_{10} = \circ(t_7 + t_8)$ According to collary 5: $t_{10} = (t_7 + t_8).(1 + \epsilon_1)$ with $|\epsilon_1| \leq u$
$t_8 = \circ(t_4 + t_5)$ According to collary 5: $t_8 = (t_4 + t_5).(1 + \epsilon_2)$ with $|\epsilon_2| \leq u$
$t_9 + t_7 = Add112(t_2, t_3)$ According to $Add112 \Rightarrow |t_7| \leq u.|t_9|$ and $t_9 + t_7 = t_2 + t_3$.
$t_5 = \circ(a.b_\ell)$ According to collary 5: $t_5 = (a.b_\ell).(1 + \epsilon_3)$ with $|\epsilon_3| \leq u$
$t_3 + t_4 = Add112(a, b_m)$ According to $Add112 \Rightarrow |t_4| \leq u.|t_3|$ and $t_3 + t_4 = a + b_m$.
$r_h + t_2 = Mul112(a, b_h)$ According to $Mul112 \Rightarrow |t_2| \leq u.|r_h|$ and $r_h + t_2 = a.b_h$.
We search the upper bound for $|a.b_h|$ in function of $|r_h|$ to find the upper bound of $|a.b_m|$

$$|a.b_h| \leq |r_h| + |t_2|$$

As $|t_2| \leq u.|r_h| \Rightarrow$

$$|a.b_h| \leq (1 + u).|r_h|$$

As $|b_m| \leq 2^{-\beta_o}.|b_h| \Rightarrow$

$$|a.b_m| \leq 2^{-\beta_o}.|a.b_h|$$

$$|a.b_m| \leq 2^{-\beta_o}.(1 + u).|r_h|$$

We have $t_3 + t_4 = a.b_m$, so:

$$|t_3 + t_4| \leq 2^{-\beta_o}.(1 + u).|r_h|$$

As $|t_4|u.|t_3|$, so we have:

$$(1 - u).|t_3| \leq |t_3 + t_4|$$

$\Rightarrow$

$$|t_3| \leq 2^{-\beta_o}.(1 + u).\frac{1}{1 - u}.|r_h|$$

and

$$|t_4| \leq 2^{-\beta_o}.(1+u).\frac{u}{1-u}.|r_h|$$

As $|b_\ell| \leq 2^{-\beta_u}.|b_m| \Rightarrow |b_\ell| \leq 2^{-\beta_o-\beta_u}.|b_h| \Rightarrow$

$$|a.b_\ell| \leq 2^{-\beta_o-\beta_u}.|a.b_h|$$

$$|a.b_\ell| \leq 2^{-\beta_o-\beta_u}.(1+u).|r_h|$$

As $t_5 = (a.b_\ell).(1+\epsilon_3) \Rightarrow$

$$|t_5| \leq |(a.b_\ell).(1+\epsilon_3)|$$

As $|\epsilon_3| \leq u \Rightarrow$

$$|t_5| \leq (1+u)^2.2^{-\beta_o-\beta_u}.|r_h|$$

As $t_9 + t_7 = t_2 + t_3$ and $|t_7| \leq u.|t_9|$:

$$(1-u).|t_9| \leq |t_9 + t_7|$$

$$|t_9| \leq \frac{1}{1-u}.(|t_2| + |t_3|)$$

$$|t_9| \leq \frac{1}{1-u}.(u.|r_h| + 2^{-\beta_o}.(1+u).\frac{1}{1-u}.|r_h|)$$

$$|t_9| \leq \frac{1}{1-u}.(u + 2^{-\beta_o}\frac{u+1}{1-u}).|r_h|$$

and so: $|t_7| \leq \frac{u}{1-u}.(u + 2^{-\beta_o}\frac{u+1}{1-u}).|r_h|$ We know that $t_8 = (t_4+t_5).(1+\epsilon_2)$ and that $|\epsilon_2| \leq u \Rightarrow$

$$|t_8| \leq (1+u).(|t_4| + |t_5|)$$

$$|t_8| \leq (1+u).(2^{-\beta_o}.(1+u).\frac{u}{1-u}.|r_h| + (1+u)^2.2^{-\beta_o-\beta_u}.|r_h|)$$

$$|t_8| \leq (1+u)^2.(2^{-\beta_o}.\frac{u}{1-u} + (1+u).2^{-\beta_o-\beta_u}).|r_h|$$

We have $t_{10} = (t_7+t_8).(1+\epsilon_1)$ and that $|\epsilon_1| \leq u \Rightarrow$

$$|t_{10}| \leq (1+u).(|t_7| + |t_8|)$$

$$|t_{10}| \leq (1+u).(\frac{u}{1-u}.(u+2^{-\beta_o}\frac{u+1}{1-u})+(1+u)^2.(2^{-\beta_o}.\frac{u}{1-u}+(1+u).2^{-\beta_o-\beta_u})).|r_h|$$

$$|t_{10}| \leq (\frac{u^3+u^2}{1-u} + 2^{-\beta_o}\frac{-u^5-2.u^4+u^3+4.u^2+2u}{(1-u)^2} + 2^{-\beta_o-\beta_u}.(u+1)^4).|r_h|$$

And finally, $r_h + r_\ell = t_9 + t_{10}$, so:

$$|r_m + r_\ell| \leq \frac{1}{1-u}.(u + 2^{-\beta_o}\frac{u+1}{1-u}).|r_h|$$

$$+(\frac{u^3+u^2}{1-u} + 2^{-\beta_o}\frac{-u^5-2.u^4+u^3+4.u^2+2u}{(1-u)^2} + 2^{-\beta_o-\beta_u}.(u+1)^4).|r_h|$$

$$|r_m+r_\ell| \leq (\frac{u^3+u^2+u}{1-u}+2^{-\beta_o}.\frac{-u^5-2.u^4+u^3+4.u^2+3.u+1}{(1-u)^2}+2^{-\beta_o-\beta_u}.(1+u)^4).|r_h|$$

As $\frac{u^3+u^2+u}{1-u} \leq u$, $|\frac{-u^5-2.u^4+u^3+4.u^2+3.u+1}{(1-u)^2}| \leq 2$ and $(1+u)^4 \leq 2$

$$|r_m + r_\ell| \leq (u + 2^{-\beta_o+1} + 2^{-\beta_o-\beta_u+1}).|r_h|$$

As $|r_\ell| \leq u.|r_m| \Rightarrow$

$$(1-u).|r_m| \leq |r_m + r_\ell|$$

$$(1-u).|r_m| \leq (u + 2^{-\beta_o+1} + 2^{-\beta_o-\beta_u+1}).|r_h|$$

$$|r_m| \leq |\frac{1}{1-u}|.(u + 2^{-\beta_o+1} + 2^{-\beta_o-\beta_u+1}).|r_h|$$

$$|r_m| \leq (u + 2^{-\beta_o+1} + 2^{-\beta_o-\beta_u+1}).|r_h|$$

and

$$|r_\ell| \leq u.(u^2 + 2^{-\beta_o+1} + 2^{-\beta_o-\beta_u+1}).|r_h|$$

Theorem 10 (Relative error algorithm *Mul*133) *Let a a **double** number and (b_h, b_m, b_ℓ) a **triple-double** number are the arguments of the function $Mul133$.*
So:

$$r_h + r_m + r_\ell = (a.(b_h + b_m + b_\ell)).(1+\epsilon)$$

with $|\epsilon| \leq \frac{u^4+4.u^3.2^{-\beta_o}+4.u.2^{-\beta_o-\beta_u}}{1-(2^{-\beta_o}+2^{-\beta_o-\beta_u})} \leq 2.u^4 + u^3.2^{-\beta_o+3} + u.2^{-\beta_o-\beta_u+3}$.

PROOF According to the calculation technique of the theorem 5.2 prof's([12]). According to $Add333$ and thanks to $Add112$ and $Mul112$, we have these results:

$$|t_2| \leq u.|r_h|$$

$$|t_4| \leq u.|t_3|$$

$$|t_7| \leq u.|t_9|$$

$$|r_\ell| \leq u.|r_m|$$

We search for all $|t_i|$ with $2 \leq i \leq 10$, their upper bound in function of $|a.b_h|$.
As $r_h + t_2 = a.b_h$ and $|t_2| \leq u.|r_h|$, we have $(1-u).|r_h| \leq |r_h + t_2| \Rightarrow$

$$(1-u).|r_h| \leq |a.b_h|$$

$$|r_h| \leq |\frac{1}{1-u}.|a.b_h|$$

$\Rightarrow$

$$|t_2| \leq |\frac{u}{1-u}.|a.b_h|$$

As $|b_m| \leq 2^{-\beta_o}.|b_h| \Rightarrow |a.b_m| \leq 2^{-\beta_o}.|a.b_h| \Rightarrow$

$$t_3 + t_4 = a.b_m$$

As $|t_4| \leq u.|t_3| \Rightarrow (1-u).|t_3| \leq |t_3 + t_4|$

$$(1-u).|t_3| \leq |a.b_m|$$

$$|t_3| \leq \frac{1}{1-u}.2^{-\beta_o}.|a.b_h|$$

$\Rightarrow$

$$|t_4| \leq \frac{u}{1-u}.2^{-\beta_o}.|a.b_h|$$

We have $t_5 = \circ(a.b_\ell)$ After the collary $t_5 = a.b_\ell.(1+\epsilon_3)$ with $|\epsilon_3| \leq u$.
As $|b_\ell| \leq 2^{-\beta_u}.|b_m| \leq 2^{-\beta_o-\beta_u}.|b_h| \Rightarrow |a.b_\ell| \leq 2^{-\beta_o-\beta_u}.|a.b_h|$

$$t_5 = a.b_\ell.(1+\epsilon_3)$$

$$|t_5| \leq |a.b_\ell|.(1+u)$$

$$|t_5| \leq (1+u).2^{-\beta_o-\beta_u}.|a.b_h|$$

After calculation, we have:

$$|t_9| \leq (u + 2^{-\beta_o}).|\frac{1}{1-u}.|a.b_h|$$

$$|t_7| \leq (u + 2^{-\beta_o}).|\frac{u}{1-u}.|a.b_h|$$

$$|t_8| \leq (\frac{u^3+u^2}{1-u}.2^{-\beta_o} + (1+u)^2.2^{-\beta_o-\beta_u}).|a.b_h|$$

$$|t_{10}| \leq \frac{1}{u-1}(u^3+u^2+(u^4+3.u^3+2.u^2).2^{-\beta_o}+(u^4+2.u^3-2.u-1).2^{-\beta_o-\beta_u}).|a.b_h|$$

We know that $r_m + r_\ell = t_9 + t_{10}$, we start by calculating t_{10}:

$$t_{10} = (t_7 + t_8)(1 + \epsilon_1)$$

with $|\epsilon_1| \le u$, then we calculate t_8:

$$t_8 = (t_4 + t_5).(1 + \epsilon_2)$$

with $|\epsilon_2| \le u$, then we calculate t_5:

$$t_5 = a.b_\ell.(1 + \epsilon_3)$$

with $|\epsilon_3| \le u$

$$t_8 = (t_4 + a.b_\ell.(1 + \epsilon_3)).(1 + \epsilon_2)$$

$$t_{10} = (t_7 + (t_4 + a.b_\ell.(1 + \epsilon_3)).(1 + \epsilon_2))(1 + \epsilon_1)$$

After the calculation, we have :

$$t_{10} = t_7 + t_4 + a.b_\ell + \delta$$

with $\delta = t_7.\epsilon_1 + (t_4 + a.b_\ell).(\epsilon_3 + \epsilon_2 + \epsilon_3.\epsilon_2 + \epsilon_1 + \epsilon_3.\epsilon_1 + \epsilon_2.\epsilon_1 + \epsilon_3.\epsilon_2.\epsilon_1)$ As $r_h + r_m + r_\ell = r_h + t_9 + t_{10}$, we have:

$$r_h + r_m + r_\ell = r_h + t_9 + t_7 + t_4 + a.b_\ell + \delta$$

As $t_9 + t_7 = t_2 + t_3 \Rightarrow$

$$r_h + r_m + r_\ell = r_h + t_2 + t_3 + t_4 + a.b_\ell + \delta$$

As $r_h + t_3 = a.b_h$ and $t_3 + t_4 = a.b_m \Rightarrow$

$$r_h + r_m + r_\ell = a.b_h + a.b_m + a.b_\ell + \delta$$

$$r_h + r_m + r_\ell = a.(b_h + b_m + b_\ell) + \delta$$

We seek the upper bound of $|\delta|$ in function of $|a.b_h|$, So:

$$|\delta| \le |t_7.\epsilon_1| + (|t_4| + |a.b_\ell|).|\epsilon_3 + \epsilon_2 + \epsilon_3.\epsilon_2 + \epsilon_1 + \epsilon_3.\epsilon_1 + \epsilon_2.\epsilon_1 + \epsilon_3.\epsilon_2.\epsilon_1|$$

As $|\epsilon_i| \le u$ with $1 \le i \le 3 \Rightarrow$

$$|\delta| \le u.|t_7| + (3.u + 3.u^2 + u^3).(|t_4| + |a.b_\ell|)$$

$$|\delta| \le u.(u+2^{-\beta_o}).|\frac{u^2}{u-1}.|a.b_h|+(3.u+3.u^2+u^3).(\frac{u^2}{u-1}.2^{-\beta_o}.|a.b_h|+2^{-\beta_o-\beta_u}.|a.b_h|)$$

$$|\delta| \leq (\frac{u^4}{u-1} + \frac{u^5+3.u^4+4.u^3}{u-1}.2^{-\beta_o} + (u^3+3.u^2+3.u).2^{-\beta_o-\beta_u}).|a.b_h|$$

We seek the upper bound of $|a.b_h|$ in function of $|a.(b_h+b_m+b_\ell)|$.

$$|a.b_m + a.b_\ell| \leq |a.b_m| + |a.b_\ell|$$

$$|a.b_m + a.b_\ell| \leq 2^{-\beta_o}.|a.b_h| + 2^{-\beta_o-\beta_u}.|a.b_h|$$

$$|a.b_m + a.b_\ell| \leq (2^{-\beta_o} + 2^{-\beta_o-\beta_u}).|a.b_h|$$

So we have:

$$|a.b_h + a.b_m + a.b_\ell| \geq (1-(2^{-\beta_o} + 2^{-\beta_o-\beta_u})).|a.b_h|$$

$$\frac{1}{1-(2^{-\beta_o} + 2^{-\beta_o-\beta_u})}|a.b_h + a.b_m + a.b_\ell| \geq |a.b_h|$$

$\Rightarrow$

$$|\delta| \leq (\frac{u^4}{u-1} + \frac{u^5+3.u^4+4.u^3}{u-1}.2^{-\beta_o} + (u^3+3.u^2+3.u).2^{-\beta_o-\beta_u}).\frac{1}{1-(2^{-\beta_o} + 2^{-\beta_o-\beta_u})}|a.b_h+a.b$$

As $\frac{u^4}{u-1} \leq u^4$, $\frac{u^5+3.u^4+4.u^3}{u-1} \leq 4.u^3$ and $u^3+3.u^2+3.u \leq 4.u \Rightarrow$

$$|\delta| \leq \frac{u^4 + 4.u^3.2^{-\beta_o} + 4.u.2^{-\beta_o-\beta_u}}{1-(2^{-\beta_o} + 2^{-\beta_o-\beta_u})}.|a.b_h + a.b_m + a.b_\ell|$$

As $1-(2^{-\beta_o} + 2^{-\beta_o-\beta_u}) \geq \frac{1}{2}$ with $\beta_o \geq 2$ and $\beta_u \geq 2$

$$|\delta| \leq (2.u^4 + u^3.2^{-\beta_o+3} + u.2^{-\beta_o-\beta_u+3}).|a.b_h + a.b_m + a.b_\ell|$$

So :

$$|\epsilon| \leq 2.u^4 + u^3.2^{-\beta_o+3} + u.2^{-\beta_o-\beta_u+3}$$

4.2.2 Mul233

See algorithm 10

Lemma 10 (Mul233) *Let* (a_h, a_ℓ) *a* ***double-double*** *number and* (b_h, b_m, b_ℓ) *a* ***triple-Double*** *number,* r_h*,* r_m *and* r_ℓ *result of* $Mul233(a_h, a_\ell, b_h, b_m, b_\ell)$ *for the 4 modes of rounding, considering that there is no* ***overflow*** *so:* $|r_\ell| \leq 4.u.|r_m|$ *,* $|r_m| \leq u.(2.u + 2^{-\beta_o} + 2^{-\beta_o-\beta_u}).|r_h|$ *and* $|r_\ell| \leq u.(2.u + 2^{-\beta_o} + 2^{-\beta_o-\beta_u}).|r_h|$*.*

PROOF According to $Mul233$,we have:

- $r_h + t_1 = Mul112(a_h, b_h)$ according to $Mul112 \Rightarrow r_h + t_1 = a_h.b_h$ and $|t_1| \leq u.|r_h| \Rightarrow$
$$|a_h.b_h| \leq (1+u).|r_h|$$
$$|a_h.b_h| \leq \frac{1}{u+1}.|r_h|$$

- $t_2 + t_3 = Mul112(a_h, b_m)$ according to $Mul112 \Rightarrow t_2 + t_3 = a_h.b_m$ and $|t_3| \leq u.|t_2| \Rightarrow$
$$|t_2 + t_3| \leq |a_h.b_m|$$
As $|b_m| \leq 2^{-\beta_o}.|b_h| \Rightarrow |a_h.b_m| \leq 2^{-\beta_o}.|a_h.b_h| \Rightarrow$
$$|t_2 + t_3| \leq 2^{-\beta_o}.|a_h.b_h|$$
As $|a_h.b_h| \leq \frac{1}{u+1}.|r_h|$
$$|t_2 + t_3| \leq 2^{-\beta_o}.\frac{1}{u+1}.|r_h|$$
As $|t_3| \leq u.|t_2| \Rightarrow$
$$(1-u).|t_2| \leq |t_2 + t_3|$$
$$(1-u).|t_2| \leq 2^{-\beta_o}.\frac{1}{u+1}.|r_h|$$
$$|t_2| \leq 2^{-\beta_o}.\frac{1}{1-u^2}.|r_h|$$
$\Rightarrow$
$$|t_3| \leq 2^{-\beta_o}.\frac{u}{1-u^2}.|r_h|$$
So not to write the repetitive calculations, I give the results for $|t_i|$ with $4 \leq i \leq 9$:
$$|t_4| \leq 2^{-\beta_o-\beta_u}.\frac{1}{1-u^2}.|r_h|$$
$$|t_5| \leq 2^{-\beta_o-\beta_u}.\frac{u}{1-u^2}.|r_h|$$
$$|t_6| \leq \frac{u}{1-u^2}.|r_h|$$
$$|t_7| \leq \frac{u^2}{1-u^2}.|r_h|$$
$$|t_8| \leq 2^{-\beta_o}.\frac{u}{1-u^2}.|r_h|$$
$$|t_9| \leq 2^{-\beta_o}.\frac{u^2}{1-u^2}.|r_h|$$

- $t_{10} = \circ(a_\ell.b_\ell)$ According to collary 5: $t_{10} = a_\ell.b_\ell(1+\epsilon_5)$ with $|\epsilon_5| \leq u$ but $|a_\ell| \leq u.|a_h|$ and $|b_\ell| \leq 2^{-\beta_o-\beta_u}.|b_h| \Rightarrow$

$$|t_{10}| \leq u.2^{-\beta_o-\beta_u}.|a_h.b_h|.(1+u)$$

$$|t_{10}| \leq u.(1+u).2^{-\beta_o-\beta_u}.\frac{1}{u+1}.|r_h|$$

$$|t_{10}| \leq u.2^{-\beta_o-\beta_u}.|r_h|$$

- $t_{11} + t_{12} = Add222(t_2, t_3, t_4, t_5)$ according to $Add222 \Rightarrow t_{11} + t_{12} = (t_2 + t_3 + t_4 + t_5).(1+\epsilon_4)$ with $|\epsilon_4| \leq 6.u^2$ and $|t_{12}| \leq 6.u.|t_{11}| \Rightarrow$

$$|t_{11} + t_{12}| \geq (1 - 6.u).|t_{11}|$$

$\Rightarrow$

$$(1-6.u).|t_{11}| \leq (|t_2| + |t_3| + |t_4| + |t_5|).(1+6.u^2)$$

$$|t_{11}| \leq \frac{1+6.u^2}{1-6.u}(|t_2| + |t_3| + |t_4| + |t_5|)$$

$$|t_{11}| \leq \frac{1+6.u^2}{1-6.u}.(2^{-\beta_o}.\frac{1}{1-u^2}.|r_h| + 2^{-\beta_o}.\frac{u}{1-u^2}.|r_h| + 2^{-\beta_o-\beta_u}.\frac{1}{1-u^2}.|r_h|$$

$$+2.2^{-\beta_o-\beta_u}.\frac{u}{1-u^2}.|r_h|)$$

$$|t_{11}| \leq \frac{(1+6.u^2).(u+1)}{(1-6.u).(1-u^2)}.(2^{-\beta_o} + 2^{-\beta_o-\beta_u}).|r_h|$$

$$|t_{12}| \leq \frac{(1+6.u^2).(u^2+u)}{(1-6.u).(1-u^2)}.(2^{-\beta_o} + 2^{-\beta_o-\beta_u}).|r_h|$$

- $t_{13} + t_{14} = Add222(t_6, t_7, t_8, t_9)$ according to $Add222 \Rightarrow t_{13} + t_{14} = (t_6 + t_7 + t_8 + t_9).(1+\epsilon_3)$with $|\epsilon_3| \leq 6.u^2$ and $|t_{14}| \leq 6.u.|t_{13}|$
We do the same calculations as before.
$\Rightarrow$

$$|t_{13}| \leq \frac{1}{1-6.u}.(|t_6| + |t_7| + |t_8| + |t_9|).(1+6.u^2)$$

$$|t_{13}| \leq \frac{1+6.u^2}{1-6.u}.(\frac{u}{1-u^2}.|r_h| + \frac{u^2}{1-u^2}.|r_h| + 2^{-\beta_o}.\frac{u}{1-u^2}.|r_h| + 2^{-\beta_o}.\frac{u^2}{1-u^2}.|r_h|).$$

$$|t_{13}| \leq \frac{(1+6.u^2).(u^2+u).(u+1)^2}{(1-6.u).(1-u^2)}.(1+2^{-\beta_o}).|r_h|.$$

$$|t_{14}| \le \frac{(1+6.u^2).(u^3+u^2).(u+1)^2}{(1-6.u).(1-u^2)}.(1+2^{-\beta_o}).|r_h|.$$

we do the same calculation for t_{15} and t_{16}, after the calculations then the simplifications, we have:

$$|t_{15}| \le \frac{(1+6.u^2).(u+1)^2}{(1-6.u).(1-u^2)}.(u.(u+1)^2+(u^3+2.u^2+u+1).2^{-\beta_o}+2^{-\beta_o-\beta_u}).|r_h|$$

$$|t_{16}| \le \frac{u.(1+6.u^2).(u+1)^2}{(1-6.u).(1-u^2)}.(u.(u+1)^2+(u^3+2.u^2+u+1).2^{-\beta_o}+2^{-\beta_o-\beta_u}).|r_h|$$

- $t_{17}+t_{18} = Add112(t_1, t_{10})$ according to $Add112 \Rightarrow t_{17}+t_{18} = t_1+t_{10}$ and $|t_{18}| \le u.|t_{17}|$, as done before $\Rightarrow$

$$|t_{17}| \le |\frac{1}{1-u}|.(|t_1|+|t_{10}|)$$

$$|t_{17}| \le |\frac{1}{1-u}|.(u.|r_h|+u.2^{-\beta_o-\beta_u}.|r_h|)$$

$$|t_{17}| \le |\frac{u}{1-u}|.(|r_h|+2^{-\beta_o-\beta_u}.|r_h|)$$

$$|t_{18}| \le |\frac{u^2}{1-u}|.(|r_h|+2^{-\beta_o-\beta_u}.|r_h|)$$

- $r_m + r_\ell = Add222(t_{17}, t_{18}, t_{15}, t_{16})$ according to $Add222 \Rightarrow r_m + r_\ell = (t_{17}+t_{18}+t_{15}+t_{16})(1+\epsilon_1)$ with $|\epsilon_1| \le 6.u^2$ and $|r_\ell| \le 6.u.|r_m|$, after calculation :

$$|r_m| \le \frac{1}{1-6.u}.(|t_{17}|+|t_{18}|+|t_{15}|+|t_{16}|).(1+6.u^2)$$

after we calculate:

$$|r_m| \le (2.u+2^{-\beta_o}+2^{-\beta_o-\beta_u}).|r_h|$$

and

$$|r_\ell| \le u.(2.u+2^{-\beta_o}+2^{-\beta_o-\beta_u}).|r_h|$$

Theorem 11 (Relative error algorithm *Mul233*) *Let (a_h, a_ℓ) a **double-double** number and (b_h, b_m, b_ℓ) a **triple-double** number are the arguments of the function $Mul233$.*
So:

$$r_h + r_m + r_\ell = (a_h + a_\ell).(b_h + b_m + b_\ell)).(1+\epsilon)$$

with $|\epsilon| \le \frac{24.u^3+19.u^2.2^{-\beta_o}+20.u^2.2^{-\beta_o-\beta_u}}{1-(u+(1+u).2^{-\beta_o}+(1+u).2^{-\beta_o-\beta_u})} \le 48.u^3+19.u^2.2^{-\beta_o+1}+5.u^2.2^{-\beta_o-\beta_u+3}$

PROOF (10PT) According to the conditions of $Mul233$, we begin to calculate $r_m + r_\ell$:
$r_m + r_\ell = Add222(t_{17}, t_{18}, t_{15}, t_{16})$ According to $Add222 \Rightarrow r_m + r_\ell = (t_{17} + t_{18} + t_{15} + t_{16}).(1 + \epsilon_1)$ with $|\epsilon_1| \leq 6.u^2$ As $t_{17} + t_{18} = t_1 + t_{10}$ and $t_{15} + t_{16} = (t_{11} + t_{12} + t_{13} + t_{14}).(1 + \epsilon_2)$ with $|\epsilon_2| \leq 6.u^2$

$$r_m + r_\ell = (t_1 + t_{10} + (t_{11} + t_{12} + t_{13} + t_{14}).(1 + \epsilon_2)).(1 + \epsilon_1)$$

As $t_{13} + t_{14} = (t_6 + t_7 + t_8 + t_9)(1 + \epsilon_3)$ and $t_{11} + t_{12} = (t_2 + t_3 + t_4 + t_5)(1 + \epsilon_4)$ with $|\epsilon_2| \leq 6.u^2$ and $|\epsilon_3| \leq 6.u^2 \Rightarrow$

$$r_m + r_\ell = (t_1 + t_{10} + ((t_2 + t_3 + t_4 + t_5)(1 + \epsilon_4) + (t_6 + t_7 + t_8 + t_9)(1 + \epsilon_3)).(1 + \epsilon_2)).(1 + \epsilon_1)$$

As $t_2 + t_3 = a_h.b_m$, $t_4 + t_5 = a_h.b_\ell$, $t_6 + t_7 = a_\ell.b_h$ and $t_8 + t_9 = a_\ell.b_m \Rightarrow$

$$r_m + r_\ell = (t_1 + t_{10} + ((a_h.b_m + a_h.b_\ell)(1 + \epsilon_4) + (a_\ell.b_h + a_\ell.b_m)(1 + \epsilon_3)).(1 + \epsilon_2)).(1 + \epsilon_1)$$

As $t_{10} = a_\ell.b_\ell.(1 + \epsilon_5)$ with $|\epsilon_5| \leq u \Rightarrow$

$$r_m + r_\ell = (t_1 + a_\ell.b_\ell.(1 + \epsilon_5) + ((a_h.b_m + a_h.b_\ell)(1 + \epsilon_4) + (a_\ell.b_h + a_\ell.b_m)(1 + \epsilon_3)).(1 + \epsilon_2)).(1 + \epsilon_1)$$

As $r_h + t_1 = a_h.b_h \Rightarrow$

$$r_h + r_m + r_\ell = (a_h + a_\ell).(b_h + b_m + b_\ell) + \delta$$

with $\delta = t_1.\epsilon_1 + a_\ell.b_\ell.(\epsilon_1 + \epsilon_5 + \epsilon_1.\epsilon_5) + (a_h.b_m + a_h.b_\ell).(\epsilon_1 + \epsilon_2\epsilon_4 + \epsilon_1.\epsilon_2 + \epsilon_1.\epsilon_4 + \epsilon_2.\epsilon_4 + \epsilon_1.\epsilon_2.\epsilon_4) + (a_\ell.b_h + a_\ell.b_m).(\epsilon_1 + \epsilon_3\epsilon_4 + \epsilon_1.\epsilon_3 + \epsilon_1.\epsilon_4 + \epsilon_3.\epsilon_4 + \epsilon_1.\epsilon_3.\epsilon_4)$.
We seek the upper bound of $|\delta|$ in function of $|a_h.b_h|$:

$$|\delta| \leq |t_1.\epsilon_1| + |a_\ell.b_\ell.(\epsilon_1 + \epsilon_5 + \epsilon_1.\epsilon_5)| + (|a_h.b_m| + |a_h.b_\ell|).|\epsilon_1 + \epsilon_2\epsilon_4 + \epsilon_1.\epsilon_2 + \epsilon_1.\epsilon_4 + \epsilon_2.\epsilon_4 + \epsilon_1.\epsilon_2.\epsilon_4|$$
$$+ (|a_\ell.b_h| + |a_\ell.b_m|).|\epsilon_1 + \epsilon_3\epsilon_4 + \epsilon_1.\epsilon_3 + \epsilon_1.\epsilon_4 + \epsilon_3.\epsilon_4 + \epsilon_1.\epsilon_3.\epsilon_4|$$

$$|\delta| \leq 6.u^2.|t_1| + (6.u^3 + 6.u^2 + u).|a_\ell.b_\ell.| + (216.u^6 + 108.u^4 + 18.u^2).(|a_h.b_m| + |a_h.b_\ell| + |a_\ell.b_h| + |a_\ell.b$$

We seek the upper bound of $|a_h.b_m|$, $|a_h.b_\ell|$, $|a_\ell.b_h|$, $|a_\ell.b_m|$ and $|a_\ell.b_\ell|$ in function of $|a_h.b_h|$.
We have:

$$|a_h.b_m| \leq 2^{-\beta_o}.|a_h.b_h|$$
$$|a_h.b_\ell| \leq 2^{-\beta_o-\beta_u}.|a_h.b_h|$$
$$|a_\ell.b_h| \leq u.|a_h.b_h|$$
$$|a_\ell.b_m| \leq u.2^{-\beta_o}.|a_h.b_h|$$
$$|a_\ell.b_\ell| \leq u.2^{-\beta_o-\beta_u}.|a_h.b_h|$$

We search the upper bound of $(|a_h.b_m|+|a_h.b_\ell|+|a_\ell.b_h|+|a_\ell.b_m|)$ in function of $|a_h.b_h|$

$|a_h.b_m|+|a_h.b_\ell|+|a_\ell.b_h|+|a_\ell.b_m| \leq 2^{-\beta_o}.|a_h.b_h|+2^{-\beta_o-\beta_u}.|a_h.b_h|+u.|a_h.b_h|+u.2^{-\beta_o}.|a_h.b_h|$

$$|a_h.b_m| + |a_h.b_\ell| + |a_\ell.b_h| + |a_\ell.b_m| \leq (u + (1+u).2^{-\beta_o} + 2^{-\beta_o-\beta_u}).|a_h.b_h|$$

as $|a_\ell.b_\ell| \leq u.2^{-\beta_o-\beta_u}.|a_h.b_h|$
According to $Mul112$, we have :

$$|t_1| \leq u.|r_h| \; and \; r_h + t_1 = a_h.b_h$$

We begin to calculate the lower bound of $|r_h + t_1|$ compared to $|r_h|$.
As $|t_1| \leq u.|r_h|$, we have :

$$|r_h + t_1| \geq (1-u).|r_h|$$

$\Rightarrow$

$$(1-u).|r_h| \leq |a_h.b_h|$$

$$|r_h| \leq \frac{1}{1-u}.|a_h.b_h|$$

$\Rightarrow$

$$|t_1| \leq \frac{u}{1-u}.|a_h.b_h|$$

$|\delta| \leq (\frac{6u^3}{1-u}.+(6.u^3+6.u^2+u).u.2^{-\beta_o-\beta_u}+(216.u^6+108.u^4+18.u^2).(u+(1+u).2^{-\beta_o}+2^{-\beta_o-\beta_u})).|a_h.b_h|$

we simplify with the upper bounds of each coefficient.

$$|\delta| \leq (24.u^3 + 19.u^2.2^{-\beta_o} + 20.u^2.2^{-\beta_o-\beta_u}).|a_h.b_h|$$

Now, we seek the lower bound of $|a_h.b_h|$ in function of $|(a_h+a_\ell).(b_h+b_m+b_\ell)|$,
as $|a_h.b_m| + |a_h.b_\ell| + |a_\ell.b_h| + |a_\ell.b_m| \leq (u + (1+u).2^{-\beta_o} + 2^{-\beta_o-\beta_u}).|a_h.b_h|$
and $|a_\ell.b_\ell| \leq u.2^{-\beta_o-\beta_u}.|a_h.b_h| \Rightarrow$

$|a_h.b_m+a_h.b_\ell+a_\ell.b_h+a_\ell.b_m+a_\ell.b_\ell| \leq |a_h.b_m|+|a_h.b_\ell|+|a_\ell.b_h|+|a_\ell.b_m|+|a_\ell.b_\ell|$

$|a_h.b_m+a_h.b_\ell+a_\ell.b_h+a_\ell.b_m+a_\ell.b_\ell| \leq (u+(1+u).2^{-\beta_o}+(1+u).2^{-\beta_o-\beta_u}).|a_h.b_h|$

So we have:

$|(a_h + a_\ell).(b_h + b_m + b_\ell)| \geq (1 - (u + (1+u).2^{-\beta_o} + (1+u).2^{-\beta_o-\beta_u})).|a_h.b_h|$

$|a_h.b_h| \leq \frac{1}{1-(u+(1+u).2^{-\beta_o}+(1+u).2^{-\beta_o-\beta_u})}.|(a_h+a_\ell).(b_h+b_m+b_\ell)|$

at the end:

$$|\delta| \leq \frac{24.u^3 + 19.u^2.2^{-\beta_o} + 20.u^2.2^{-\beta_o-\beta_u}}{1-(u+(1+u).2^{-\beta_o}+(1+u).2^{-\beta_o-\beta_u})}.|(a_h+a_\ell).(b_h+b_m+b_\ell)|$$

$\Rightarrow |\epsilon| \leq \frac{24.u^3+19.u^2.2^{-\beta_o}+20.u^2.2^{-\beta_o-\beta_u}}{1-(u+(1+u).2^{-\beta_o}+(1+u).2^{-\beta_o-\beta_u})}$

As $\beta_o \geq 2$ and $\beta_u \geq 1 \Rightarrow 1-(u+(1+u).2^{-\beta_o}+(1+u).2^{-\beta_o-\beta_u}) \geq \frac{1}{2}$ $\Rightarrow$
$|\epsilon| \leq 48.u^3 + 19.u^2.2^{-\beta_o+1} + 5.u^2.2^{-\beta_o-\beta_u+3}$

4.2.3 Mul333

See algorithm 11

Lemma 11 (Mul333) *Let* (a_h, a_m, a_ℓ) *and* (b_h, b_m, b_ℓ) *are* ***triple-Double*** *number,* r_h*,* r_m *and* r_ℓ *result of* $Mul333(a_h, a_m, a_\ell, b_h, b_m, b_\ell)$ *for the* 4 *modes of rounding, considering that there is no* ***overflow*** *so:* $|r_\ell| \leq 6.u.|r_m|$, $|r_m| \leq .(u + 2^{-\alpha_o} + 2^{-\beta_o} + 2^{-\alpha_o-\beta_o+1}).|r_h|$ *and* $|r_\ell| \leq u.(u + 2^{-\alpha_o} + 2^{-\beta_o} + 2^{-\alpha_o-\beta_o+1}).|r_h|$

Proof According to $Mul333$, we seek for each $|t_i|$ with $1 \leq i \leq 22$ their upper bound in function of $|r_h|$
We begin to search the upper bound of $|a_h.b_m|$, $|a_h.b_\ell|$, $|a_m.b_h|$,$|a_m.b_m|$, $|a_m.b_\ell|$, $|a_\ell.b_h|$, $|a_\ell.b_m|$ and $|a_\ell.b_\ell|$ in function of $|a_h.b_h$.
Thanks to the conditions of $Add333$, we have:

$$|a_h.b_m| \leq 2^{-\beta_o}.|a_h.b_h|$$

$$|a_h.b_\ell| \leq 2^{-\beta_o-\beta_u}.|a_h.b_h|$$

$$|a_m.b_h| \leq 2^{-\alpha_o}.|a_h.b_h|$$

$$|a_m.b_m| \leq 2^{-\alpha_o-\beta_o}.|a_h.b_h|$$

$$|a_m.b_\ell| \leq 2^{-\alpha_o-\beta_o-\beta_u}.|a_h.b_h|$$

$$|a_\ell.b_h| \leq 2^{-\alpha_o-\alpha_u}.|a_h.b_h|$$

$$|a_\ell.b_m| \leq 2^{-\alpha_o-\alpha_u-\beta_o}.|a_h.b_h|$$

$$|a_\ell.b_\ell| \leq 2^{-\alpha_o-\alpha_u-\beta_o-\beta_u}.|a_h.b_h|$$

We have:

- $r_h + t_1 = Mul112(a_h, b_h)$ according to $Mul112 \Rightarrow r_h + t_1 = a_h.b_h$ and $|t_1| \leq u.|r_h| \Rightarrow$
$$|a_h.b_h| \leq (1+u).|r_h|$$

- $t_2 + t_3 = Mul112(a_h, b_m)$ according to $Mul112 \Rightarrow t_2 + t_3 = a_h.b_m$ and $|t_3| \leq u.|t_2| \Rightarrow$
$$|t_2 + t_3| \geq (1-u).|t_2|$$
$$(1-u).|t_2| \leq |a_h.b_m|$$
$$|t_2| \leq \frac{1}{1-u}.|a_h.b_m|$$
$$|t_2| \leq \frac{1}{1-u}.2^{-\beta_o}.|a_h.b_h|$$
$$|t_2| \leq \frac{1}{1-u}.2^{-\beta_o}.(1+u).|r_h|$$
$$|t_2| \leq \frac{1+u}{1-u}.2^{-\beta_o}.|r_h|$$
$\Rightarrow$
$$|t_3| \leq \frac{u^2+u}{1-u}.2^{-\beta_o}.|r_h|$$
For $|t_i|$ with $4 \leq i \leq 7$,we don't repeat the calculations:
$$|t_4| \leq \frac{1+u}{1-u}.2^{-\alpha_o}.|r_h|$$
$$|t_5| \leq \frac{u^2+u}{1-u}.2^{-\alpha_o}.|r_h|$$
$$|t_6| \leq \frac{1+u}{1-u}.2^{-\alpha_o-\beta_o}.|r_h|$$
$$|t_7| \leq \frac{u^2+u}{1-u}.2^{-\alpha_o-\beta_o}.|r_h|$$

- $t_8 = \circ(a_h.b_\ell)$ according to the collary 5 $\Rightarrow t_8 = a_h.b_\ell.(1+\epsilon_{11})$ with $|\epsilon_{11}| \leq u \Rightarrow$
$$|t_8| \leq |a_h.b_\ell|.(1+u)$$
$$|t_8| \leq 2^{-\beta_o-\beta_u}.|a_h.b_h|.(1+u)$$
$$|t_8| \leq 2^{-\beta_o-\beta_u}.(1+u).|r_h|.(1+u)$$
$$|t_8| \leq (1+u)^2.2^{-\beta_o-\beta_u}.|r_h|$$

We do the same operations for $|t_i|$ with $9 \leq i \leq 11$:

$$|t_9| \leq (1+u)^2.2^{-\alpha_o-\alpha_u}.|r_h|$$

$$|t_{10}| \leq (1+u)^2.2^{-\alpha_o-\beta_o-\beta_u}.|r_h|$$

$$|t_{11}| \leq (1+u)^2.2^{-\alpha_o-\alpha_u-\beta_o}.|r_h|$$

- $t_{12} = \circ(t_8 + t_9)$ according to collary 5: $t_{12} = (t_8 + t_9).(1 + \epsilon_7)$ with $|\epsilon_7| \leq u \Rightarrow$

$$|t_{12}| \leq |t_8 + t_9|.(1+u)$$

$$|t_{12}| \leq ((1+u)^2.2^{-\beta_o-\beta_u} + (1+u)^2.2^{-\alpha_o-\alpha_u}).|r_h|.(1+u)$$

$$|t_{12}| \leq ((1+u)^3.2^{-min(\beta_o+\beta_u,\alpha_o+\alpha_u)} + .|r_h|$$

similarly:

$$|t_{13}| \leq ((1+u)^3.2^{-min(\alpha_o+\beta_o+\beta_u,\alpha_o+\alpha_u+\beta_o)}|r_h|$$

- $t_{14} + t_{15} = Add112(t_1, t_6)$ according to $Add112 \Rightarrow t_{14} + t_{15} = t_1 + t_6$ and $|t_{15}| \leq u.|t_{14}| \Rightarrow$

$$|t_{14}| \leq \frac{1}{1-u}.|t_1 + t_6|$$

$$|t_{14}| \leq \frac{1}{1-u}.(u + \frac{1+u}{1-u}.2^{-\alpha_o-\beta_o}).|r_h|$$

$$|t_{14}| \leq \frac{1}{(1-u)^2}.(u - u^2 + (1+u).2^{-\alpha_o-\beta_o}).|r_h|$$

$\Rightarrow$

$$|t_{15}| \leq \frac{u}{(1-u)^2}.(u - u^2 + (1+u).2^{-\alpha_o-\beta_o}).|r_h|$$

- We calculate $|t_i|$ with $16 \leq i \leq 18$ as we did previously with the calculation of $|t_{12}|$

$$|t_{16}| \leq \frac{1+u}{(1-u)^2}.(u^2 - u^3 + (-u^3 + u).2^{-\alpha_o-\beta_o}).|r_h|$$

$$|t_{17}| \leq (1+u)^4.2^{-min(\alpha_o+\beta_o+\beta_u,\alpha_o+\alpha_u+\beta_o)}.|r_h|.$$

$$|t_{18}| \leq (\frac{1+u}{(1-u)^2}.(u^2 - u^3 + (-u^3 + u).2^{-\alpha_o-\beta_o})$$
$$+(1+u)^4.2^{-min(\alpha_o+\beta_o+\beta_u,\alpha_o+\alpha_u+\beta_o)}).(1+u).|r_h|$$

As $\alpha_o + \beta_o < \alpha_o + \beta_o + \beta_u$ and $\alpha_o + \beta_o < \alpha_o + \beta_o + \alpha_u \Rightarrow$

$$|t_{18}| \leq (\frac{(1+u)^2}{(1-u)^2}.(u^2 - u^3 + ((-u^3 + u) + (1+u)^3.(1-u^2)).2^{-\alpha_o-\beta_o})).|r_h|$$

- $t_{19}+t_{20} = Add112(t_{14}, t_{18})$ according to $Add112 \Rightarrow t_{19}+t_{20} = t_{14}+t_{18}$ and $|t_{20}| \leq u.|t_{19}| \Rightarrow$, we simplify:

$$|t_{19}| \leq (u + 2^{-\alpha_o-\beta_o+1}).|r_h|$$

$$|t_{20}| \leq u.(u + 2^{-\alpha_o-\beta_o+1}).|r_h|$$

- $t_{21} + t_{22} = Add222(t_2, t_3, t_4, t_5)$ according to $Add222 \Rightarrow t_{21} + t_{22} = (t_2 + t_3 + t_4 + t_5).(1 + \epsilon_2)$ with $|\epsilon_2| \leq 6.u^2$ and $|t_{22}| \leq 6.u.|t_{21}| \Rightarrow$

$$|t_{21}| \leq \frac{(u+1)^2.(1+6u^2)}{(1-u)^2}.(2^{-\alpha_o} + 2^{-\beta_o}).|r_h|$$

$$|t_{22}| \leq \frac{u.(u+1)^2.(1+6u^2)}{(1-u)^2}.(2^{-\alpha_o} + 2^{-\beta_o}).|r_h|$$

- $r_m+r_\ell = Add222(t_{21}, t_{22}, t_{19}, t_{20})$,according to $Add222 \Rightarrow |r_\ell| \leq 6.u.|r_m|$. After the calculations, we have:

$$|r_m| \leq \frac{1+6.u^2}{1-u^2}.(\frac{(u+1)^3.(1+6u^2)}{(1-u)^2}.(2^{-\alpha_o}+2^{-\beta_o})+(u+1).(u+2^{-\alpha_o-\beta_o+1})).|r_h|$$

after simplification:

$$|r_m| \leq .(u + 2^{-\alpha_o} + 2^{-\beta_o} + 2^{-\alpha_o-\beta_o+1}).|r_h|$$

and

$$|r_\ell| \leq u.(u + 2^{-\alpha_o} + 2^{-\beta_o} + 2^{-\alpha_o-\beta_o+1}).|r_h|$$

Theorem 12 (Relative error algorithm $\boldsymbol{Mul333}$) *Let (a_h, a_m, a_ℓ) and (b_h, b_m, b_ℓ) are **triple-double** numbers and the arguments of the function $Mul233$.*
So:

$$r_h + r_m + r_\ell = (a_h + a_m + a_\ell).(b_h + b_m + b_\ell)).(1 + \epsilon)$$

with

$$|\epsilon| \leq 32.u^3+3.u.(2^{-\beta_o+2}+2^{-\alpha_o+2})+2^{-\beta_o-\beta_u+3}+2^{-\alpha_o-\alpha_u+3}+2^{-\alpha_o-\alpha_u-\beta_u+3}+2^{-\alpha_o-\beta_o-\beta_u+3}$$

Proof According to $Mul333$, we have:

- $r_m + r_\ell = Add222(t_{21}, t_{22}, t_{19}, t_{20})$ based on $Add222 \Rightarrow$ $r_m + r_\ell = (t_{21} + t_{22} + t_{19} + t_{20})(1 + \epsilon_1)$ with $|\epsilon_1| \leq 6.u^2$

- $t_{21} + t_{22} = (t_2 + t_3 + t_4 + t_5).(1 + \epsilon_2)$ with $|\epsilon_2| \leq 6.u^2$
- $t_2 + t_3 = a_h.b_m$ and $t_4 + t_5 = a_m.b_m \Rightarrow$

$$t_{21} + t_{22} = (a_h.b_m + a_m.b_m).(1 + \epsilon_2)$$
$$r_m + r_\ell = ((a_h.b_m + a_m.b_m).(1 + \epsilon_2) + t_{19} + t_{20})(1 + \epsilon_1)$$

- $t_{19} + t_{20} = t_{14} + t_{18}$ thanks to $Add112$
- $t_{18} = (t_{16} + t_{17})(1 + \epsilon_3)$ with $|\epsilon_3| \leq u$ thanks to the collary 5
- $t_{17} = (t_{12} + t_{13})(1 + \epsilon_4)$ with $|\epsilon_4| \leq u$ thanks to the collary 5
- $t_{12} = (t_8 + t_9)(1 + \epsilon_7)$ with $|\epsilon_7| \leq u$ thanks to the collary 5
- $t_8 = (a_h.b_\ell).(1 + \epsilon_{11})$ with $|\epsilon_{11}| \leq u$ and $t_9 = (a_\ell.b_h).(1 + \epsilon_{10})$ with $|\epsilon_{10}| \leq u$ thanks to the collary 5
-

$$t_{12} = ((a_h.b_\ell).(1 + \epsilon_{11}) + (a_\ell.b_h).(1 + \epsilon_{10}))(1 + \epsilon_7)$$
$$t_{17} = (((a_h.b_\ell).(1 + \epsilon_{11}) + (a_\ell.b_h).(1 + \epsilon_{10}))(1 + \epsilon_7) + t_{13})(1 + \epsilon_4)$$

- $t_{13} = (t_{10} + t_{11}).(1 + \epsilon_6)$ with $|\epsilon_6| \leq u$
- $t_{10} = a_m.b_\ell.(1 + \epsilon_9)$ and $t_{11} = a_\ell.b_m.(1 + \epsilon_8)$ with $|\epsilon_8| \leq u$ and $|\epsilon_9| \leq u$

$$t_{13} = (a_m.b_\ell.(1 + \epsilon_9) + a_\ell.b_m.(1 + \epsilon_8)).(1 + \epsilon_6)$$

$t_{17} = (((a_h.b_\ell).(1+\epsilon_{11})+(a_\ell.b_h).(1+\epsilon_{10}))(1+\epsilon_7)+(a_m.b_\ell.(1+\epsilon_9)+a_\ell.b_m.(1+\epsilon_8)).(1+\epsilon_6))(1$

At the end of the calculation of t_{17}, we have:

$$\begin{aligned} t_{17} = {} & a_h.b_\ell.(1 + \epsilon_4 + \epsilon_7 + \epsilon_{11} + \epsilon_4.\epsilon_7 + \epsilon_4.\epsilon_{11} + \epsilon_7.\epsilon_{11} + \epsilon_4.\epsilon_7.\epsilon_{11}) \\ & + a_\ell.b_h.(1 + \epsilon_4 + \epsilon_7 + \epsilon_{10} + \epsilon_4.\epsilon_7 + \epsilon_4.\epsilon_{10} + \epsilon_7.\epsilon_{10} + \epsilon_4.\epsilon_7.\epsilon_{10}) \\ & + a_m.b_\ell.(1 + \epsilon_4 + \epsilon_6 + \epsilon_9 + \epsilon_4.\epsilon_6 + \epsilon_4.\epsilon_9 + \epsilon_6.\epsilon_9 + \epsilon_4.\epsilon_6.\epsilon_9) \\ & + a_m.b_\ell.(1 + \epsilon_4 + \epsilon_6 + \epsilon_8 + \epsilon_4.\epsilon_6 + \epsilon_4.\epsilon_8 + \epsilon_6.\epsilon_8 + \epsilon_4.\epsilon_6.\epsilon_8) \end{aligned}$$

- $t_{16} = (t_7 + t_{15}).(1 + \epsilon_5)$ with $|\epsilon_5| \leq u$

$$t_{18} = ((t_7 + t_{15}).(1 + \epsilon_5) + t_{17})(1 + \epsilon_3)$$
$$t_{19} + t_{20} = t_{14} + ((t_7 + t_{15}).(1 + \epsilon_5) + t_{17})(1 + \epsilon_3)$$
$$t_{19} + t_{20} = t_{14} + (t_7 + t_{15}).(1 + \epsilon_3 + \epsilon_5 + \epsilon_3.\epsilon_5) + t_{17}.(1 + \epsilon_3)$$

$$t_{19} + t_{20} = t_{14} + t_{15} + t_7 + (t_7 + t_{15}).(\epsilon_3 + \epsilon_5 + \epsilon_3.\epsilon_5) + t_{17}.(1 + \epsilon_3)$$

As $t_{14} + t_{15} = t_1 + t_6$

$$t_{19} + t_{20} = t_1 + t_6 + t_7 + (t_7 + t_{15}).(\epsilon_3 + \epsilon_5 + \epsilon_3.\epsilon_5) + t_{17}.(1 + \epsilon_3)$$

As $t_6 + t_7 = a_m.b_m$

$$t_{19} + t_{20} = t_1 + a_m.b_m + (t_7 + t_{15}).(\epsilon_3 + \epsilon_5 + \epsilon_3.\epsilon_5) + t_{17}.(1 + \epsilon_3)$$

$r_m+r_\ell = ((a_h.b_m+a_m.b_m).(1+\epsilon_2)+t_1+a_m.b_m+(t_7+t_{15}).(\epsilon_3+\epsilon_5+\epsilon_3.\epsilon_5)+t_{17}.(1+\epsilon_3))(1+\epsilon_1)$

$r_m+r_\ell = ((a_h.b_m+a_m.b_m).(1+\epsilon_2+\epsilon_3+\epsilon_2.\epsilon_3)+t_1.(1+\epsilon_1)+t_{17}.(1+\epsilon_1+\epsilon_3+\epsilon_1.\epsilon_3)$

$\quad + a_m.b_m.(1 + \epsilon_1) + (t_7 + t_{15}).(\epsilon_3 + \epsilon_5 + \epsilon_1.\epsilon_5 + \epsilon_3.\epsilon_5 + \epsilon_1.\epsilon_3.\epsilon_5)$

We add $r_h \Rightarrow$:

$$r_h+r_m+r_\ell = r_h+t_1+(a_h.b_m+a_m.b_m).(1+\epsilon_2+\epsilon_3+\epsilon_2.\epsilon_3)+t_1.\epsilon_1+t_{17}.(1+\epsilon_1+\epsilon_3+\epsilon_1.\epsilon_3)$$

$$+ a_m.b_m.(1 + \epsilon_1) + (t_7 + t_{15}).(\epsilon_3 + \epsilon_5 + \epsilon_1.\epsilon_5 + \epsilon_3.\epsilon_5 + \epsilon_1.\epsilon_3.\epsilon_5)$$

$$r_h + r_m + r_\ell = (a_h + a_m).(b_h, b_m, b_\ell) + a_\ell.(b_h, b_m) + \delta$$

with $\delta = t_1.\epsilon_1 + (a_h.b_m + a_m.b_h).(\epsilon_2 + \epsilon_3 + \epsilon_2.\epsilon_3) + a_m.b_m.\epsilon_1 + (t_7 + t_{15}).(\epsilon_3 + \epsilon_5 + \epsilon_1.\epsilon_3 + \epsilon_1.\epsilon_5 + \epsilon_3.\epsilon_5 + \epsilon_1.\epsilon_3.\epsilon_5) + t_{17}.(1 + \epsilon_1 + \epsilon_3 + \epsilon_1.\epsilon_3) - (a_h.b_\ell + a_\ell.b_h + a_m.b_\ell + a_\ell.b_m)$.
We search the upper bound of $|\delta|$ in function of $|a_h.b_h|$.

$|\delta| \leq |t_1.\epsilon_1| + |(a_h.b_m + a_m.b_h).(\epsilon_2 + \epsilon_3 + \epsilon_2.\epsilon_3)| + |a_m.b_m.\epsilon_1|$

$+|(t_7+t_{15}).(\epsilon_3+\epsilon_5+\epsilon_1.\epsilon_3+\epsilon_1.\epsilon_5+\epsilon_3.\epsilon_5+\epsilon_1.\epsilon_3.\epsilon_5)|+|t_{17}.(1+\epsilon_1+\epsilon_3+\epsilon_1.\epsilon_3)-(a_h.b_\ell+a_\ell.b_h+a_m.b_\ell+a_\ell.b$

.

We begin to calculate $t_{17}.(1 + \epsilon_1 + \epsilon_3 + \epsilon_1.\epsilon_3) - (a_h.b_\ell + a_\ell.b_h + a_m.b_\ell + a_\ell.b_m)$:

$$t_{17}.(1 + \epsilon_1 + \epsilon_3 + \epsilon_1.\epsilon_3) - (a_h.b_\ell + a_\ell.b_h + a_m.b_\ell + a_\ell.b_m) =$$

$a_h.b_\ell.((1+\epsilon_4+\epsilon_7+\epsilon_{11}+\epsilon_4.\epsilon_7+\epsilon_4.\epsilon_{11}+\epsilon_7.\epsilon_{11}+\epsilon_4.\epsilon_7.\epsilon_{11}).(1+\epsilon_1+\epsilon_3+\epsilon_1.\epsilon_3)-1)$

$+ a_\ell.b_h.((1+\epsilon_4+\epsilon_7+\epsilon_{10}+\epsilon_4.\epsilon_7+\epsilon_4.\epsilon_{10}+\epsilon_7.\epsilon_{10}+\epsilon_4.\epsilon_7.\epsilon_{10}).(1+\epsilon_1+\epsilon_3+\epsilon_1.\epsilon_3)-1)$

$+ a_m.b_\ell.((1+\epsilon_4+\epsilon_6+\epsilon_9+\epsilon_4.\epsilon_6+\epsilon_4.\epsilon_9+\epsilon_6.\epsilon_9+\epsilon_4.\epsilon_6.\epsilon_9).(1+\epsilon_1+\epsilon_3+\epsilon_1.\epsilon_3)-1)$

$+ a_m.b_\ell.((1+\epsilon_4+\epsilon_6+\epsilon_8+\epsilon_4.\epsilon_6+\epsilon_4.\epsilon_8+\epsilon_6.\epsilon_8+\epsilon_4.\epsilon_6.\epsilon_8).(1+\epsilon_1+\epsilon_3+\epsilon_1.\epsilon_3)-1)$

As $|\epsilon_i| \leq 6.u^2$ with $i \in [1, 2]$ and $|\epsilon_j| \leq u$ with $3 \leq j \leq 11$
We have so $|t_{17}.(1 + \epsilon_1 + \epsilon_3 + \epsilon_1.\epsilon_3) - (a_h.b_\ell + a_\ell.b_h + a_m.b_\ell + a_\ell.b_m)|$
it's equal to

$|a_h.b_\ell+a_\ell.b_h+a_m.b_\ell+a_\ell.b_m|.(6.u^6+24.u^5+37.u^4+23.u^3+9.u^2+6.u+1)$
$\Rightarrow$

$$|\delta| \leq 6.u^2.|t_1| + (6.u^3+6.u^2+u).|a_h.b_m+a_m.b_h| + 6.u^2.|a_m.b_m|$$
$$+(6.u^4+12.u^3+u^2+2.u).|t_7+t_{15}|$$
$$+(6.u^6+24.u^5+37.u^4+23.u^3+9.u^2+6.u+1).|a_h.b_\ell+a_\ell.b_h+a_m.b_\ell+a_\ell.b_m|$$

. According to the conditions of $Mul333$:

$$|a_h.b_m| \leq 2^{-\beta_o}.|a_h.b_h|$$
$$|a_h.b_\ell| \leq 2^{-\beta_o-\beta_u}.|a_h.b_h|$$
$$|a_m.b_h| \leq 2^{-\alpha_o}.|a_h.b_h|$$
$$|a_m.b_m| \leq 2^{-\alpha_o-\beta_o}.|a_h.b_h|$$
$$|a_m.b_\ell| \leq 2^{-\alpha_o-\beta_o-\beta_u}.|a_h.b_h|$$
$$|a_\ell.b_h| \leq 2^{-\alpha_o-\alpha_u}.|a_h.b_h|$$
$$|a_\ell.b_m| \leq 2^{-\alpha_o-\alpha_u-\beta_o}.|a_h.b_h|$$

$\Rightarrow$

$$|\delta| \leq 6.u^2.|t_1|+(6.u^3+6.u^2+u).(2^{-\beta_o}.|a_h.b_h|+2^{-\alpha_o}.|a_h.b_h|)+6.u^2.2^{-\alpha_o-\beta_o}.|a_h.b_h|$$
$$+(6.u^4+12.u^3+u^2+2.u).|t_7+t_{15}|$$
$$+(6.u^6+24.u^5+37.u^4+23.u^3+9.u^2+6.u+1)$$
$$\times\,(2^{-\beta_o-\beta_u}.|a_h.b_h|+2^{-\alpha_o-\alpha_u}.|a_h.b_h|+2^{-\alpha_o-\beta_o-\beta_u}.|a_h.b_h|+2^{-\alpha_o-\alpha_u-\beta_o}.|a_h.b_h|)$$

We seek the upper bound of $|t_1|$ in function of $|a_h.b_h|$.
As $r_h+t_1 = a_h.b_h$ thanks to $Add112 \Rightarrow |t_1| \leq u.|r_h| \Rightarrow$

$$|r_h| \leq \frac{1}{1-u}.|a_h.b_h|$$

$\Rightarrow$

$$|t_1| \leq \frac{u}{1-u}.|a_h.b_h|$$

Now, doing the same operation :

$$|t_7| \leq \frac{u}{1-u}.|a_m.b_m|$$

$$|t_7| \leq \frac{u}{1-u}.2^{-\alpha_o-\beta_o}.|a_h.b_h|$$

$$|t_{15}| \leq \frac{u}{1-u}.|t_1 + t_6|$$

$$|t_{15}| \leq \frac{u}{1-u}.(|t_1| + |t_6|)$$

We have:

$$|t_6| \leq \frac{1}{1-u}.2^{-\alpha_o-\beta_o}.|a_h.b_h|$$

$\Rightarrow$

$$|t_{15}| \leq \frac{u}{1-u}.(\frac{u}{1-u}.|a_h.b_h| + \frac{1}{1-u}.2^{-\alpha_o-\beta_o}.|a_h.b_h|)$$

$$|t_{15}| \leq \frac{u}{(1-u)^2}.(u + 2^{-\alpha_o-\beta_o}).|a_h.b_h|$$

. After simplify, we have:

$$|\delta| \leq (8.u^3 + 3.u.(2^{-\beta_o} + 2^{-\alpha_o}) + 2^{-\beta_o-\beta_u+1} + 2^{-\alpha_o-\alpha_u+1} + 2^{-\alpha_o-\alpha_u-\beta_u+1} + 2^{-\alpha_o-\beta_o-\beta_u+1}).|a_h.b_h|$$

We seek the upper bound of $|a_h.b_h|$ in function of $|a_m.(b_h, b_m, b_\ell) + a_\ell.(b_h, b_m) + a_h.(b_m, b_\ell)|$

$$|a_m.(b_h, b_m, b_\ell) + a_\ell.(b_h, b_m) + a_h.(b_m, b_\ell)| \leq$$

$$(2^{-\beta_o} + 2^{-\beta_o-\beta_u} + 2^{-\alpha_o} + 2^{-\alpha_o-\beta_o-\beta_u+1} + 2^{-\alpha_o-\alpha_u+1} + 2^{-\alpha_o-\beta_o+1} + 2^{-\alpha_o-\alpha_u-\beta_o+1}).|a_h.b_h|$$

But $1 - (2^{-\beta_o} + 2^{-\beta_o-\beta_u} + 2^{-\alpha_o} + 2^{-\alpha_o-\beta_o-\beta_u} + 2^{-\alpha_o-\alpha_u} + 2^{-\alpha_o-\beta_o} + 2^{-\alpha_o-\alpha_u-\beta_o}) \geq \frac{1}{4}$ We have :

$$|a_h.b_h| \leq 4.|a_m.(b_h, b_m, b_\ell) + a_\ell.(b_h, b_m) + a_h.(b_m, b_\ell)|$$

So we have:

$$|\delta| \leq$$

$$(32.u^3 + 3.u.(2^{-\beta_o+2} + 2^{-\alpha_o+2}) + 2^{-\beta_o-\beta_u+3} + 2^{-\alpha_o-\alpha_u+3} + 2^{-\alpha_o-\alpha_u-\beta_u+3} + 2^{-\alpha_o-\beta_o-\beta_u+3})$$

$$\times |(a_h + a_m + a_\ell).(b_h + b_m + b_\ell)|$$

so :

$$|\epsilon| \leq$$

$$32.u^3 + 3.u.(2^{-\beta_o+2} + 2^{-\alpha_o+2}) + 2^{-\beta_o-\beta_u+3} + 2^{-\alpha_o-\alpha_u+3} + 2^{-\alpha_o-\alpha_u-\beta_u+3} + 2^{-\alpha_o-\beta_o-\beta_u+3}$$

Chapter 5

log

5.1 crlogfast

See algorithm 12

In practice with the C program, we found as an error bound $err_{fast} = 2^{58.4}$. (see the C program in appendix).

5.2 crlogaccurate

See algorithm 13

In practice with the C program, we found as an error bound $err_{accurate} = 2^{98.4}$. (see appendix the C program).

5.3 crlogadvanced

See algorithm 14

5.4 log

See algorithm 15

The log function is composed of the 3 $cr\log$. We start using $crlog_{fast}$, if this function cannot find the result of the log, it returns to $cr\log_{accurate}$, it does the same thing as the previous function. The last $cr\log$ which is $cr\log_{advanced}$ will calculate all the calculations not solved by the previous $cr\log$.

The relative errors calculated for $crlog_{fast}$ and $cr\log_{accurate}$, will be used for the function to know if the calculation passes for each.

Let x be a **double** ,err_{fast} is the relative error calculated for $cr\log_{fast}$ and err_{acc} is that of $cr\log_{accurate}$.
The computation of $\log(x)$ proceeds by first computing $cr\log_{fast}(x)$ which gives us a **double-double**. We will name this **double-double** (h_1, ℓ_1).
We take $right = h_1 + err_{fast} * h_1 + \ell_1$ and $left = h_1 - err_{fast} * h_1 + \ell_1$.
If $right = left$ then we have the result of $\log(x)$ otherwise we calculate with $cr\log_{accurate}(x)$. Let (h_2, ℓ_2) be the result of $cr\log_{accurate}(x)$.
We set $right = h_2 + err_{accurate} * h_2 + \ell_2$ and $left = h_2 - err_{accurate} * h_2 + \ell_2$.
If $right = left$ then we have the result of $\log(x)$ otherwise we calculate with $cr\log_{advancedaccurate}(x)$. At the end of the log function calculation, we have the real double value of $\log(x)$.

Conclusion

. In our report, we studied addition and multiplication functions which have as arguments either **Double** , **Double-Double** or **Triple-Double** that resulted in **Double-Double** or in **Triple-Double**.
We noticed that the calculations of **Double-Double** numbers were more precise than the calculations of **Doubles** numbers and also that the operations of **Triple-Double** numbers were more precise than those of **Double-Double** numbers and **Double** numbers.

We also shown how to implement the $cr \log$ and calculate the bounds of each thanks to their relative error calculations. We managed to implement our logarithm thanks to the bounds of $cr \log_{fast}$ and $cr \log_{accurate}$. Inaddition to this, We have shown how bounds were used in the logarithm.
For each $cr \log$, we calculated α_i and also their $\log(\alpha_i)$. $cr \log fast$ used α_i which are **Double** and have precision of 71 bits, while $cr \log accurate$ used α_i which are **Double-Double** and have a precision of approximately 107 bits and the $cr \log_{advanced}$ used α_i which are **Triple-Double** and which have a precision of about 160 bits.

We noticed that, while the $cr \log_{fast}$ has faster internal calculations than those of $cr \log_{accurate}$, the internal operations of $cr \log_{accurate}$ is faster than those of du $cr \log_{advanced}$.

In our research, we managed to implement the logarithm with the calculation algorithms, and We tested it on millions of worst cases.

To our mind ,it would be necessary to test the speed of the code, by trying to make it faster. Like for example, changing the $cr \log_{fast}$ so that its error bound would be more smaller.
Thanks to this logarithm, we could implement the logarithm in base 2 ($\log_2$) and base 10 ($\log_{10}$). Moreover, we could Implement the logarithm with correct rounding for $binary80$ and $binary128$ and the same for ($\log_2$)

and ($\log_{10}$).

Annexes

In appendix, We find each algorithm with their programs in **Sage** and in **C**.

Add112

Algorithm 1 Algorithm **Add112** (**FastTwoSum**)

Input: a and b are 53-bit floating-point numbers
Condition: $|a| \geq |b|$
Output: s and t are 53-bit floating-point numbers : s: main value and t: error value.

1: $s = a + b$
2: $z = s - a$
3: $t = b - z$
4: return s, t

```
#Input : a and b are 53-bit floating-point numbers
def Add112(a,b):
    s = a+b
    z = s-a
    t = b-z
    # output: s and t are 53-bit floating-point numbers.
    # s: main value and t: error value
    return s,t
```

```
/*a and b are double numbers */
void Add112(double a, double b, double *s, double *t){
    *s = a+b ;
    double z = *s-a;
    *t = b-z;
    /* (s,t) is a double-double number*/
}
```

Add122

Algorithm 2 Algorithm **Add122**

Input: a is 53-bit floating-point numbers, b_h: main value and b_ℓ: error value
Condition: $|a| \geq |b_h|$
Condition: $|b_\ell| \leq u.|b_h|$
Output: s and t are 53-bit floating-point numbers : s: main value and t: error value.

1: $s, \ell = Add112(a, b_h)$
2: $t = \ell + b_\ell$
3: return s, t

```python
#Input : a is 53-bit floating-point numbers
# bh: main value and bl: error value
def Add122(a,bh,bl):
    s,l = Add112(a,bh)
    t = l+bl
    # output: s and t are 53-bit floating-point numbers.
    # s: main value and t: error value
    return s,t
```

```c
/* a is double-number and (bh,bl) is double-double number*/
void Add122(double a, double bh, double bl, double *s, double *t){
    double l;
    Add112(a, bh,s,&l);
    *t = l+bl;
    /*(s,t) is a double-double number*/
}
```

Add222

Algorithm 3 Algorithm **Add222**

Input: a_h and b_h are main values, a_ℓ and b_ℓ are error values
Condition: $|a_h| \geq |b_h|$
Condition: $|a_\ell| \leq u.|a_h|$ and $|b_\ell| \leq u.|b_h|$
Output: s and t are 53-bit floating-point numbers : s: main value and t: error value.

1: $s, \ell = Add112(a_h, b_h)$
2: $m = \ell + a_\ell$
3: $t = m + b_\ell$
4: return s, t

```python
#Input : ah and bh are main values
# al and bl are error values
def Add222(ah,al,bh,bl):
    s,l = Add112(ah,bh)
    m = l+al
    t = m+bl
    # output: s and t are 53-bit floating-point numbers.
    # s: main value and t: error value
    return s,t
```

```c
/* (ah,al) and (bh,bl) double-double numbers*/
void Add222(double ah, double al, double bh, double bl, double *s, double *t){
    double l,m;
    Add112(ah,bh,s,&l);
    m = l+al;
    *t = m+bl;
    /*(s,t) is a double-double number*/
}
```

Mul112

Algorithm 4 Algorithm **Mul112** (DEKKER-PRODUCT)

Input: a and b are 53-bit floating-point numbers.

Output: r_1 and r_2 are 53-bit floating-point numbers: r_1: main value and r_2: error value.

$r1 = a \times b$

$r2 = FMA(a, b, -r1)$

return (r_1, r_2)

```c
/*a and b are double numbers */
void Mul112(double a, double b, double *r1, double *r2){
    *r1 = a * b;
    *r2 = __builtin_fma (a, b, -*r1);
    /* (r1,r2) is a double number */
}
```

```python
#Input : a and b are 53-bit floating-point numbers
def Mul112(a,b):
    r1 = a*b
    r2 = fma(a,b,-r1)
    # output: r1 and r2 are 53-bit floating-point numbers.
    # r1: main value and r2: error value
    return (r1,r2)
```

Mul122

Algorithm 5 Algorithm **Mul122**

Input: a is 53-bit floating-point numbers, b_h: main value and b_ℓ: error value

Condition: $|b_\ell| \leq u.|b_h|$

Output: r_1 and r_2 are 53-bit floating-point numbers : r_1: main value and r_2: error value

1: $t_1, t_2 = Mul112(a, b_h)$
2: $t_3 = a \times b_\ell$
3: $t_4 = t_2 + t_3$
4: $r_1, r_2 = Add112(t_1, t_4)$
5: return (r_1, r_2)

```python
#Input : a is 53-bit floating-point numbers
# bh: main value and bl: error value
def Mul122(a,bh,bl):
    t1,t2 = Mul112(a,bh)
    t3 = a*bl
    t4 = t2+t3
    r1,r2 = Add112(t1,t4)
    # output: r1 and r2 are 53-bit floating-point numbers.
    # r1: main value and r2: error value
    return (r1,r2)
```

```c
/* a is double-number and (bh,bl) is double-double number*/
void Mul122(double a, double bh, double bl, double *r1, double *r2){
    double t1,t2;
    Mul112(a,bh,&t1,&t2);
    double t3,t4;
    t3 = a*bl;
    t4 = t2+t3;
    Add112(t1,t4,r1,r2);
    /* (r1,r2) is a double number */
}
```

Mul222

Algorithm 6 Algorithm **Mul222**

Input: a_h and b_h are main values, a_ℓ and b_ℓ are error values
Output: r_1 and r_2 are 53-bit floating-point numbers: r_1: main value and r_2: error value.

1: $t_1, t_2 = Mul112(a_h, b_h)$
2: $t_3 = a_h \times b_\ell$
3: $t_4 = b_h \times a_\ell$
4: $t_5 = t_3 + t_4$
5: $t_6 = t_2 + t_5$
6: $r_1, r_2 = Add112(t_1, t_6)$
7: return (r_1, r_2)

```
#Input : ah and bh are main values
# al and bl are error values
def Mul222(ah,al,bh,bl):
    t1,t2 = Mul112(ah,bh)
    t3 = ah*bl
    t4 = bh*al
    t5,t6 = Add112Cond(t3,t4)
    r1,r2 = Add222Cond(t1,t2,t5,t6)
    # output: r1 and r2 are 53-bit floating-point numbers.
    # r1: main value and r2: error value
    return (r1,r2)
```

```
/* (ah,al) and (bh,bl) double-double numbers*/
void Mul222(double ah, double al, double bh, double bl, double *r1, double *r2){
    double t1,t2;
    Mul112(ah, bh, &t1, &t2);
    double t3,t4;
    t3 = ah*bl;
    t4 = bh*al;
    double t5,t6;
    t5 = t3+t4;
    t6 = t2+t5;
    Add112(t1,t6,r1,r2);
    /* (r1,r2) is a double number */
}
```

Add133

Algorithm 7 Algorithm **Add133**

Input: a is a double number and b_h, b_m, b_ℓ is a triple-double numbers.
Condition: $|a| \geq |b_h|$, $|b_m| \leq u.|b_h|$, $|b_\ell| \leq u.|b_m|$ and $|b_\ell| \leq u^2.|b_h|$
Output: r_h, r_m, r_ℓ is a triple-double numbers.

1: $r_h, t_1 = Add112(a, b_h)$
2: $t_2, t_3 = Add112(t_1, b_m)$
3: $t_4 = t_3 + b_\ell$
4: $r_m, r_\ell = Add112(t_2, t_4)$
5: return r_h, r_m, r_ℓ

```
#Input : a is a double numbers
# bh,bm,bl is a triple-double numbers
def Add133(a,bh,bm,bl):
    rh,t1 = Add112(a,bh)
    t2,t3 = Add112(t1,bm)
    t4 = t3+bl
    rm,rl = Add112(t2,t4)
    #output :rh,rm,rl is a triple-double numbers
    return rh,rm,rl
```

```
/* a is a double numbers
   bh,bm,bl is a triple-double numbers*/
void Add133(double a, double bh, double bm, double bl, double *rh, double *rm, double *rl){
    double t1,t2,t3,t4;
    Add112(a,bh,rh,&t1);
    Add112(t1,bm,&t2,&t3);
    t4 = t3+bl;
    Add112(t2,t4,rm,rl);
    /*rh,rm,rl is a triple-double numbers*/
}
```

Add333

Algorithm 8 Algorithm **Add333**

Input: a_h, a_m, a_ℓ and b_h, b_m, b_ℓ are triple-double numbers
Condition: $|b_h| \leq \frac{3}{4}.|a_h|$
Condition: $|a_m| \leq 2^{-\alpha_o}.|a_h|$
Condition: $|a_\ell| \leq 2^{-\alpha_u}.|a_m|$
Condition: $|b_m| \leq 2^{-\beta_o}.|b_h|$
Condition: $|b_\ell| \leq 2^{-\beta_u}.|b_m|$
Condition: $\alpha_o \geq 4$, $\alpha_u \geq 1$, $\beta_o \geq 4$, $\beta_u \geq 1$
Output: r_h, r_m, r_ℓ is a triple-double numbers.

$r_h, t_1 = Add112(a_h, b_h)$
$t_2, t_3 = Add112Cond(a_m, b_m)$
$t_7, t_4 = Add112(t_1, t_2)$
$t_6 = a_\ell + b_\ell$
$t_5 = t_3 + t_4$
$t_8 = t_5 + t_6$
$r_m, r_\ell = Add112(t_7, t_8)$
return r_h, r_m, r_ℓ

```
def Add333(ah,am,al,bh,bm,bl):
    rh,t1 = Add112(ah,bh)
    t2,t3 =Add112(am,bm)
    t7,t4 = Add112(t1,t2)
    t6 = al+bl
    t5 = t3+t4
    t8 = t5+t6
    rm,rl = Add112(t7,t8)
    #output :rh,rm,rl is a triple-double numbers
    return rh,rm,rl
```

```
/*ah,am,al and bh,bm,bl are triple-double numbers*/
void Add333(double ah, double am, double al, double bh, double bm, double bl, double *rh,
     double *rm, double *rl){
    double t1,t2,t3,t4,t5,t6,t7,t8;
    Add112(ah,bh,rh,&t1);
    Add112Cond(am,bm,&t2,&t3);
    Add112(t1,t2,&t7,&t4);
    t6 = al+bl;
    t5 = t3+t4;
    t8 = t5+t6;
    Add112(t7,t8,rm,rl);
    /*rh,rm,rl is a triple-double numbers*/
}
```

Mul133

Algorithm 9 Algorithm **Mul133**

Input: a is a double number and b_h, b_m, b_ℓ is a triple-double numbers.
Condition: $|b_m| \le 2^{-\beta_o}.|b_h|$ with $\beta_o \ge 2$
Condition: $|b_\ell| \le 2^{-\beta_u}.|b_m|$ with $\beta_u \ge 2$
Output: r_h, r_m, r_ℓ is a triple-double numbers.

1: $r_h, t_2 = Mul112(a, b_h)$
2: $t_3, t_4 = Mul112(a, b_m)$
3: $t_5 = a \times b_\ell$
4: $t_9, t_7 = Add112(t_2, t_3)$
5: $t_8 = t_4 + t_5$
6: $t_{10} = t_7 + t_8$
7: $r_m, r_\ell = Add112(t_9, t_{10})$
8: return r_h, r_m, r_ℓ

```
#Input : a is a double numbers
# bh,bm,bl is a triple-double numbers
def Mul133(a,bh,bm,bl):
    rh,t2 = Mul112(a,bh)
    t3,t4 = Mul112(a,bm)
    t5 = a*bl
    t9,t7 = Add112Cond(t2,t3)
    t8 = t4+t5
    t10 = t7+t8
    rm,rl = Add112Cond(t9,t10)
    #output :rh,rm,rl is a triple-double numbers
    return rh,rm,rl
```

```
/* a is a double numbers
   bh,bm,bl is a triple-double numbers*/
void Mul133(double a, double bh, double bm, double bl, double *rh, double *rm, double *rl){
    double t2,t3,t4,t5,t7,t8,t9,t10;
    Mul112(a,bh,rh,&t2);
    Mul112(a,bm,&t3,&t4);
    t5 = a*bl;
    Add112(t2,t3,&t9,&t7);
    t8 = t4+t5;
    t10 = t7+t8;
    Add112(t9,t10,rm,rl);
    /*rh,rm,rl is a triple-double numbers*/
}
```

Mul233

Algorithm 10 Algorithm **Mul233**

Input: a_h, a_ℓ is a double-double and b_h, b_m, b_ℓ is a triple-double numbers.
Condition: $|a_\ell| \leq u.|a_h|$,
Condition: $|b_m| \leq 2^{-\beta_o}.|b_h|$ with $\beta_o \geq 2$
Condition: $|b_\ell| \leq 2^{-\beta_u}.|b_m|$ with $\beta_u \geq 1$.
Output: r_h, r_m, r_ℓ is a triple-double numbers.

$r_h, t_1 = Mul112(a_h, b_h)$
$t_2, t_3 = Mul112(a_h, b_m)$
$t_4, t_5 = Mul112(a_h, b_\ell)$
$t_6, t_7 = Mul112(a_\ell, b_h)$
$t_8, t_9 = Mul112(a_\ell, b_m)$
$t_{10} = a_\ell \times b_\ell$
$t_{11}, t_{12} = Add222(t_2, t_3, t_4, t_5)$
$t_{13}, t_{14} = Add222(t_6, t_7, t_8, t_9)$
$t_{15}, t_{16} = Add222(t_{11}, t_{12}, t_{13}, t_{14})$
$t_{17}, t_{18} = Add112(t_1, t_{10})$
$r_m, r_\ell = Add222(t_{17}, t_{18}, t_{15}, t_{16})$
return r_h, r_m, r_ℓ

```
#Input : ah,al is a double-double numbers
# bh,bm,bl is a triple-double numbers
def Mul233(ah,al,bh,bm,bl):
    rh,t1 = Mul112(ah,bh)
    t2,t3 = Mul112(ah,bm)
    t4,t5 = Mul112(ah,bl)
    t6,t7 = Mul112(bh,al)
    t8,t9 = Mul112(al,bm)
    t10 = al*bl
    t11,t12 = Add222Cond(t2,t3,t4,t5)
    t13,t14 = Add222Cond(t6,t7,t8,t9)
    t15,t16 = Add222Cond(t11,t12,t13,t14)
    t17,t18 = Add112Cond(t1,t10)
    rm,rl = Add222Cond(t17,t18,t15,t16)
    #output :rh,rm,rl is a triple-double numbers
    return rh,rm,rl
```

```
/* ah,al is a double-double numbers
   bh,bm,bl is a triple-double numbers*/
void Mul233(double ah, double al, double bh, double bm, double bl, double *rh, double *rm, double *rl){
    double t1,t2,t3,t4,t5,t6,t7,t8,t9,t10;
    double t11,t12,t13,t14,t15,t16,t17,t18;
```

```
    Mul112(ah,bh,rh,&t1);
    Mul112(ah,bm,&t2,&t3);
    Mul112(ah,bl,&t4,&t5);
    Mul112(bh,al,&t6,&t7);
    Mul112(al,bm,&t8,&t9);
    t10 = al*bl;
    Add222(t2,t3,t4,t5,&t11,&t12);
    Add222(t6,t7,t8,t9,&t13,&t14);
    Add222(t11,t12,t13,t14,&t15,&t16);
    Add112(t1,t10,&t17,&t18);
    Add222(t17,t18,t15,t16,rm,rl);
    /*rh,rm,rl is a triple-double numbers*/
}
```

Mul333

Algorithm 11 Algorithm **Mul333**

Input: a_h, a_m, a_ℓ and b_h, b_m, b_ℓ are triple-double numbers
Condition: $|a_m| \le 2^{-\alpha_o}.|a_h|$
Condition: $|a_\ell| \le 2^{-\alpha_u}.|a_m|$
Condition: $|b_m| \le 2^{-\beta_o}.|b_h|$
Condition: $|b_\ell| \le 2^{-\beta_u}.|b_m|$
Condition: $\alpha_o \ge 2$, $\alpha_u \ge 2$, $\beta_o \ge 2$, $\beta_u \ge 2$
Output: r_h, r_m, r_ℓ is a triple-double numbers

$r_h, t_1 = Mul112(a_h, b_h)$
$t_2, t_3 = Mul112(a_h, b_m)$
$t_4, t_5 = Mul112(a_m, b_h)$
$t_6, t_7 = Mul112(a_m, b_m)$
$t_8 = a_h \times b_\ell$
$t_9 = a_\ell \times b_h$
$t_{10} = a_m \times b_\ell$
$t_{11} = a_\ell \times b_m$
$t_{12} = t_8 + t_9$
$t_{13} = t_{10} + t_{11}$
$t_{14}, t_{15} = Add112(t_1, t_6)$
$t_{16} = t_7 + t_{15}$
$t_{17} = t_{12} + t_{13}$
$t_{18} = t_{16} + t_{17}$
$t_{19}, t_{20} = Add112(t_{14}, t_{18})$
$t_{21}, t_{22} = Add222(t_2, t_3, t_4, t_5)$
$r_m, r_\ell = Add222(t_{21}, t_{22}, t_{19}, t_{20})$
return r_h, r_m, r_ℓ

```
#Input : ah,am,al and bh,bm,bl are triple-double numbers
def Mul333(ah,am,al,bh,bm,bl):
    rh,t1 = Mul112Cond(ah,bh)
    t2,t3 = Mul112Cond(ah,bm)
    t4,t5 = Mul112Cond(bh,am)
    t6,t7 = Mul112Cond(am,bm)
    t8 = ah*bl
    t9 = al*bh
    t10 = am*bl
    t11 = al*bm
    t12 = t8+t9
    t13 = t10+t11
    t14,t15 = Add112Cond(t1,t6)
    t16 = t7+t15
```

```
    t17 = t12+t13
    t18 = t16+t17
    t19,t20 = Add112Cond(t14,t18)
    t21,t22 = Add222Cond(t2,t3,t4,t5)
    rm,rl = Add222Cond(t21,t22,t19,t20)
    #output :rh,rm,rl is a triple-double numbers
    return rh,rm,rl
```

```
/*ah,am,al and bh,bm,bl are triple-double numbers*/
void Mul333(double ah, double am, double al, double bh, double bm, double bl, double *rh,
    double *rm, double *rl){
    double t1,t2,t3,t4,t5,t6,t7,t8,t9,t10;
    double t11,t12,t13,t14,t15,t16,t17,t18,t19,t20;
    double t21,t22;
    Mul112(ah,bh,rh,&t1);
    Mul112(ah,bm,&t2,&t3);
    Mul112(am,bh,&t4,&t5);
    Mul112(am,bm,&t6,&t7);
    t8 = ah*bl;
    t9 = al*bh;
    t10 = am*bl;
    t11 = al*bm;
    t12 = t8+t9;
    t13 = t10+t11;
    Add112(t1,t6,&t14,&t15);
    t16 = t7+t15;
    t17 = t12+t13;
    t18 = t16+t17;
    Add112(t14,t18,&t19,&t20);
    Add222(t4,t5,t2,t3,&t21,&t22);
    Add222(t21,t22,t19,t20,rm,rl);
    /*rh,rm,rl is a triple-double numbers*/
}
```

crlogfast

Algorithm 12 Algorithm $crlog_{fast}$ **for x is not to close to** 1

Condition: x is a **double** number and is not to close to 1.
Condition: m_1 and i are results made after calculating x.
Condition: $table_{\alpha_i}$ $table_{\log(\alpha_i)}$ and are calculated with our method with **Tang** and **Gal**'s method.
Condition: $log(2)$ in **double-double**: $h_{\log 2} = 0x1.62e42fefa38p-1$ and $l_{\log 2} = 0x1.ef35793c7673p-45$
Input: m_1 is a **double** number, i is a **Integer** number. and $table_{\alpha_i}$ and $table_{\log(\alpha_i)}$ are tables of **double** numbers and f a polynomial of degree with coefficient f_j calculated with **Sollya** with $1 \le j \le 7$
Condition: $1 \le m_1 \le 2$, $0 \le i < 256$ and size of $table_{\alpha_i} = 256$ and size of $table_{\log(\alpha_i)} = 256$, $\alpha_i = table_{\alpha_i}[i]$ and $\log_{\alpha_i} = table_{\log(\alpha_i)}[i]$
Output: $\log(x)$ in **double** number.

$h_r, \ell_r = Mul112(m_1, \alpha_i)$
$h, \ell = Add122(-1, h_r, \ell_r)$
$hh_2 = h \times h$
$ff_5 = f_5 \times h$
$ff_4 = f_4 + ff_5$
$ff_7 = f_7 \times h$
$ff_6 = f_6 + ff_7$
$cr_{logfast_2} = hh_2 \times ff_6$
$cr_{logfast_1} = ff_4 + cr_{logfast_2}$
$hh_2, \ell\ell_2 = Mul222(h, \ell, h, \ell)$
$hh_4, \ell\ell_4 = Mul222(hh_2, \ell\ell_2, hh_2, \ell\ell_2)$
$ffh_0, ff\ell_0 = Mul122(f_1, h, \ell)$
$ffh_3, ff\ell_3 = Mul122(f_3, h, \ell)$
$ffh_2, ff\ell_2 = Add122(f_2, ffh_3, ff\ell_3)$
$fhx_2, ff\ell x_2 = Mul222(hh_2, \ell\ell_2, ffh_2, ff\ell_2)$
$ffhx_4, ff\ell x_4 = Mul122(cr_{logfast_1}, hh_4, \ell\ell_4)$
$ffhx_0, ff\ell x_0 = Add222(ffh_0, ff\ell_0, ffhx_2, ff\ell x_2)$
$h_4, \ell_4 = Add222(ffhx_0, ff\ell x_0, ffhx_4, ff\ell x_4)$
$h_5, \ell_5 = Add122(\log_{\alpha_i}, h_4, \ell_4)$
$e_{hlog2}, e_{\ell log2} = Mul122(e, h_{\log 2}, \ell_{\log 2})$
$h_6, \ell_6 = Add222(e_{hlog2}, e_{\ell log2}, h_5, \ell_5)$
return (h_6, ℓ_6)

```
# Input : x is a 53-bit floating-point numbers
# f is an approximation polynomial of log(1+x)
def cr_log_fast_path(x,f):

    #If x is NAN return NAN
    if x==NaN:
        return NaN

    # If x=+0 or -0 return -INFINITY
    if x == 0:
        return -oo

    # If x = +INFINITY return +INFINITY
    if x == +oo:
        return +oo

    # if x<1 and close to 1 return with the function f
    if (x> RR('0x1.fe7814e49392fp-1',16) and x<1) :

        # we calculated len(f.list())
        F =f.list()
        F.reverse()
        lenf = len(F)

        hr,lr =Add112(-1,x)

        # F[i] is the coefficient of each monomial :x^(len(F)-1-i)

        #We are turning F[0] in hf,lf with hf,: main value
        # lf: error value
        hf,lf = Split(F[0])

        # multiply (hr,lr) by (hf,lf) result (h,l)
        h,l = Mul222(hf,lf,hr,lr)

        for i in range(1,lenf-1):
            #We are turning F[i] in hfi,lfi with hfi: main value
            # lfi: error value
            hfi,lfi = Split(F[i])

            # add (hfi,lfi) by (h,l) result (h1,l1)
            h1,l1 = Add222(hfi,lfi,h,l)

            # multiply (hr,lr) by (h1,l1) result (h,l)
            h,l = Mul222(h1,l1,hr,lr)

        return h,l

    # if x>1 and close to 1 return with function f
    if x> 1 and x < RR('0x1.00068db8bac71p+0',16) :
         # we calculated len(f.list())
        F =f.list()
        F.reverse()
        lenf = len(F)

        hr,lr =Add112(x,-1)

        # F[i] is the coefficient of each monomial :x^(len(F)-1-i)

```

```
        #We are turning F[0] in hf,lf with hf,: main value
        # lf: error value
        hf,lf = Split(F[0])

        # multiply (hr,lr) by (hf,lf) result (h,l)
        h,l = Mul222(hf,lf,hr,lr)

        for i in range(1,lenf-1):
            #We are turning F[i] in hfi,lfi with hfi: main value
            # lfi: error value
            hfi,lfi = Split(F[i])

            # add (hfi,lfi) by (h,l) result (h1,l1)
            h1,l1 = Add222(hfi,lfi,h,l)

            # multiply (hr,lr) by (h1,l1) result (h,l)
            h,l = Mul222(h1,l1,hr,lr)

        return h,l


    #s represents the sign
    #e represents the exposant
    #m represents fraction
    (s,m,e)=RR(x).sign_mantissa_exponent()
    #e=e-53


    e = e+53


    #If x is a negative number return NAN
    if s==-1 and e!=0:
        return NaN

    # If x is a subnormal
    if s==1 and e<0 and m!=0:

        v = m
        e = e-1

        while v< 2^52:
            v*=2

            e=e-1
        m1 =v*2.^(-52)


    #If x is normal
    else:
        #print(s,e,m)
        m1 = m*2.^(-52)
        e =e-1




    binary = RR(m1).str(2)[2:10] # we recover the 8 bits after the initial 1.
    i = int(binary,2) # i is the 8-bit integer
```

```


halpha_i_m,lalpha_i_m = Split(RR(table_alpha_modified[i],16)) #table computed for all i of alpha_i_m
                                        # such that log(alpha_i_m) is 71 bits accurate.
hlog_alpha_i_m,llog_alpha_i_m= Split(RR(table_log_alpha_modified[i],16))

# multiply (halpha_i_m,lalpha_i_m) by (hm,lm)
hr,lr = Mul122(m1,halpha_i_m,lalpha_i_m)

# add hR,lR by (-1) in double,double
h,l = Add122(-1,hr,lr)

# we calculated len(f.list())
F =f.list()
F.reverse()
lenf = len(F)

# F[i] is the coefficient of each monomial :x^(len(F)-1-i)

#We are turning F[0] in hf,lf with hf,: main value
# lf: error value
hf,lf = Split(F[0])

# multiply (hr,lr) by (hf,lf) result (h,l)
h1,l1 =Mul222(hf,lf,h,l)

for i in range(1,lenf-1):
    #We are turning F[i] in hfi,lfi with hfi: main value
    # lfi: error value
    hfi,lfi = Split(F[i])

    # add (hfi,lfi) by (h1,l1) result (h2,l2)
    h2,l2 = Add222(hfi,lfi,h1,l1)

    # multiply (h,l) by (h2,l2) result (h1,l1)
    h1,l1 = Mul222(h2,l2,h,l)



# Add (h1,l1) by (hlog_alpha_i_m,llog_alpha_i_m)
h5,l5 = Add222(hlog_alpha_i_m,llog_alpha_i_m,h1,l1)


# log(2) =(h_log2,l_log2)
h_log2 =RR('0x1.62e42fefa38p-1',16)
l_log2 = RR('0x1.ef35793c7673p-45',16)

#e*log(2)
elog2_h,elog2_l = Mul122(RR(e),h_log2,l_log2)


hlog_fi,llog_fi = Add222(elog2_h,elog2_l,h5,l5)


#output: hlog_fi and llog_fi are 53-bit floating-point numbers.
# hlog_fi: main value and llog_fi: error value

return RR(hlog_fi),RR(llog_fi)
```

```
static double cr_log_fast_path(double x, double *h6, double *l6){
    int s;
    int e;
    uint64_t m;
    extract (x,&s,&e, &m);

    /* If x is a negative number return NAN*/
    if ((s == 1) && (e != 0)){
        return NAN;
    }
    /*If x is NAN return NAN*/
    if ((s == 0) && (e == 0x7ff) && (m != 0)){
        return NAN;
    }
    /*If x=+0 ou -0 return INFINITY*/
    if (((s==1) && (e == 0) && (m == 0)) || ((s == 0) && (e == 0) && (m == 0))){
        return -(0x1p1023 +0x1p1023);
    }
    /*If x = +INFINITY return +INFINITY */
    if ((s == 0) && (e == 0x7ff)){
        return 0x1p1023 +0x1p1023;
    }
    /*The coefficients of the approximation polynomial of degree 7. */
    double f7 = 0x1.2152a2de69894p-3;
    double f6 = -0x1.555147415c204p-3;
    double f5 = 0x1.999997342c184p-3;
    double f4 = -0x1.fffffff57268p-3;
    double f3 = 0x1.55555555554cep-2;
    double f2 = -0x1p-1;
    double f1 = 0x1p+0;

    if ( (x> 0x1.fe7814e49392fp-1) && (x<1)) {
        double xx2 =(x-1) * (x-1);
        double ff4 = f4 + f5 * (x-1);
        double ff6 = f6 + f7 * (x-1);


        double cr_log_fast_1 = ff4 + xx2 * ff6;

        /* Add x by -1 result hr,lr */
        double hr,lr;
        Add112(x, -1, &hr, &lr);


        double hr2,lr2;
        /* Multiply (hr,lr) by (hr,lr) result (hr2,lr2) */
        Mul222(hr,lr,hr,lr,&hr2,&lr2);

        double hr4,lr4;
        /* Multiply (hr2,lr2) by (hr2,lr2) result (hr4,lr4) */
        Mul222(hr2,lr2,hr2,lr2,&hr4,&lr4);

        double ffh0,ffl0;
        /* Multiply f1 by (hr,lr) result (ffh0,ffl0) */
        Mul122(f1,hr,lr,&ffh0,&ffl0);

        double ffh3,ffl3;
        /* Multiply f3 by (hr,lr) result (ffh3,ffl3) */
        Mul122(f3,hr,lr,&ffh3,&ffl3);
```

```
        double ffh2,ffl2;
        /*Add f2 by (ffh3,ffl3) result (ffh2,ffl2) */
        Add122(f2,ffh3,ffl3,&ffh2,&ffl2);

        double ffhx2,fflx2;
        /* Multiply (hr2,lr2) by (ffh2,ffl2) result (ffhx2,fflx2) */
        Mul222(hr2,lr2,ffh2,ffl2,&ffhx2,&fflx2);

        double ffhx4,fflx4;
        /* Multiply (hr4,lr4) by cr_log_fast_1 result (ffhx4,fflx4) */
        Mul122(cr_log_fast_1,hr4,lr4,&ffhx4,&fflx4);

        double ffhx0,fflx0;
        /* Add (ffh0,ffl0) by (ffhx2,fflx2) result (ffhx0,fflx0) */
        Add222(ffh0, ffl0, ffhx2, fflx2, &ffhx0,&fflx0);

        /* Add (ffhx0,fflx0) by (ffhx4,fflx4) result (h6,l6) */
        Add222(ffhx0,fflx0, ffhx4,fflx4,h6,l6);

        return (*h6+*l6);


    }

    if ((x> 1) && (x < 0x1.00068db8bac71p+0)){

        double xx2 =(x-1) * (x-1);
        double ff4 = f4 + f5 * (x-1);
        double ff6 = f6 + f7 * (x-1);


        double cr_log_fast_1 = ff4 + xx2 * ff6;

        /* Add x by -1 result hr,lr */
        double hr,lr;
        Add112(x, -1, &hr, &lr);


        double hr2,lr2;
        /* Multiply (hr,lr) by (hr,lr) result (hr2,lr2) */
        Mul222(hr,lr,hr,lr,&hr2,&lr2);

        double hr4,lr4;
        /* Multiply (hr2,lr2) by (hr2,lr2) result (hr4,lr4) */
        Mul222(hr2,lr2,hr2,lr2,&hr4,&lr4);

        double ffh0,ffl0;
        /* Multiply f1 by (hr,lr) result (ffh0,ffl0) */
        Mul122(f1,hr,lr,&ffh0,&ffl0);

        double ffh3,ffl3;
        /* Multiply f3 by (hr,lr) result (ffh3,ffl3) */
        Mul122(f3,hr,lr,&ffh3,&ffl3);

        double ffh2,ffl2;
        /*Add f2 by (ffh3,ffl3) result (ffh2,ffl2) */
        Add122(f2,ffh3,ffl3,&ffh2,&ffl2);
```

```
        double ffhx2,fflx2;
        /* Multiply (hr2,lr2) by (ffh2,ffl2) result (ffhx2,fflx2) */
        Mul222(hr2,lr2,ffh2,ffl2,&ffhx2,&fflx2);

        double ffhx4,fflx4;
        /* Multiply (hr4,lr4) by cr_log_fast_1 result (ffhx4,fflx4) */
        Mul122(cr_log_fast_1,hr4,lr4,&ffhx4,&fflx4);

        double ffhx0,fflx0;
        /* Add (ffh0,ffl0) by (ffhx2,fflx2) result (ffhx0,fflx0) */
        Add222(ffh0, ffl0, ffhx2, fflx2, &ffhx0,&fflx0);

        /* Add (ffhx0,fflx0) by (ffhx4,fflx4) result (h6,l6) */
        Add222(ffhx0,fflx0, ffhx4,fflx4,h6,l6);

        return (*h6+*l6);

    }

    double m1;
    /*If x is a subnormal */
    if ((s ==0) && (e == 0) && (m !=0)) {
        uint64_t v = m;
        e = e - 1023;
        v = v*2;
        while (v < 0x10000000000000) {
            v *= 2;
            e--;
        }

        m1 =v *0x1p-52;

        u u;
        u.x = m1;
        m = u.i & 0xFFFFFFFFFFFFF;
    }
    /*If x is normal */
    else {
        m1 = 1 + m*0x1p-52;

        e = e - 1023;
    }
    /*We shift by 44 bits for to get the first 8 bits.*/
    uint64_t i = m>>44;

    /*k=8
    alpha_i_m is the double close to 2^k/(2^k+i) such that its log is very close to a double.
        */
    double alpha_i_m = table_alpha_i_modified[(int)i];


    double hr,lr;
    /*multiply (hm1,lm1) by (halpha_i_m,lalpha_i_m)*/
    Mul112(m1,alpha_i_m,&hr,&lr);

    double h,l;
    /*add (hr,lr) by -1*/
    Add122(-1,hr,lr,&h,&l);

```

```
      double hh2 = h * h;
      double ff4 = f4 + f5 * h;
      double ff6 = f6 + f7 * h;


   double cr_log_fast_1 = ff4 +hh2*ff6;

   double ll2,hh4,ll4;

   /* Multiply (h,l) by (h,l) result (hh2,ll2) */
   Mul222(h,l,h,l,&hh2,&ll2);

   /* Multiply (hh2,ll2) by (hh2,ll2) result (hh4,ll4) */
   Mul222(hh2,ll2,hh2,ll2,&hh4,&ll4);

   double ffh0,ffl0;
   /* Multiply f1 by (h,l) result (ffh0,ffl0) */
   Mul122(f1,h,l,&ffh0,&ffl0);

   double ffh3,ffl3;
   /* Multiply f3 by (h,l) result (ffh3,ffl3) */
   Mul122(f3,h,l,&ffh3,&ffl3);

   double ffh2,ffl2;
   /*Add f2 by (ffh3,ffl3) result (ffh2,ffl2) */
   Add122(f2,ffh3,ffl3,&ffh2,&ffl2);

   double ffhx2,fflx2;
   /* Multiply (hh2,ll2) by (ffh2,ffl2) result (ffhx2,fflx2) */
   Mul222(hh2,ll2,ffh2,ffl2,&ffhx2,&fflx2);

   double ffhx4,fflx4;
   /* Multiply (hh4,ll4) by cr_log_fast_1 result (ffhx4,fflx4) */
   Mul122(cr_log_fast_1,hh4,ll4,&ffhx4,&fflx4);

   double ffhx0,fflx0;
   /* Add (ffh0,ffl0) by (ffhx2,fflx2) result (ffhx0,fflx0) */
   Add222(ffh0, ffl0, ffhx2, fflx2, &ffhx0,&fflx0);

   double h4,l4;
   /* Add (ffhx0,fflx0) by (ffhx4,fflx4) result (h4,l4) */
   Add222(ffhx0,fflx0, ffhx4,fflx4,&h4,&l4);

   double log_alpha_i_m = table_log_alpha_i_modified[i];


   double h5,l5;
   /* add (hlog_alpha_i_m, llog_alpha_i_m)*/
   Add122(log_alpha_i_m, h4, l4, &h5, &l5);




   double h_log2 = 0x1.62e42fefa38p-1;
   double l_log2 = 0x1.ef35793c7673p-45;
   double e_hlog2,e_llog2;
```

```
    /* e*log(2) in (double,double) precision.*/
    Mul122(e, h_log2, l_log2, &e_hlog2, &e_llog2);



    /*adding with (h10,l10)*/
    Add222(e_hlog2, e_llog2,h5,l5,h6,l6);

    return (*h6+*l6);
}
```

crlogaccurate

Algorithm 13 Algorithm $crlog_{accurate}$ **for** x **is not to close to** 1

Condition: x is a **double** number and is not to close to 1.
Condition: m_1 and i are results made after calculating x.
Condition: $table_{\alpha_i}$ $table_{\log(\alpha_i)}$ are of tables of **Double-Double** and are calculated with **Tang**.
Condition: $log(2)$ in **double-double**: $h_{\log 2} = 0x1.62e42fefa39efp-1$ and $l_{\log 2} = 0x1.abc9e3b39803fp-56$
Input: m_1 is a **double** number, i is a **Integer** number. and $table_{\alpha_i}$ and $table_{\log(\alpha_i)}$ are tables of **double-double** numbers and f a polynomial of degree 11 with coefficient f_j calculated with **Sollya** with $1 \leq j \leq 11$ and transformed into **double-double**, G is table of f_j
Condition: $1 \leq m_1 \leq 2$, $0 \leq i < 256$ and size of $table_{\alpha_i} = 256$ and size of $table_{\log(\alpha_i)} = 256$, $\alpha_i = table_{h\alpha_i, l\alpha_i}[i]$ and $\log_{\alpha_i} = table_{h\log(\alpha_i), l\log(\alpha_i)}[i]$
Output: $\log(x)$ in **double** number.

1: $h_r, \ell_r = Mul122(m_1, \alpha_i)$
2: $h, \ell = Add122(-1, h_r, \ell_r)$
3: $h_1, \ell_1 = Mul222(G[0][0], G[0][1], h, \ell)$
4: $for\ (i = 1, i < 11, i++)$
5: $\quad h_2, \ell_2 = Add222(G[i][0], G[i][1], h_1, \ell_1)$
6: $\quad h_1, \ell_1 = Mul222(h_2, \ell_2, h, \ell)$
7: $h_5, \ell_5 = Add222(\log_{\alpha_i}, h_1, \ell_1)$
8: $e_{hlog2}, e_{\ell log2} = Mul122(e, h_{\log 2}, \ell_{\log 2})$
9: $h_6, \ell_6 = Add222(e_{hlog2}, e_{\ell log2}, h_5, \ell_5)$
10: return (h_6, ℓ_6)

```
# Input : x is a 53-bit floating-point numbers
# F is a list with Coefficient of f decomposed into double,double
def cr_log_accurate_path(x,F):

    # if x<1 and close to 1 return with the function f
    if x> RR('0x1.fe73451b9c74fp-1',16) and x<1 :


        # we calculated len(F)
        lenf = len(F)

        hr,lr = Add112(-1,x)

        # F[i] is the coefficient of each monomial :x^(len(F)-1-i)

        #We are turning F[0] in hf,lf with hf,: main value
```

```
        # lf: error value
        hf,lf = RR(F[0][0],16),RR(F[0][1],16)

        # multiply (hr,lr) by (hf,lf) result (h,l)
        h,l = Mul222(hr,lr,hf,lf)

        for i in range(1,lenf):
            #We are turning F[i] in hfi,lfi with hfi: main value
            # lfi: error value
            hfi,lfi = RR(F[i][0],16),RR(F[i][1],16)

            # add (hfi,lfi) by (h,l) result (h1,l1)
            h1,l1 = Add222(hfi,lfi,h,l)

            # multiply (hr,lr) by (h1,l1) result (h,l)
            h,l = Mul222(hr,lr,h1,l1)

        return h,l



     # if x>1 and close to 1 return with function f
    if x> 1 and x <= RR('0x1.0178e5916f543p+0',16) :


        # we calculated len(F)
        lenf = len(F)

        hr,lr = Add112(x,-1)

        # F[i] is the coefficient of each monomial :x^(len(F)-1-i)

        #We are turning F[0] in hf,lf with hf,: main value
        # lf: error value
        hf,lf = RR(F[0][0],16),RR(F[0][1],16)

        # multiply (hr,lr) by (hf,lf) result (h,l)
        h,l = Mul222(hr,lr,hf,lf)

        for i in range(1,lenf):
            #We are turning F[i] in hfi,lfi with hfi: main value
            # lfi: error value
            hfi,lfi = RR(F[i][0],16),RR(F[i][1],16)

            # add (hfi,lfi) by (h,l) result (h1,l1)
            h1,l1 = Add222(hfi,lfi,h,l)

            # multiply (hr,lr) by (h1,l1) result (h,l)
            h,l = Mul222(hr,lr,h1,l1)

        return h,l

    #s represents the sign
    #e represents the exposant
    #m represents fraction
    (s,m,e)=RR(x).sign_mantissa_exponent()
    #e=e-53
```

```
    e = e+53

    # If x is a subnormal
    if s==1 and e<=0 and m!=0:

        v = m
        e = e-1

        while v< 2^52:
            v*=2

            e=e-1
        m1 =v*2.^(-52)


    #If x is normal
    else:
        #print(s,e,m)
        m1 = m*2.^(-52)
        e =e-1




    binary = RR(m1).str(2)[2:10] # we recover the 8 bits after the initial 1.
    i = int(binary,2) # i is the 8-bit integer


    halpha_i,lalpha_i = RR(table_alpha_i[i][0],16),RR(table_alpha_i[i][1],16) #table computed for all i of alpha_i_m
                                                  # such that log(alpha_i_m) is 71 bits accurate.
    hlog_alpha_i,llog_alpha_i= RR(table_log_alpha_i[i][0],16),RR(table_log_alpha_i[i][1],16)

    # multiply (halpha_i_m,lalpha_i_m) by m1
    hr,lr = Mul122(m1,halpha_i,lalpha_i)

    # add hr,lr by (-1) in double,double
    h,l = Add122(-1.0,hr,lr)

    # we calculated len(f.list())
    lenf = len(F)

    # F[i] is the coefficient of each monomial :x^(len(F)-1-i) in double,double

    #We are turning F[0] in hf,lf with hf,: main value
    # lf: error value
    hf,lf = RR(F[0][0],16),RR(F[0][1],16)

    # multiply (hr,lr) by (hf,lf) result (h,l)
    h1,l1 = Mul222(hf,lf,h,l)

    for i in range(1,lenf):
        #We are turning F[i] in hfi,lfi with hfi: main value
        # lfi: error value
        hfi,lfi = RR(F[i][0],16),RR(F[i][1],16)

        # add (hfi,lfi) by (h1,l1) result (h2,l2)
        h2,l2 = Add222(hfi,lfi,h1,l1)
```

```
        # multiply (h,l) by (h2,l2) result (h1,l1)
        h1,l1 = Mul222(h2,l2,h,l)



    # Add (h1,l1) by (hlog_alpha_i,llog_alpha_i
    h5,l5 = Add222(hlog_alpha_i,llog_alpha_i,h1,l1)


    # log(2) =(h_log2,l_log2)
    h_log2 =RR('0x1.62e42fefa39efp-1',16)
    l_log2 = RR('0x1.abc9e3b39803fp-56',16)

    #e*log(2)
    elog2_h,elog2_l = Mul122(RR(e),h_log2,l_log2)


    hlog_fi,llog_fi = Add222(elog2_h,elog2_l,h5,l5)


    #output: hlog_fi and llog_fi are 53-bit floating-point numbers.
    # hlog_fi: main value and llog_fi: error value

    return hlog_fi,llog_fi
```

```
static double cr_log_accurate_path(double x, double *h6, double *l6){
    int s;
    int e;
    uint64_t m;
    extract (x,&s,&e, &m);
/* Special case*/
    if (x == 0x1.fb85251a3f26fp-1){
        *h6 = -0x1.1ff9b8e8b38bep-7;
        *l6 = -0x1.0b393919c1fa3p-109;
        return (*h6+*l6);
    }

    if (x == 0x1.fc65aa1908a66p-1){
        *h6 = -0x1.cecc4ad8d358bp-8;
        *l6 = -0x1.65a43e3cf2b61p-107;
        return (*h6+*l6);
    }

    /*The coefficients in double,double of the approximation polynomial of degree 11. */
    static const double G[11][2]= {
        {0x1.6d24dd22a92bp-4, 0x1.3e16deaf82f98p-58}, /* g11 decomposed into double, double*/
        {-0x1.99889bcf944ecp-4, -0x1.14f5635667f7p-58}, /* g10 decomposed into double,double*/
        {0x1.c71c5b2ab9b1bp-4, -0x1.bba8402b8dab8p-58}, /* g9 decomposed into double, double*/
        {-0x1.fffffed56645p-4, 0x1.fa475383e2fep-60}, /* g8 decomposed into double, double*/
        {0x1.249249248d599p-3, 0x1.a5c92da9d350cp-57}, /* g7 decomposed into double, double*/
        {-0x1.555555555553bp-3, 0x1.887e7d473e27p-57}, /* g6 decomposed into double,
```

```
            double*/
        {0x1.999999999999ap-3, -0x1.affa1a4be0d04p-57}, /* g5 decomposed into double,
            double*/
        {-0x1p-2, 0x1.590555132p-72}, /* g4 decomposed into double,double*/
        {0x1.5555555555555p-2, 0x1.55555540b5114p-56}, /* g3 decomposed into double,
            double*/
        {-0x1p-1, 0x1.b78p-98}, /* g2 decomposed into double,double*/
        {0x1p+0, 0x0p+0} /* g1 decomposed into double,double*/
    };


    if ((x> 0x1.fe73451b9c74fp-1) && (x<1)) {
        double hr,lr ;
        /*Add x by -1 result hr,lr*/
        Add112(-1,x,&hr,&lr);

        double hf = G[0][0];
        double lf = G[0][1];

        double h,l;
        /*Multiply (hf,lf) by r result (h,l)*/
        Mul222(hr,lr,hf,lf,&h,&l);

        double hfi,lfi,h1,l1;

        for (int i = 1;i<11;i++){
            hfi = G[i][0];
            lfi = G[i][1];

            /*Add (hfi,lfi) by (h,l) result (h1,l1)*/
            Add222(hfi,lfi,h,l,&h1,&l1);

            /*multiply (hr,lr) by (h1,l1) result (h,l)*/
            Mul222(hr,lr,h1,l1,&h,&l);

        }
        *h6 = h;
        *l6 = l;

        return (h+l);
    }

    if ((x> 1) && (x <= 0x1.0178e5916f543p+0)){
        double hr,lr ;
        /*Add x by -1 result r*/
        Add112(x,-1,&hr,&lr);

        double hf = G[0][0];
        double lf = G[0][1];

        double h,l;
        /*Multiply (hf,lf) by r result (h,l)*/
        Mul222(hr,lr,hf,lf,&h,&l);

        double hfi,lfi,h1,l1;

        for (int i = 1;i<11;i++){
            hfi = G[i][0];
            lfi = G[i][1];
```

```

        /*Add (hfi,lfi) by (h,l) result (h1,l1)*/
        Add222(hfi,lfi,h,l,&h1,&l1);

        /*multiply (hr,lr) by (h1,l1) result (h,l)*/
        Mul222(hr,lr,h1,l1,&h,&l);

    }
    *h6 = h;
    *l6 = l;

    return (h+l);
}
double m1;
/*If x is a subnormal */
if ((s ==0) && (e == 0) && (m !=0)) {
    uint64_t v = m;
    e = e - 1023;
    v = v*2;
    while (v < 0x10000000000000) {
        v *= 2;
        e--;
    }

    m1 =v *0x1p-52;

    u u;
    u.x = m1;
    m = u.i & 0xFFFFFFFFFFFFF;
}
/*If x is normal */
else {
    m1 = 1 + m*0x1p-52;

    e = e - 1023;
}

/*We shift by 44 bits for to get the first 8 bits.*/
uint64_t i = m>>44;

/*k=8
halpha_i,lalpha_i is the double,double of 2^k/(2^k+i).*/
double halpha_i = table_alpha_i[(int)i][0];
double lalpha_i = table_alpha_i[(int)i][1];

double hr,lr;
/*Multiply (halpha_i,lalpha_i_m) by m1 result (hr,lr) */
Mul122(m1,halpha_i, lalpha_i,&hr,&lr);

double h,l;
/*Add (hr,lr) by (-1) result (h,l)*/
Add122(-1,hr,lr,&h,&l);

double hf = G[0][0];
double lf = G[0][1];

double h1,l1;
/*Multiply (hf,lf) by (h,l) result (h1,l1)*/
Mul222(hf,lf,h,l,&h1,&l1);
```

```
    double hfi,lfi,h2,l2;

    for (int i = 1;i<11;i++){
        hfi = G[i][0];
        lfi = G[i][1];

        /*Add (hfi,lfi) by (h1,l1) result (h2,l2)*/
        Add222(hfi,lfi,h1,l1,&h2,&l2);

        /*multiply (h,l) by (h2,l2) result (h1,l1)*/
        Mul222(h2,l2,h,l,&h1,&l1);

    }
    /*k=8
    hlog_alpha_i,llog_alpha_i is the double,double of log(2^k/(2^k+i)).*/
    double hlog_alpha_i = table_log_alpha_i[(int)i][0];
    double llog_alpha_i = table_log_alpha_i[(int)i][1];


    double h5,l5;
    /*Add (h1,l1) by (hlog_alpha_i,llog_alpha_i) result (h5,l5)*/
    Add222(hlog_alpha_i,llog_alpha_i,h1,l1,&h5,&l5);

    double h_log2 = 0x1.62e42fefa39efp-1;
    double l_log2 = 0x1.abc9e3b39803fp-56;
    double e_hlog2,e_llog2;
    /* e*log(2) in (double,double) precision.*/
    Mul122(e, h_log2, l_log2, &e_hlog2, &e_llog2);



    /*adding (e_hlog2,e_llog2) by (h5,l5) result (h6,l6)*/
    Add222(e_hlog2, e_llog2, h5, l5, h6,l6);

    return (*h6+*l6);


}
```

crlogadvanced

Algorithm 14 Algorithm $crlog_{advanced}$ **for x is not to close to** 1

Condition: x is a **double** number and is not to close to 1.

Condition: m_1 and i are results made after calculating x.

Condition: $table_{\alpha_i}$ $table_{\log(\alpha_i)}$ are of tables of **triple-Double** and are calculated with **Tang**.

Condition: $log(2)$ in **triple-double**: $h_{\log 2} = 0x1.62e42fefa39efp - 1$, $lm_{\log 2} = 0x1.abc9e3b39803fp - 56$ and $\ell_{\log 2} = 0x1.7b57a079a1934p - 111$

Input: m_1 is a **double** number, i is a **Integer** number. and $table_{\alpha_i}$ and $table_{\log(\alpha_i)}$ are tables of **triple-double** numbers and f a polynomial of degree 13 with coefficient f_j calculated with **Sollya** with $1 \leq j \leq 13$ and transformed into **triple-double**, U is table of f_j

Condition: $1 \leq m_1 \leq 2$, $0 \leq i < 256$ and size of $table_{\alpha_i} = 256$ and size of $table_{\log(\alpha_i)} = 256$, $\alpha_i = table_{h\alpha_i, m\alpha_i, l\alpha_i}[i]$ and $\log_{\alpha_i} = table_{h\log(\alpha_i), m\log(\alpha_i, l\log(\alpha_i)}[i]$

Output: $\log(x)$ in **double** number.

$h_r, m_r, \ell_r = Mul133(m_1, \alpha_i)$
$h, m, \ell = Add133(-1, h_r, m_r, \ell_r)$
$h_1, m_1, \ell_1 = Mul222(U[0][0], U[0][1], U[0][2], h, m, \ell)$
$for\ (i = 1, i < 13, i++)$
  $h_2, \ell_2 = Add222(U[i][0], U[i][1], U[i][2], h_1, m_1, \ell_1)$
  $h_1, m_1, \ell_1 = Mul222(h_2, m_2, \ell_2, h, m, \ell)$
$h_5, m_5, \ell_5 = Add222(\log_{\alpha_i}, h_1, m_1, \ell_1)$
$e_{hlog2}, e_{mlog2}, e_{\ell log2} = Mul122(e, h_{\log 2}, m_{\log 2}, \ell_{\log 2})$
$h_6, m_6, \ell_6 = Add222(e_{hlog2}, e_{mlog2}, e_{\ell log2}, h_5, m_5, \ell_5)$
$m_\ell = m_6 + \ell_6$ return (h_6, m_ℓ)

```
# Input : x is a 53−bit floating−point numbers
# F is a list with Coefficient of f decomposed into triple−double
def cr_log_accurate_path_advanced(x,F):

    # if x> 1.17549435∗10^(−38) and x<1 return with the function f
    if x> RR('0x1.fba5e353f7ceep−1',16) and x<1 :


        # we calculated len(F)
        lenf = len(F)

        hr,lr = Add112(−1.0,x)

```

```
        # F[i] is the coefficient of each monomial :x^(len(F)-1-i)

        #We are turning F[0] in hf,lf with hf,: main value
        # lf: error value
        hf,mf,lf = RR(F[0][0],16),RR(F[0][1],16),RR(F[0][2],16)

        # multiply (hr,lr) by (hf,lf) result (h,l)
        h,m,l = Mul233(hr,lr,hf,mf,lf)

        for i in range(1,lenf):
            #We are turning F[i] in hfi,lfi with hfi: main value
            # lfi: error value
            hfi,mfi,lfi = RR(F[i][0],16),RR(F[i][1],16),RR(F[i][2],16)

            # add (hfi,lfi) by (h,l) result (h1,l1)
            h1,m1,l1 = Add333(hfi,mfi,lfi,h,m,l)

            # multiply (hr,lr) by (h1,l1) result (h,l)
            h,m,l = Mul233(hr,lr,h1,m1,l1)

        return h,m,l



    # if x> 1.17549435*10^(-38) and x<1 return with the function f
    if x> 1 and x <= RR('0x1.0178e5916f543p+0',16) :


        # we calculated len(F)
        lenf = len(F)

        hr,lr = Add112(x,-1.0)

        # F[i] is the coefficient of each monomial :x^(len(F)-1-i)

        #We are turning F[0] in hf,lf with hf,: main value
        # lf: error value
        hf,mf,lf = RR(F[0][0],16),RR(F[0][1],16),RR(F[0][2],16)

        # multiply (hr,lr) by (hf,lf) result (h,l)
        h,m,l = Mul233(hr,lr,hf,mf,lf)

        for i in range(1,lenf):
            #We are turning F[i] in hfi,lfi with hfi: main value
            # lfi: error value
            hfi,mfi,lfi = RR(F[i][0],16),RR(F[i][1],16),RR(F[i][2],16)

            # add (hfi,lfi) by (h,l) result (h1,l1)
            h1,m1,l1 = Add333(hfi,mfi,lfi,h,m,l)

            # multiply (hr,lr) by (h1,l1) result (h,l)
            h,m,l = Mul233(hr,lr,h1,m1,l1)

        return h,m,l

    #s represents the sign
    #e represents the exposant
    #m represents fraction
    (s,m,e)=RR(x).sign_mantissa_exponent()
```

```
    #e=e-53


    e = e+53

    # If x is a subnormal
    if s==1 and e<=0 and m!=0:

        v = m
        e = e-1

        while v< 2^52:
            v*=2

            e=e-1
        m1 =v*2.^(-52)

   #If x is normal
    else:
        #print(s,e,m)
        m1 = m*2.^(-52)
        e =e-1

    binary = RR(m1).str(2)[2:10] # we recover the 8 bits after the initial 1.
    i = int(binary,2) # i is the 8-bit integer


    halpha_i,malpha_i,lalpha_i = RR(table_alpha_i_triple[i][0],16),RR(table_alpha_i_triple[i][1],16),RR(table_alpha_i_triple[i][2],16) #table computed for all i of alpha_i_m
                                            # such that log(alpha_i_m) is 71 bits accurate.
    hlog_alpha_i,mlog_alpha_i,llog_alpha_i= RR(table_log_alpha_i_triple[i][0],16),RR(table_log_alpha_i_triple[i][1],16),RR(table_log_alpha_i_triple[i][2],16)

    # multiply (halpha_i_m,lalpha_i_m) by m1
    hr,mr,lr = Mul133(m1,halpha_i,malpha_i,lalpha_i)

    # add hr,mr,lr by (-1) in triple-double
    h,m,l = Add133(-1.0,hr,mr,lr)

    # we calculated len(f.list())
    lenf = len(F)

    # F[i] is the coefficient of each monomial :x^(len(F)-1-i) in double,double

    #We are turning F[0] in hf,lf with hf,: main value
    # lf: error value
    hf,mf,lf = RR(F[0][0],16),RR(F[0][1],16),RR(F[0][2],16)

    # multiply (hr,lr) by (hf,lf) result (h,l)
    h1,m1,l1 = Mul333(h,m,l,hf,mf,lf)

    for i in range(1,lenf):
        #We are turning F[i] in hfi,lfi with hfi: main value
        # lfi: error value
        hfi,mfi,lfi = RR(F[i][0],16),RR(F[i][1],16),RR(F[i][2],16)

        # add (hfi,lfi) by (h1,l1) result (h2,l2)
        h2,m2,l2 = Add333(hfi,mfi,lfi,h1,m1,l1)
```

```
        # multiply (h,l) by (h2,l2) result (h1,l1)
        h1,m1,l1 = Mul333(h,m,l,h2,m2,l2)



    # Add (h1,l1) by (hlog_alpha_i,llog_alpha_i
    h5,m5,l5 = Add333(hlog_alpha_i,mlog_alpha_i,llog_alpha_i,h1,m1,l1)


    # log(2) =(h_log2,l_log2)
    h_log2 =RR('0x1.62e42fefa39efp-1',16)
    m_log2 =RR('0x1.abc9e3b39803fp-56',16)
    l_log2 = RR('0x1.7b57a079a1934p-111',16)

    #e*log(2)
    elog2_h,elog2_m,elog2_l = Mul133(RR(e),h_log2,m_log2,l_log2)


    hlog_fi,mlog_fi,llog_fi = Add333(elog2_h,elog2_m,elog2_l,h5,m5,l5)


    #output: hlog_fi and llog_fi are 53-bit floating-point numbers.
    # hlog_fi: main value and llog_fi: error value

    return hlog_fi, mlog_fi, llog_fi
```

```
static double cr_log_accurate_advanced_path(double x){
    int s;
    int e;
    uint64_t m;
    double h6,ml;
    /*special cases*/
    if (x == 0x1.c7e1077f9aec2p-1) {
        h6 = -0x1.db88b1160524bp-4;
        ml = -0x1.ffffffffffffcp-58;
         return (h6+ml);
    }
    if (x == 0x1.f8db13b0e98a3p-1) {
        h6 = -0x1.cc7365166e597p-7;
        ml = 0x1.ffffffffffccp-61;
         return (h6+ml);
    }
    if (x == 0x1.acb8cf13bc769p-2) {
        h6 = -0x1.bdc7955d1482cp-1;
        ml = 0x1.6c68c50e2ff8p-105;
         return (h6+ml);
    }
    if (x == 0x1.9309142b73ea6p-1) {
        h6 = -0x1.ea16274b0109bp-3;
        ml = 0x1.705633d4614a5p-106;
         return (h6+ml);
    }
    if (x == 0x1.98a04e0833091p-1) {
        h6 = -0x1.cddf723d3e52fp-3;
        ml = 0x1.a0636ba8c84a5p-106;
         return (h6+ml);
```

```
    }
    if (x == 0x1.baded30cbf1c4p-1) {
        h6 = -0x1.290ea09e36479p-3;
        ml = 0x1.3339fcb8b1c92p-111;
         return (h6+ml);
    }
    if (x == 0x1.c7f14af0a08ebp-1) {
        h6 = -0x1.daf693d64fadap-4;
        ml = 0x1.4f60080a4d88dp-109;
        return (h6+ml);
    }

    if (x == 0x1.f3c35328f1d5dp-1) {
        h6 = -0x1.8c56ff5326197p-6;
        ml = 0x1.24265dc55252p-105;
         return (h6+ml);
    }

    if (x == 0x1.f8156947924c8p-1) {
        h6 = -0x1.fe9ad6761218dp-7;
        ml = 0x1.39b6b04fc1d53p-105;
         return (h6+ml);
    }

    extract (x,&s,&e, &m);


    /*The coefficients in triple-double numbers of the approximation polynomial of degree
         14. */
    static const double U[14][3]= {
        {-0x1.1d35b84c5cc56p-4, 0x1.780337dcf5705p-58, -0x1.b91450d691e2p-114}, /* u14 decomposed into triple-double*/
        {0x1.3afccbd40d20fp-4, 0x1.9c4dd4dab116p-58, 0x1.3a1e112af7efp-112}, /* u13 decomposed into triple-double*/
        {-0x1.55552b4de4c76p-4, -0x1.50aa78415c4bfp-59, 0x1.2fafbf95a1a8p-115}, /* u12 decomposed into triple-double*/
        {0x1.745d171352e2dp-4, -0x1.b9a4b0517c2c6p-58, -0x1.6128b0932efc8p-112}, /* u11 decomposed into triple-double*/
        {-0x1.999999996fed4p-4, 0x1.8ec6a7a38be73p-58, -0x1.85507f01f5868p-112}, /* u10 decomposed into triple-double*/
        {0x1.c71c71c71c59ap-4, 0x1.1e1945381ebebp-58, -0x1.b30f3065fce38p-112}, /* u9 decomposed into triple-double*/
        {-0x1.fffffffffffffp-4, -0x1.89d824c2891b7p-58, 0x1.b4f656bbfa58p-115}, /* u8 decomposed into triple-double*/
        {0x1.2492492492492p-3, 0x1.2438b8708dbd8p-57, 0x1.e00436fa1c45p-113}, /* u7 decomposed into triple-double*/
        {-0x1.5555555555555p-3, -0x1.5555440ddf7efp-57, 0x1.0fce30dd1d9p-116}, /* u6 decomposed into triple-double*/
        {0x1.999999999999ap-3, -0x1.9999999bc2648p-57, 0x1.4e3f5d1a3a7p-117}, /* u5 decomposed into triple-double*/
        {-0x1p-2, 0x1.4b03dac3ad4d9p-100, 0x1.78p-156}, /* u4 decomposed into triple-double*/
        {0x1.5555555555555p-2, 0x1.5555555555555p-56, 0x1.219f0c0c44756p-110}, /* u3 decomposed into triple-double*/
        {-0x1p-1, 0x1.9745a799cp-127, 0x0p+0}, /* u2 decomposed into triple-double*/
        {0x1p+0, -0x1.09f14p-143, 0x0p+0}, /* u1 decomposed into triple-double*/
    };

   if ((x> 0x1.fc65aa1908a66p-1) && (x<1)) {
```

```
        double hr,lr ;
        /*Add x by -1 result hr,lr*/
        Add112(-1.0,x,&hr,&lr);

       double hf = U[0][0];
        double mf = U[0][1];
        double lf = U[0][2];

        double h,m,l;
        /*Multiply (hf,mf,lf) by r result (h,m,l)*/
        Mul233(hr,lr,hf,mf,lf,&h,&m,&l);
        /*printf("i = 0, h = %la, m = %la, l = %la \n",h,m,l);*/
        double hfi,mfi,lfi,h1,m1,l1;

        for (int i = 1;i<14;i++){
            hfi = U[i][0];
            mfi = U[i][1];
            lfi = U[i][2];
            /*Add (hfi,mfi,lfi) by (h,m,l) result (h1,m1,l1)*/
            Add333(hfi,mfi,lfi,h,m,l,&h1,&m1,&l1);

            /*multiply (hr,lr) by (h1,m1,l1) result (h,m,l)*/
            Mul233(hr,lr,h1,m1,l1,&h,&m,&l);

       }

       return h+m;
    }

    if ((x> 1) && (x <= 0x1.0178e5916f543p+0)) {
        double hr,lr ;
        /*Add x by -1 result hr,lr*/
     Add112(x,-1.0,&hr,&lr);

        double hf = U[0][0];
        double mf = U[0][1];
        double lf = U[0][2];

        double h,m,l;
        /*Multiply (hf,mf,lf) by r result (h,m,l)*/
        Mul233(hr,lr,hf,mf,lf,&h,&m,&l);

        double hfi,mfi,lfi,h1,m1,l1;

        for (int i = 1;i<14;i++){
            hfi = U[i][0];
            mfi = U[i][1];
            lfi = U[i][2];
            /*Add (hfi,mfi,lfi) by (h,m,l) result (h1,m1,l1)*/
            Add333(hfi,mfi,lfi,h,m,l,&h1,&m1,&l1);

            /*multiply (hr,lr) by (h1,m1,l1) result (h,m,l)*/
            Mul233(hr,lr,h1,m1,l1,&h,&m,&l);

        }


        return h+m;
    }
```

```
    double m1;
    /*If x is a subnormal */
    if ((s ==0) && (e == 0) && (m !=0)) {
        uint64_t v = m;
        e = e - 1023;
        v = v*2;
        while (v < 0x10000000000000) {
            v *= 2;
            e--;
        }

        m1 =v *0x1p-52;

        u u;
        u.x = m1;
        m = u.i & 0xFFFFFFFFFFFFF;
    }
    /*If x is normal */
    else {
        m1= 1 + m*0x1p-52;

        e = e - 1023;
    }

    /*We shift by 44 bits for to get the first 8 bits.*/
    uint64_t i = m>>44;


    /*k=8
    halpha_i,lalpha_i is the double,double of 2^k/(2^k+i).*/
    double halpha_i = table_alpha_i_triple[(int)i][0];
    double malpha_i = table_alpha_i_triple[(int)i][1];
    double lalpha_i = table_alpha_i_triple[(int)i][2];

      double hr,mr,lr;
    /*Multiply (halpha_i, malpha_i,lalpha_i_m) by m1 result (hr, mr, lr) */
    Mul133(m1,halpha_i, malpha_i, lalpha_i,&hr,&mr,&lr);

    double h,mprime,l;
    /*Add (hr,lr) by (-1) result (h,m,l)*/
    Add133(-1,hr,mr,lr,&h,&mprime,&l);

    double hf = U[0][0];
    double mf = U[0][1];
    double lf = U[0][2];
    double h1, m1prime, l1;
    /*Multiply (hf,mf,lf) by (h,m,l) result (h1,m1,l1)*/
    Mul333(h,mprime,l,hf,mf,lf,&h1,&m1prime,&l1);

    double hfi,mfi,lfi,h2,m2,l2;

    for (int i = 1;i<14;i++){
        hfi = U[i][0];
        mfi = U[i][1];
        lfi = U[i][2];
        /*Add (hfi,mfi,lfi) by (h1,m1,l1) result (h2,m2,l2)*/
        Add333(hfi,mfi,lfi,h1,m1prime,l1,&h2,&m2,&l2);

        /*multiply (h,m,l) by (h2,m2,l2) result (h1,m1,l1)*/
```

```
        Mul333(h,mprime,l,h2,m2,l2,&h1,&m1prime,&l1);

    }
    /*k=8
    hlog_alpha_i,llog_alpha_i is the double,double of log(2^k/(2^k+i)).*/
    double hlog_alpha_i = table_log_alpha_i_triple[(int)i][0];
    double mlog_alpha_i = table_log_alpha_i_triple[(int)i][1];
    double llog_alpha_i = table_log_alpha_i_triple[(int)i][2];


    double h5,m5,l5;
    /*Add (h1,m1,l1) by (hlog_alpha_i,mlog_alpha_i,llog_alpha_i) result (h5,m5,l5)*/
    Add333(hlog_alpha_i,mlog_alpha_i,llog_alpha_i,h1,m1prime,l1,&h5,&m5,&l5);

    double h_log2 =0x1.62e42fefa39efp-1;
    double m_log2 =0x1.abc9e3b39803fp-56;
    double l_log2 = 0x1.7b57a079a1934p-111;
    double e_hlog2,e_mlog2,e_llog2;
    /* e*log(2) in triple-double precision.*/
    Mul133(e, h_log2, m_log2, l_log2, &e_hlog2, &e_mlog2, &e_llog2);

    double m6,l6;

    /*adding (e_hlog2,e_mlog2,e_llog2) by (h5,m5,l5) result (h6,m6,l6)*/
    Add333(e_hlog2, e_mlog2, e_llog2, h5, m5, l5, &h6, &m6, &l6);



    ml = m6+l6;



    double m62 = 2*m6;
   if ( (fabs(m6) == ulp_0_5(h6)) && m6>0){
        if (l6>0){
            ml = nextafter(m6,m62);
        }
        if (l6<0){
             ml= nextafter(m6,0);
        }
    }
    if ( (fabs(m6) == ulp_0_5(h6)) && (m6<0)) {
        if (l6<0){
            ml = nextafter(m6,m62);
        }
        if (l6>0){
            ml= nextafter(m6,0);
        }
    }
    return h6+ml;
}
```

log

Algorithm 15 Algorithm log

Condition: x is a **double** number.

Condition: $err_{fast} = 0x1.6b6b11ea279ecp - 59$ and $err_{accurate} = 0x1.810c9a86fc45ep - 99$

Output: $\log(x)$ in **double** number.

$y, h_6, l_6 = cr\log_{fast}(x)$
$left = h6 + FMA(-errfast, h6, l6)$
$right = h6 + FMA(+errfast, h6, l6)$
$ifright == left$
  return y
$y, h_6, l_6 = cr\log_{accurate}(x)$
$right = h6 + FMA(+errfast, h6, l6)$
$ifright == left$
  return y
$y = cr\log_{advanced}(x)$ return y

```c
/* x is double*/
double cr_log(double x){
    double h6 = 0;
    double l6 = 0;
    double right,left;
    double y;
    double errfast = 0x1.6b6b11ea279ecp-59;
    /* We started with the variable errfast = 0x1.01c7d6c404f05p-63 according to
    document (CR-LIBM A library of correctly rounded elementary functions in double-
        precision : 2006).
    we had errors in all 4 rounding modes.
    We increased the power by 1 until we found only one error in the RNDN rounding mode
    and no other errors in the other rounding modes. After, we had err = 0x1p-64.
    We searched by dichotomy after the dot. If we had errors on the equality between
        cr_log_fast(x) and mpfr_log(x)
     we increase either we decrease until we have 2 consecutive digits.
    We started again until we optimize errfast and also erracc.
    */

    y = cr_log_fast_path(x,&h6,&l6);
    left = h6 + __builtin_fma (-errfast, h6, l6);
    right = h6 + __builtin_fma (+errfast, h6, l6);

    if (right== left){

        return y;
    }
    double erracc = 0x1.810c9a86fc45ep-99;
    /* We started with the variable erracc = 0x1p-102, Result obtained by calculating with
        sage.
```

```
    We do as we did with errfast*/
    y = cr_log_accurate_path(x,&h6,&l6);
    left = h6 + __builtin_fma (-erracc, h6, l6);
    right = h6 + __builtin_fma (+erracc, h6, l6);

    if (right== left){

        return y;
    }

    y = cr_log_accurate_advanced_path(x);
    return y;

/* log(x) in double*/
}
```

Bibliography

[1] Mohammed Belaid. *Résolution de contraintes sur les flottants dédiée à la vérification de programmes*. PhD thesis, Université Nice Sophia Antipolis, 2013.

[2] Sylvie Boldo and Marc Daumas. Properties of the subtraction valid for any floating point system. In *7th International Workshop on Formal Methods for Industrial Critical Systems*, pages 137–149, 2002.

[3] Sylvain Chevillard, Christoph Lauter, and Mioara Joldes. Users' manual for the sollya tool. 2013.

[4] Catherine Daramy-Loirat, David Defour, Florent De Dinechin, Matthieu Gallet, Nicolas Gast, Christoph Quirin Lauter, and Jean-Michel Muller. Cr-libm a library of correctly rounded elementary functions in double-precision. Technical report, Technical report, LIP, ENS Lyon, 2006. URL: https://hal-ens-lyon. archives . . . , 2009.

[5] Florent De Dinechin, Christoph Lauter, and Jean-Michel Muller. Fast and correctly rounded logarithms in double-precision. *RAIRO-Theoretical Informatics and Applications*, 41(1):85–102, 2007.

[6] Theodorus Jozef Dekker. A floating-point technique for extending the available precision. *Numerische Mathematik*, 18(3):224–242, 1971.

[7] Nicolas Fabiano, Jean-Michel Muller, and Joris Picot. Algorithms for triple-word arithmetic. *IEEE Transactions on Computers*, 68(11):1573–1583, 2019.

[8] Shmuel Gal. Computing elementary functions: A new approach for achieving high accuracy and good performance. In *Accurate scientific computations*, pages 1–16. Springer, 1986.

[9] Stef Graillat and Fabienne Jézéquel. Tight interval inclusions with compensated algorithms. *IEEE Transactions on Computers*, 69(12):1774–1783, 2019.

[10] Vincenzo Innocente and Paul Zimmermann. Accuracy of mathematical functions in single, double, double extended, and quadruple precision. 2021.

[11] Claude-Pierre Jeannerod, Jean-Michel Muller, and Paul Zimmermann. On various ways to split a floating-point number. In *2018 IEEE 25th Symposium on Computer Arithmetic (ARITH)*, pages 53–60. IEEE, 2018.

[12] Christoph Quirin Lauter. *Basic building blocks for a triple-double intermediate format*. PhD thesis, INRIA, LIP, 2005.

[13] Julien Le Maire, Nicolas Brunie, Florent De Dinechin, and Jean-Michel Muller. Computing floating-point logarithms with fixed-point operations. In *2016 IEEE 23nd Symposium on Computer Arithmetic (ARITH)*, pages 156–163. IEEE, 2016.

[14] Vincent Lefèvre and Paul Zimmermann. *Arithmétique flottante*. PhD thesis, INRIA, 2004.

[15] Nicolas Louvet. *Algorithmes compensés en arithmétique flottante: précision, validation, performances*. PhD thesis, Université de Perpignan Via Domitia, 2007.

[16] Peter Markstein. The new ieee-754 standard for floating point arithmetic. In *Dagstuhl Seminar Proceedings*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2008.

[17] Jean-Michel Muller. Elementary functions.

[18] Jean-Michel Muller. *On the definition of ulp (x)*. PhD thesis, INRIA, LIP, 2005.

[19] Jean-Michel Muller, Nicolas Brisebarre, Florent de Dinechin, Claude-Pierre Jeannerod, Vincent Lefèvre, Guillaume Melquiond, Nathalie Revol, Damien Stehlé, and Serge Torres. Handbook of floating-point arithmetic, 2010.

[20] Deevashwer Rathee, Anwesh Bhattacharya, Rahul Sharma, Divya Gupta, Nishanth Chandran, and Aseem Rastogi. Secfloat: Accurate floating-point meets secure 2-party computation (full version). Technical report, Cryptology ePrint Archive, Report 2022/322, 2022, https://ia. cr/2022/322.

[21] Wikipedia contributors. Ieee 754 — Wikipedia, the free encyclopedia. https://en.wikipedia.org/w/index.php?title=IEEE_754&oldid=1096894205, 2022. [Online; accessed 1-September-2022].