Please check our wiki for help on navigating the form.

Horizon 2020

Excellent Science

Call: ERC-2020-ADG

(Call for proposal for ERC Advanced Grant)

Topic: ERC-2020-ADG

Type of action: ERC-ADG

(Advanced Grant)

Proposal number: 101019156

Proposal acronym: CORE-MATH

Deadline Id: ERC-2020-ADG

Table of contents

How to fill in the forms

The administrative forms must be filled in for each proposal using the templates available in the submission system. Some data fields in the administrative forms are pre-filled based on the steps in the submission wizard.

1 - General information

Topic	ERC-2020-ADG	Type of Action	ERC-ADG
Call Identifier	ERC-2020-ADG	Deadline Id	ERC-2020-ADG

Acronym	CORE-MATH
Proposal title	Ensuring Correctly Rounded Mathematical Functions

Note that for technical reasons, the following characters are not accepted in the Proposal Title and will be removed: < > " &

Duration in months	*60*	
Primary ERC Review Panel*	PE6 - Computer Science and Informatics	
Secondary ERC Review Panel		(if applicable)
ERC Keyword 1*	PE6_12 Scientific computing, simulation and modelling tools	

Please select, if applicable, the ERC keyword(s) that best characterise the subject of your proposal in order of priority.

ERC Keyword 2	*PE1_17 Numerical analysis*
ERC Keyword 3	*PE1_18 Scientific computing and data processing*
ERC Keyword 4	*Not applicable*

Free keywords	*floating-point number, IEEE 754 standard, correct rounding, mathematical library*

Proposal ID **101019156** *Acronym* **CORE-MATH**

*Abstract**

In 1985, the IEEE 754 standard defined for the first time what the result of a computation on floating-point numbers should be. Today, any program using floating-point additions, subtractions, multiplications and divisions yields bit-to-bit identical results, whatever the hardware, compiler or operating system. This is because IEEE 754 requires the best possible result for these operations, called correct rounding.

However, most scientific or industrial applications, like the Large Hadron Collider software, developed by thousands of physicists and engineers over two decades, or Karplus' equation J(φ) = A cos^2 φ + B cos φ + C in nuclear magnetic resonance spectroscopy, also require the evaluation of various mathematical functions: sine, exponential, logarithm, etc. These functions are provided by a mathematical library, which does not always provide correct rounding. As a resulting effect, a program using such mathematical functions might yield wrong results, which could have disastrous consequences. Moreover, these results might differ depending on the mathematical library, hardware, compiler or operating system.

We strongly believe it is the right time to fix that numerical reproducibility issue once and for all. CORE-MATH will provide new numerical algorithms to evaluate mathematical functions, which will always yield correct rounding (i.e., the best possible result) with speed comparable to the best libraries currently available. This will require clever algorithms to identify the most difficult cases for correct rounding, and innovative research in the field of the evaluation of mathematical functions.

Thanks to CORE-MATH, scientists, engineers, researchers will obtain correct results and thus bit-to-bit reproducible results for their numerical computations, with the same efficiency as with currently available mathematical libraries, or even more.

Remaining characters 113

In order to best review your application, do you agree that the above non-confidential proposal title and abstract can be used, without disclosing your identity, when contacting potential reviewers?* ● Yes ○ No

Declarations

In case of a Synergy grant application 'Principal Investigator' means 'corresponding Principal Investigator on behalf of all Principal Investigators', and 'Host Institution' means 'corresponding Host Institution'.

1) The Principal Investigator declares to have the written consent of all participants on their involvement and on the content of this proposal, as well as of any researcher mentioned in the proposal as participating in the project (either as other PI, team member or collaborator). The ERCEA may request the applicants to provide the written consent of all participants at any time during the evaluation process.*	☒
2) The Principal Investigator declares that the information contained in this proposal is correct and complete.	☒
3) The Principal Investigator declares that all parts of this proposal comply with ethical principles (including the highest standards of research integrity as set out, for instance, in the European Code of Conduct for Research Integrity and including, in particular, avoiding fabrication, falsification, plagiarism or other research misconduct).	☒

4) The Principal Investigator hereby declares that *(please select one of the three options below)*:

-- in case of multiple participants in the proposal, the Host Institution has carried out the self-check of the financial capacity of the organisation on http://ec.europa.eu/research/participants/docs/h2020-funding-guide/grants/applying-for-funding/register-an-organisation/financial-capacity-check_en.htm or to be covered by a financial viability check in an EU project for the last closed financial year.Where the result was “weak” or “insufficient”, the Host Institution confirms being aware of the measures that may be imposed in accordance with the H2020 Grants Manual (Chapter on Financial capacity check).	○
- in case of multiple participants in the proposal, the Host Institution is exempt from the financial capacity check being a public body including international organisations, higher or secondary education establishment or a legal entity, whose viability is guaranteed by a Member State or associated country, as defined in the H2020 Grants Manual (Chapter on Financial capacity check).	○
- in case of a sole participant in the proposal, the applicant is exempt from the financial capacity check.	●
5) The Principal Investigator hereby declares that each applicant has confirmed to have the financial and operational capacity to carry out the proposed action. Where the proposal is to be retained for EU funding, each beneficiary applicant will be required to present a formal declaration in this respect.	☒

The Principal Investigator is only responsible for the correctness of the information relating to his/her own organisation. Each applicant remains responsible for the correctness of the information related to him and declared above. Where the proposal to be retained for EU funding, the Host Institution and each beneficiary applicant will be required to present a formal declaration in this respect.

Note:

For **multi-beneficiary applications**, the coordinator vouches for its own organization and that all other participants confirmed their participation and compliance with conditions set out in the call. If the proposal is retained for funding, each participant will be required to submit a formal declaration of honour confirming this.

False statements or incorrect information may lead to administrative sanctions under the Financial Regulation 2018/1046.

Personal data will be collected, used and processed in accordance with Regulation 2018/1725 and the **Funding & Tenders Portal privacy statement**.

Please be however aware that, to protect EU financial interests, your data may be transferred to other EU institutions and bodies and be registered in the EDES database. Data in the EDES database is also subject to Regulation 2018/1725 and the EDES privacy statement.

2 - Participants & contacts

#	Participant Legal Name	Country	Action
1	INSTITUT NATIONAL DE RECHERCHE ENINFORMATIQUE ET AUTOMATIQUE	France	

2 - Administrative data of participating organisations

Host Institution

PIC	***Legal name***
999547074	*INSTITUT NATIONAL DE RECHERCHE ENINFORMATIQUE ET AUTOMATIQUE*

Short name: INRIA

Address

Street DOMAINE DE VOLUCEAU ROCQUENCOURT

Town LE CHESNAY CEDEX

Postcode 78153

Country France

Webpage www.inria.fr

Specific Legal Statuses

Legal person ....yes

Public body ....yes

Industry (private for profit)..................no

Non-profit ....yes

International organisation ....no

International organisation of European interest ....no

Secondary or Higher education establishment ....no

Research organisation ....yes

Enterprise Data

Based on the below details from the Beneficiary Registry the organisation is not an SME (small- and medium-sized enterprise) for the call.

SME self-declared status... unknown

SME self-assessment .... unknown

SME validation sme... unknown

Department(s) carrying out the proposed work

Department 1

Department name	Project-team Caramba, Inria Nancy - Grand Est	□ not applicable
	□ Same as proposing organisation's address	
Street	615 rue du jardin botanique	
Town	Villers-lès-Nancy	
Postcode	54600	
Country	France	

Principal Investigator

The following information of the Principal Investigator is used to personalise the communications to applicants and the evaluation reports. Please make sure that your personal information is accurate and please inform the ERC in case your e-mail address changes by using the call specific e-mail address:

For Advanced Grant Applicants: ERC-2020-AdG-applicants@ec.europa.eu

The name and e-mail of contact persons including the Principal Investigator, Host Institution contact are read-only in the administrative form, only additional details can be edited here. To give access rights and contact details of contact persons, please save and close this form, then go back to Step 4 of the submission wizard and save the changes.

ORCID	0000-0003-0718-4458		
Researcher ID		*The maximum length of the identifier is 11 characters (ZZZ-9999-2010) and the minimum length is 9 characters (A-1001-2010).*	
Other ID	*Please enter the type of ID here*	*Please enter the identifier number here*	
Last Name*	Zimmermann	Last Name at Birth	Zimmermann
First Name(s)*	Paul	Gender*	● Male ○ Female
Title	Dr.	Country of residence*	France
Nationality*	France	Country of Birth*	France
Date of Birth* (DD/MM/YYYY)	13/11/1964	Place of Birth*	Saint-Avold

Contact address

Current organisation name	Inria Nancy - Grand Est
Current Department/Faculty/Institute/ Laboratory name	Caramba Project-team

□ Same as organisation address

Street	615 rue du jardin botanique		
Postcode/Cedex	54600	Town*	Villers-les-Nancy
Phone	+33 (0)3 83 59 30 41	Country*	France
Phone2 / Mobile	+33 (0)6 66 47 22 15		
E-mail*	paul.zimmermann@inria.fr		

Qualifications

Earliest award (PhD, Doctorate)	Date of award (DD/MM/YYYY)	06/03/1991

Contact address of the Host Institution and contact person

The name and e-mail of Host Institution contact persons are read-only in the administrative form, only additional details can be edited here. To give access rights and contact details of Host Institution, please save and close this form, then go back to Step 4 of the submission wizard and save the changes. Please note that the submission is blocked without a contact person and e-mail address for the Host Institution.

Organisation Legal Name **INSTITUT NATIONAL DE RECHERCHE ENINFORMATIQUE ET AUTOMATIQUE**

First name* **Fabienne** Last name* **Elbar**

E-Mail* **polecaf-nancy@inria.fr**

Position in org. *European Project Manager*

Department *Transfert And Innovation* □ Same as organisation

□ Same as organisation address

Street 615 rue du Jardin Botanique

Town Villers les Nancy Postcode 54600

Country France

Phone +33 3 54 95 84 50 Phone2/Mobile +xxx xxxxxxxxx

Other contact persons

First Name	***Last Name***	***E-mail***	***Phone***
Rosa	Bernal-Carrera	zoila-rosa.bernal-carrera@inria.fr	*+33 7 61 20 01 88*

3 - Budget

Beneficiary Short Name	Direct costs														A. Total Direct Costs	B. Indirect Costs	C1. Subcontracting Costs	C2. Costs of in kind contributions not used on the beneficiary's premises	Total Estimated Eligible Costs	Requested EU contribution
	Personnel						Other direct costs							A.3 Internally invoiced goods and services						
	PI	Senior Staff	Postdocs	Students	Other Personnel costs	A.1. Total direct costs for personnel	Travel	Equipment - including major equipment	Other goods and services				A.2. Total Other Direct Costs							
									Consumables incl. fieldwork and animal costs	Publications (incl. Open Access fees) and dissemination	Other additional direct costs	Total other goods and services								
Institut National De Recherche Eninformatique Et Automatique	412781	570000	288000	333000	0	1603781.00	112500	0	0	0	180000	180000.00	292500.00	0	1896281.00	474070.25	0	0	2370351.25	2370351.25
Total	412781	570000	288000	333000	0	1603781.00	112500	0	0	0	180000	180000.00	292500.00	0	1896281.00	474070.25	0	0	2370351.25	2370351.25

Section C. Resources (Maximum 8000 characters allowed)

The Research Environment.
The PI is hired by Inria, the French National Institute for Research in Mathematics and Informatics. His research group is located in the “Nancy-Grand Est” research centre (22 research teams) and maintains long-term collaborations with several world-class groups in computer arithmetic. These research groups will contribute to the work of CORE-MATH as external collaborators. Connections with other Inria teams in France already exist, notably with the research team of Jean-Michel Muller (located at the Lyon Inria center) specializing in computer arithmetic.

The CORE-MATH Research Team is composed of the PI, a confirmed researcher to be hired by the project, together with three PhD students, and three postdoctoral researchers:

- the Principal Investigator (412,781 euros), with a 65% commitment to CORE-MATH. He will lead the project, hire the members of the CORE-MATH team, and guide the research of the PhD students, of the postdoctoral researchers, and more generally of all members of the team. On the technical side, he will work mainly on the search for HR-cases (RT1-a, RT2-a, RT3-a), in collaboration with PhD 1, PhD 2, and PhD 3;
- a Confirmed Researcher with 4-8 years of experience (570,000 euros) who will work mainly on the Validation Track, and collaborate with Postdoc 1, Postdoc 2 and Postdoc 3 for the design of correct rounding algorithms (RT1-b, RT2-b, RT3-b). She/he will be responsible of the dissemination of CORE-MATH through the GNU libc, and of the development of Meta-MPFR. She/he will be hired for 5 years;
- one PhD student (PhD 1) for 3 years (111,000 euros). PhD 1 will work on the search of HR-cases for bivariate functions (RT1-a), in collaboration with Postdoc 1 and the PI;
- one postdoctoral researcher (Postdoc 1) for 2 years (96,000 euros). Postdoc 1 will work on correctly-rounded algorithms for single precision bivariate functions (RT1-b), in collaboration with PhD 1 and the CORE-MATH Confirmed Researcher;
- one PhD student (PhD 2) for 3 years (111,000 euros). PhD 2 will work on the search for HR-cases of periodic functions for large arguments (RT2-a), in collaboration with Postdoc 2 and the PI;
- one postdoctoral researcher (Postdoc 2) for 2 years (96,000 euros). Postdoc 2 will work on correctly-rounded algorithms for double precision periodic functions (RT2-b), in collaboration with PhD 2 and the CORE-MATH Confirmed Researcher;
- one PhD student (PhD 3) for 3 years (111,000 euros). PhD 3 will work on the search for HR-cases in quadruple precision (RT3-a), in collaboration with Postdoc 2 and the PI;
- one postdoctoral researcher (Postdoc 3) for 2 years (96,000 euros). Postdoc 3 will work on correctly-rounded algorithms for quadruple precision (RT3-b), in collaboration with PhD 3 and the CORE-MATH Confirmed Researcher;
- other people already hired by Inria will also contribute partly to the CORE-MATH team and as such are part of the CORE-MATH team: Emmanuel Thomé (Inria Nancy), Pierrick Gaudry (Inria Nancy) and Stéphane Glondu (Inria Nancy) will contribute to the Validation Track, Jean-Michel Muller (Inria Lyon), Vincent Lefèvre (Inria Lyon) and Claude-Pierre Jeannerod (Inria Lyon) will contribute to the search for HR-cases (RT1-a, RT2-a, RT3-a), Andreas Enge (Inria Bordeaux) and Fredrik Johansson (Inria Bordeaux) will contribute to the Validation Track.

Visitors and External Collaborators (70,000 euros).
Several researchers (4 months per year in total) will be regular visitors of the CORE-MATH team. In particular we expect to invite David Bailey (Lawrence Berkeley Laboratory, USA) to work on Research Track RT3-b, John Gustafson (National University of Singapore) to work on Research Track RT1-b, Siegfried Rump (Hamburg University of Technology, Germany) and Norbert Müller (University of Trier, Germany) to work on the Validation Track, and Alexander Godunov (Russian Academy of Sciences, Moscow) to work on Research Track RT2-b. We expect the following external collaborators will contribute to CORE-MATH: Christoph Lauter (currently at University of Alaska Anchorage, Alaska) will work on Research Track RT2-b, and Patrick Pélissier (Toulouse, France) will work on the Validation Track.

CORE-MATH Workshops (80,000 euros).
Two CORE-MATH workshops will be organized, one in Year 2, and one in Year 4. A natural place for these workshops is Schloss Dagstuhl (Leibniz Center for Informatics, Germany), which is an exceptional place for scientific exchanges. Each workshop will bring together about 20 researchers from the field of computer arithmetic and developers from numerical libraries. The first workshop will focus on the search for HR-cases, while the second workshop will focus on the evaluation of mathematical functions, and will celebrate the CORE-MATH bug bounties (on single, double and quadruple precision).

CORE-MATH Meetings (30,000 euros).

In between the CORE-MATH workshops, at Years 1, 3, and 5, three smaller meetings will be organized with the members of the CORE-MATH Research Team and up to 5 invited researchers.

Travel and conferences (112,500 euros).
Each member of the CORE-MATH team is expected to attend one international conference per year.
This sums up to 25 international conferences for the whole project. Main target conferences are the IEEE Symposium on Computer Arithmetic (ARITH), the International Congress on Mathematical Software (ICMS), the Algorithmic Number Theory Symposium (ANTS).

In addition, the PI and the CORE-MATH Researcher will perform a total of five one-month visits in the Silicon Valley, to present the CORE-MATH results and convince the next IEEE 754 revision committee to require correctly rounded mathematical functions.

Available Resources.
The Caramba team has access to the Grid5000 platform, a French scientific instrument supporting experiment-driven research in all areas of computer science, including high performance computing, distributed computing, networking and big data, with more than 15000 high-performance cores available. The use of this platform will be free of charge for the CORE-MATH team.

Remaining characters 1876

4 - Ethics

1. HUMAN EMBRYOS/FOETUSES		Page
Does your research involve Human Embryonic Stem Cells (hESCs)?	○ Yes ● No	
Does your research involve the use of human embryos?	○ Yes ● No	
Does your research involve the use of human foetal tissues / cells?	○ Yes ● No	
2. HUMANS		Page
Does your research involve human participants?	○ Yes ● No	
Does your research involve physical interventions on the study participants?	○ Yes ● No	
3. HUMAN CELLS / TISSUES		Page
Does your research involve human cells or tissues (other than from Human Embryos/ Foetuses, i.e. section 1)?	○ Yes ● No	
4. PERSONAL DATA		Page
Does your research involve personal data collection and/or processing?	○ Yes ● No	
Does your research involve further processing of previously collected personal data (secondary use)?	○ Yes ● No	
5. ANIMALS		Page
Does your research involve animals?	○ Yes ● No	
6. THIRD COUNTRIES		Page
In case non-EU countries are involved, do the research related activities undertaken in these countries raise potential ethics issues?	○ Yes ● No	
Do you plan to use local resources (e.g. animal and/or human tissue samples, genetic material, live animals, human remains, materials of historical value, endangered fauna or flora samples, etc.)?	○ Yes ● No	
Do you plan to import any material - including personal data - from non-EU countries into the EU?	○ Yes ● No	
Do you plan to export any material - including personal data - from the EU to non-EU countries?	○ Yes ● No	
In case your research involves low and/or lower middle income countries, are any benefits-sharing actions planned?	○ Yes ● No	
Could the situation in the country put the individuals taking part in the research at risk?	○ Yes ● No	

7. ENVIRONMENT & HEALTH and SAFETY		Page
Does your research involve the use of elements that may cause harm to the environment, to animals or plants?	○ Yes ● No	
Does your research deal with endangered fauna and/or flora and/or protected areas?	○ Yes ● No	
Does your research involve the use of elements that may cause harm to humans, including research staff?	○ Yes ● No	
8. DUAL USE		Page
Does your research involve dual-use items in the sense of Regulation 428/2009, or other items for which an authorisation is required?	○ Yes ● No	
9. EXCLUSIVE FOCUS ON CIVIL APPLICATIONS		Page
Could your research raise concerns regarding the exclusive focus on civil applications?	○ Yes ● No	
10. MISUSE		Page
Does your research have the potential for misuse of research results?	○ Yes ● No	
11. OTHER ETHICS ISSUES		Page
Are there any other ethics issues that should be taken into consideration? Please specify	○ Yes ● No	

I confirm that I have taken into account all ethics issues described above and that, if any ethics issues apply, I will complete the ethics self-assessment and attach the required documents. ☒

How to Complete your Ethics Self-Assessment

5 - Call specific questions

Please indicate your percentage of working time in an EU Member State or Associated Country over the period of the grant: Please note that you are expected to spend a minimum of 50% of your total working time in an EU Member State or Associated Country.	90
Please indicate the % of working time the PI dedicates to the project over the period of the grant. Please note that the PI is expected to dedicate a minimum of working time to the project (30% for AdG, 40% for CoG and 50% for StG). The personnel cost for the PI provided in section “3-Budget” cannot be higher than the percentage indicated here. This information will be provided to the experts at Step 2 together with the section “3-Budget”.	65
I acknowledge that I am aware of the eligibility requirements for applying for this ERC call as specified in the ERC Annual Work Programme, and certify that, to the best of my knowledge my application is in compliance with all these requirements. I understand that my proposal may be declared ineligible at any point during the evaluation or granting process if it is found not to be compliant with these eligibility criteria.*	☒

Data-Related Questions and Data Protection (Consent to any question below is entirely voluntary. A positive or negative answer will not affect the evaluation of your project proposal in any form and will not be communicated to the evaluators of your project.)

For communication purposes only, the ERC asks for your permission to publish, in whatever form and medium, your name, the proposal title, the proposal acronym, the panel, and host institution, should your proposal be retained for funding.	● Yes ○ No
Some national and regional public research funding authorities run schemes to fund ERC applicants that score highly in the ERC's evaluation but which can not be funded by the ERC due to its limited budget. In case your proposal could not be selected for funding by the ERC do you consent to allow the ERC to disclose the results of your evaluation (score and ranking range) together with your name, non-confidential proposal title and abstract, proposal acronym, host institution and your contact details to such authorities? This consent is entirely voluntary and refusal to give it will in no way affect the evaluation of your proposal.	● Yes ○ No
The ERC is sometimes contacted for lists of ERC funded researchers by institutions that are awarding prizes to excellent researchers. Do you consent to allow the ERC to disclose your name, non-confidential proposal title and abstract, proposal acronym, host institution and your contact details to such institutions? This consent is entirely voluntary and refusal to give it will in no way affect the evaluation of your proposal.	● Yes ○ No
The European Research Council Executive Agency (ERCEA) occasionally contacts Principal Investigators of funded proposals for various purposes such as communication campaigns, pitching events, presentation of their project's evolution or outcomes to the public, invitations to represent the ERC in national and international forums, studies etc. Should your proposal be funded, do you consent to the ERCEA staff contacting you for such purposes?	● Yes ○ No

For purposes related to monitoring, study and evaluating implementation of ERC actions, the ERC may need that submitted proposals and their respective evaluation data be processed by external parties. Any processing will be conducted in compliance with the requirements of Regulation (EU) 2018/1725.

Have you previously submitted a proposal to the ERC? If known, please specify your most recent ERC application details.	● Yes ○ No

Proposal number 669971

Other details	N/C

Excluded Reviewers

You can provide up to three names of persons that should not act as an evaluator in the evaluation of the proposal for potential competitive reasons.

First Name	
Last Name	
Institution	
Town	
Country	
Webpage	

Extended Open Research Data Pilot in Horizon 2020

If selected, all applicants will by default participate in the Pilot on Open Research Data in Horizon 2020[1] , which aims to improve and maximise access to and re-use of research data generated by actions.

However, participation in the Pilot is flexible in the sense that it does not mean that all research data needs to be open. After the action has started, participants will formulate a Data Management Plan (DMP), which should address the relevant aspects of making data FAIR - findable, accessible, interoperable and re-usable, including what data the project will generate, whether and how it will be made accessible for verification and re-use, and how it will be curated and preserved. Through this DMP projects can define certain datasets to remain closed according to the principle "as open as possible, as closed as necessary". A Data Management Plan does **not** have to be submitted at the proposal stage.

Furthermore, applicants also have the possibility to opt out of this Pilot completely at any stage (before or after the grant signature), thereby freeing themselves retroactively from the associated obligations.

Please note that participation in this Pilot does not constitute part of the evaluation process. Proposals will not be penalised for opting out.

We wish to opt out of the Pilot on Open Research Data in Horizon 2020.	○ Yes	● No

[1] *According to article 43.2 of Regulation (EU) No 1290/2013 of the European Parliament and of the Council, of 11 December 2013, laying down the rules for participation and dissemination in "Horizon 2020 - the Framework Programme for Research and Innovation (2014-2020)" and repealing Regulation (EC) No 1906/2006.*

ERC Advanced Grant
Research Proposal (Part B1)

CORE-MATH: Ensuring Correctly Rounded Mathematical Functions

Principal Investigator (PI):	**Dr Paul Zimmermann**
PI's host institution:	**Inria, France**
Proposal full title:	**Ensuring Correctly Rounded Mathematical Functions**
Proposal short name:	**CORE-MATH**
Project duration:	**60 months**
Targeted Review Panel:	**PE6 (Computer Science and Informatics)**

Proposal Abstract

In 1985, the IEEE 754 standard defined for the first time what the result of a computation on floating-point numbers should be. Today, any program using floating-point additions, subtractions, multiplications and divisions yields bit-to-bit identical results, whatever the hardware, compiler, or operating system. This is because IEEE 754 requires the best possible result for these operations, called **correct rounding**.

However, most scientific or industrial applications, like the Large Hadron Collider software, developed by thousands of physicists and engineers over two decades, or Karplus' equation $J(\phi) = A\cos^2\phi + B\cos\phi + C$ in nuclear magnetic resonance spectroscopy, also require the evaluation of various **mathematical functions**: sine, exponential, logarithm, etc. These functions are provided by a **mathematical library**, which does not always provide correct rounding. As a resulting effect, a program using such mathematical functions might yield **wrong results**, which could have disastrous consequences [24]. Moreover, these results might **differ** depending on the mathematical library, hardware, compiler or operating system.

We strongly believe it is the right time to fix that numerical reproducibility issue **once and for all**. CORE-MATH will provide new numerical algorithms to evaluate mathematical functions, which will always yield **correct rounding** (i.e., the best possible result) with speed comparable to the best libraries currently available. This will require **clever algorithms** to identify the most difficult cases for correct rounding, and **innovative research** in the field of the evaluation of mathematical functions.

Thanks to CORE-MATH, scientists, engineers, researchers will obtain **correct results** and thus **bit-to-bit reproducible results** for their numerical computations, with the same efficiency as with currently available mathematical libraries, or even more.

Extended Synopsis of the Scientific Proposal

1 Motivation and State-of-the-Art

When programs manipulate numerical values, these values can only be approximated by floating-point numbers. Computations with floating-point numbers are by nature inexact. For example 1/3 cannot be represented exactly in binary. To overcome this issue, well-defined semantics for floating-point computations have been established by the IEEE 754 standard. The first version of this standard was published in 1985. Thanks to IEEE 754, and to its inventor William Kahan (Turing Award, 1989), *most people today would never expect different answers to the same mathematical calculation performed on different microprocessors,* as one can read in *Intel and Floating-Point* [14]. However, this holds for computations using only basic arithmetic operations, and in more complex programs like the Large Hadron Collider software (millions of lines of C++ and Python code), it has been noticed that changing the underlying mathematical library resulted in some collisions being missed or misidentified [1]. How can it be?

For addition, subtraction, multiplication, division, fused multiply-add, and square root, IEEE 754 requires correct rounding, i.e., the best possible result. For any other mathematical operation, for example sin, exp, log, IEEE 754 only *recommends* correct rounding. As a consequence, different mathematical libraries might give different results (and indeed they do) which explains the differences seen with the Large Hadron Collider software. The main reason is that authors of IEEE 754 were more concerned by speed of mathematical functions than accuracy, and research in computer arithmetic was not far enough to allow efficient correct rounding. Nowadays, the scientific community is more and more concerned by accuracy and reproducibility, and research in computer arithmetic has made great advances. Thus we propose to **require correct rounding of mathematical functions**, which will in turn ensure bit-to-bit reproducibility. The goal of CORE-MATH is to prove this is possible, with no loss of efficiency, and thus convince the next IEEE 754 revision committee to require correct rounding for mathematical functions.

Floating-Point Formats and Correct Rounding. The IEEE 754 standard defines three binary formats for numerical computations, `binary32`, `binary64`, and `binary128`:

IEEE 754 format	precision (bits)	$\lvert x\rvert_{\min}$	$\lvert x\rvert_{\max}$
`binary32` (single precision)	24	$1.4 \cdot 10^{-45}$	$3.4 \cdot 10^{38}$
`binary64` (double precision)	53	$4.9 \cdot 10^{-324}$	$1.8 \cdot 10^{308}$
`binary128` (quadruple precision)	113	$6.5 \cdot 10^{-4966}$	$1.2 \cdot 10^{4932}$

For these formats, IEEE 754 also defines rounding modes: to nearest, toward $-\infty$, toward $+\infty$, and toward zero. For a mathematical function f, a floating-point input x, the *correctly rounded* value of $f(x)$ is the floating-point number closest to the infinitely precise value $f(x)$ according to the given rounding mode. Thus the correctly rounded value is the best possible answer and it is unique, which ensures bit-to-bit reproducibility.

Previous Work and State-of-the-Art. The efficient design of correct rounding algorithms depends on two fundamental problems: the search for Hard-to-Round cases (HR-cases), and efficient algorithms for evaluating the target mathematical function.

The HR-cases are the floating-point numbers x in the target format (single, double, or quadruple precision) such that $f(x)$ is very close to a floating-point number in the same format (or to the middle of two floating-point numbers for rounding to nearest). **Knowing them is crucial to guarantee correct rounding, while in the same time having efficient algorithms.** However, determining them is a very hard scientific problem since the number of possible inputs is huge (up to 2^{128} for the `binary128` format). The first non-trivial algorithm to search for HR-cases is due to Lefèvre [17] (L-algorithm), and another more efficient algorithm (SLZ) was invented by Lefèvre, Stehlé and the PI [21, 22]. The SLZ algorithm was extensively studied in Serge Torres' PhD thesis [23], and rigorous estimates are given in [3] for the number of Hard-to-Round cases using tools from analytic number theory.

	GNU libc	Intel Math Library	AMD libm	Newlib	OpenLibm	Musl
asin	0.898	0.528	0.861	0.926	0.743	0.743
exp2	0.502	0.519	1.00	1.02	0.501	0.502
log2	0.752	0.508	0.586	1.65	0.865	0.752
sqrt	**0.500**	**0.500**	**0.500**	**0.500**	**0.500**	**0.500**

function	binary32	binary64	binary128
sin	71/79	306/299	3060/3361
exp	47/43	45/46	3546/3342
pow	82/76	115/108	9412/9027

Table 1: Top: maximal error in units in last place for some mathematical libraries in single precision [25]. Bottom: latency (in clock cycles) for some GNU libc 2.31 functions on an Intel Core i7-8750H (left) and an AMD Ryzen 5-2400G (right), for random inputs in $[-10, 10]$ for exp, in $[0, 10]^2$ for pow, and in $[2^{e-1}, 2^e]$ for sin, with $e = 128, 1024, 16384$ for `binary32`, `binary64` and `binary128` respectively.

For the evaluation of mathematical functions, the main difficulty is to design algorithms that fully exploit modern hardware (fused multiply-add, larger caches, faster integer operations). The main scientific breakthroughs are Gal's *accurate table* method [10], which enables one to generate very accurate tables, and the optimization of minimax polynomials [6], which yields quasi-optimal polynomials to evaluate a given function on a given interval. Lauter worked on a correct-rounding powering function in double precision [15]. This research field is quite active recently, with the generation of efficient code [4], the use of integer operations to obtain a correctly rounded `binary64` logarithm in 49 cycles on average [16], the use of clever lookup tables [9, 11], accurate Horner polynomials [20], or new approaches [12].

On the implementation side, Ziv was the first in 1991 to design correct rounding code for a few mathematical functions in double precision, within the `libultim` library [26]. Another correct rounding library, CRLIBM, was developed in 2004-2006 [7, 8]. These libraries are no longer maintained, and addressed some IEEE formats only; in particular they did not consider quadruple precision. Table 1 shows the maximal error in units in last place for some mathematical libraries in single precision, and the number of computer cycles needed by the reference GNU libc implementation (it is free, widely available, highly optimized, and well tested).

In conclusion of this state-of-the-art, while good progress has been made in the search for HR-cases and toward efficient evaluation algorithms, current mathematical libraries still do not provide correct rounding, at best they document some estimated errors bounds.

2 Grand Challenge and Research Tracks

The IEEE 754 standard only *recommends* mathematical functions with correct rounding. We strongly believe it is the right time to actually *require* correct rounding for these functions used in many scientific and industrial applications. The CORE-MATH Grand Challenge is thus:

Design correct rounding algorithms for mathematical functions, and corresponding IEEE 754 conforming implementations for single, double, and quadruple precision, with better efficiency than current non-conforming mathematical libraries.

To guarantee correct rounding for a function f and an input x, one usually uses the algorithm from Fig. 1, where $\mathrm{round}_p(y)$ means rounding y to precision p. The crucial point for the correctness of that algorithm is that $\mathrm{round}_p(y_2)$ always yields the correct rounding. If the error on y_2 is bounded by say ε_2, this means that all numbers in the interval $[y_2 - \varepsilon_2, y_2 + \varepsilon_2]$ should round to the same number in the target precision. For rounding toward zero for example, this implies the infinitely-precise value $f(x)$ is never closer than ε_2 from a floating-point number in the target

fast path	call a routine f_1, using a working precision $p_1 > p$, yielding an approximation y_1 such that $\lvert y_1 - f(x)\rvert < \varepsilon_1$;
rounding test	if $\mathrm{round}_p(y_1 - \varepsilon_1) = \mathrm{round}_p(y_1 + \varepsilon_1)$, return that number;
accurate path	otherwise call a routine f_2, using a working precision $p_2 > p_1$, yielding an approximation y_2, and return $\mathrm{round}_p(y_2)$.

Figure 1: The correct rounding algorithm for a univariate function $f(x)$ and target precision p.

format. It is thus crucial to be able to determine the HR-cases, to deduce the smallest possible precision p_2. Once the HR-cases are known, the required accuracy of the *accurate path* f_2 is known. It remains to choose the accuracy of the *fast path* f_1, and to efficiently implement f_1 and f_2. The search for HR-cases and the evaluation algorithms depend heavily on the target mathematical function and on the target precision: **Research Track 1** considers IEEE 754 single precision, **Research Track 2** double precision, and **Research Track 3** quadruple precision. Finally, the results of CORE-MATH will be disseminated via the **Validation Track**.

Research Track 1: Single Precision

Recall the HR-cases are the numbers x in the target format such that $f(x)$ is closest to a number y in that format (or to the middle of two consecutive numbers for rounding to nearest). Determining them is known as the Table Maker's Dilemma, where research has been active since more than 20 years [18]. For example, the worst case for the cube-root function in the `binary32` format and rounding to nearest is $x = 8606645 \cdot 2^{47}$: the cube-root of x differs from the middle of the two consecutive `binary32` numbers 10659771 and 10659772 by less than $2 \cdot 10^{-8}$. The HR-cases determine the working precision p_2 of the *accurate path* function f_2 (Fig. 1). Determining these HR-cases, or at least tight bounds for them, is thus a crucial step.

Determining the HR-cases for the IEEE `binary32` format (single precision) is nowadays an easy task for the univariate functions, since there are only up to 2^{32} different inputs to check, and this can be done by exhaustive search. However, bivariate functions like x^y, $\arctan(y/x)$, or $\sqrt{x^2 + y^2}$ are still out of reach by this approach, since they have up to 2^{64} different inputs. In order to overcome this difficulty, new algorithms for functions of two variables will be designed. **Criterion of Success for Research Track 1: new algorithms and a reference IEEE 754-conforming implementation for the power function in single precision within 50 cycles on average (compared to 76-82 cycles for the current non-conforming GNU libc implementation)[1].**

Research Track 2: Double Precision

For double precision (`binary64`), thanks to the work of Lefèvre and Muller [18], the HR-cases are known for several mathematical functions. One noticeable exception is the case of periodic functions like $\sin(x)$ for huge arguments, namely for x near 2^{1024}. A first non-naive algorithm to find HR-cases for such functions was proposed in [13]. The authors report a total time of 4 core-years to check the binade $[2^{1023}, 2^{1024}]$, thus it will take about 4000 core-years to check the whole `binary64` exponent range for $\sin(x)$. Our objective is to reduce this time to less than 400 core-years, using both algorithmic and implementation breakthroughs.

Once the HR-cases are known, one has to correctly round $\sin(x)$. A first step is called *argument reduction*: one first computes $x' = x - k\pi$ such that $|x'| \le \pi/2$. For large x, the subtraction $x - k\pi$ produces a huge cancellation, and therefore this operation has to be performed with a large precision (up to more than 1024 bits) to obtain a sufficiently accurate reduced value x'.

[1] While CORE-MATH will target **all functions** of Annex F from the C language standard, within each research track we identify hard problems that remain currently unsolved, and give corresponding success criteria.

CORE-MATH will design brand new algorithms for both argument reduction with huge inputs, and for the evaluation on the reduced argument.

Criterion of Success for Research Track 2: new algorithms and a reference IEEE 754-conforming implementation for the sine function within 100 cycles on average for the whole double precision exponent range (compared to about 300 cycles for the current non-conforming GNU libc implementation).

Research Track 3: Quadruple Precision

For non-algebraic `binary128` functions, no HR-cases are known. In [21] the current best approach, namely the SLZ-algorithm with parameters $d = 3$ and $\alpha = 2$, is said to take about 3 Gyears to find HR-cases of $\exp(x)$ for $1/2 \le x < 1$ (which reduces to about 420 Myears today). New research ideas will be needed, in order to make possible the search for HR-cases in quadruple precision.

The accuracy of f_1 (Fig. 1) is chosen as follows. If t_1 (resp. t_2) is the average time of f_1 (resp. f_2), and ξ the probability that f_1 is not able to round correctly, the average time is:

$$t = t_1 + \xi t_2.$$

Since t_2 is known, it suffices to try different implementations of f_1, measure their time t_1, their probability of failure ξ, and keep the one giving the smallest average time t.

The design of the functions f_1 and f_2 follows the same principles:

- first use *argument reduction* to reduce the input range to a small interval, say $[a, b]$;
- approximate the function f on the small interval $[a, b]$ using a polynomial approximation. Well known algorithms (for example Remez' algorithm) give the best polynomial for a given degree [19]. Remez' algorithm is implemented in the Sollya software tool [6].

What is critical is the efficiency of the arithmetic layer to implement the *fast path* f_1 and the *accurate path* f_2. Assume one wants to implement the function $f(x)$. After argument reduction, one can assume $|x| \le C/2^k$. Then one needs polynomial approximations of degree ℓ for some integer ℓ depending on the function, the size 2^k of the tables, the wanted accuracy. To determine the best values of the parameters $(k, \ell, ...)$, a research tool will be designed (Meta-MPFR, see below) that will automatically generate the corresponding functions f_1 and f_2.

Criterion of Success for Research Track 3: new algorithms and a reference IEEE 754-conforming implementation for the exponential function in quadruple precision within 200 cycles on average (compared to 3300-3500 cycles for the current non-conforming GNU libc implementation).

Validation Track

The fourth track is a validation track, and consists in the design and implementation of a mathematical library, which (i) will always provide correct rounding and (ii) will be faster on average than existing implementations (which do not guarantee correct rounding). This track will be based on the results from Research Tracks 1-3, and will give continuous feedback to them. All functions of Annex F from the C language standard will be considered.

An essential component of the validation track will be Meta-MPFR, a generator of optimized floating-point arithmetic. It will take as input the target function and format, the parameters for argument reduction, the degree of the approximation polynomials, the working precision, and automatically generate very efficient arithmetic operations (mainly addition and multiplication), on top of which the fast/accurate path functions f_1 and f_2 will be built (Fig. 1).

The C Floating Point Study Group working on the current revision of the C standard has reserved names for correct rounding functions, for example `cr_sin` for the correct rounding sine function [5]. Thus the timeline is excellent for CORE-MATH.

Criterion of Success for the Validation Track: ensure the correct rounding functions designed within CORE-MATH are integrated into at least one of the current mathematical libraries: GNU libc, Intel Math Library, etc.

3 Risk Assessment and Management

For Research Track 1, we are confident we will be able to determine the hard-to-round cases for x^y in the binary32 format, by inventing an algorithm similar to SLZ for bivariate functions, if needed with the help of parallel computations.

For Research Track 2, saving a factor of 10 over the state-of-the-art search for HR-cases [13] entails a high risk. If we only save a smaller factor, for example 3, the total time will still be reachable using distributed computations, for which the PI has a very solid experience [2].

The risk for Research Track 3 is very high. Indeed, the current estimation of 420 Myears for the HR-cases search is huge (for just one binade), and here a factor of about one million should be saved to make it feasible. If the algorithmic and implementation improvements are not sufficient, another research direction would be to add a second rounding test in Figure 1 after the call to f_2, to check whether $\mathrm{round}_p(y_2 - \varepsilon_2) = \mathrm{round}_p(y_2 + \varepsilon_2)$, where ε_2 is the maximal error for y_2. If that is not the case, a third function f_3 will be called with a larger precision $p_3 > p_2$. Since the cost of determining HR-cases with the SLZ algorithm decreases with the precision, p_3 will be chosen such that this search becomes possible.

The Validation Track will consolidate the results obtained by Research Tracks 1-3. The main risk for this track is that the output of CORE-MATH will not be adopted by the scientific community. To mitigate this risk, contributions will be made to the main mathematical libraries used in scientific applications very early during CORE-MATH.

4 Expected Scientific and Economic Impact

As main scientific impact of CORE-MATH, the *accuracy* of applications using mathematical functions will automatically improve, thanks to the correct rounding property. CORE-MATH will open new possibilities for engineers and researchers. Firstly, numerical applications will become *bit-to-bit reproducible*, across hardware processors, compilers, operating systems. Secondly, since the roundoff error for every mathematical function will be bounded, it will become possible to compute rigorous error bounds for a whole computation. In particular, it will be possible to perform rigorous interval arithmetic with mathematical functions.

The economic impact of CORE-MATH will be multiple. On the one hand, the cost of developing numerical applications will decrease, since it will no longer be required to test them on every different combination of hardware, compiler, operating system (or worse to tweak them so that a test suite runs) and we will get for free *forward reproducibility*, i.e., a program written at year Y will still yield the same results at year $Y + 10$. This is not the case currently, since any tiny change in the mathematical library (either improving or degrading the accuracy) might change the final result. We also expect it will enable to join or share the development efforts of the different mathematical libraries currently available (in particular GNU libc). Finally, *vendor lock-in* will no longer be possible, where the library designed by a vendor calls non-optimal routines on hardware from a different vendor.

5 Commitment of the PI and Research Group

The PI, who will dedicate 65% of his work time to CORE-MATH, is a member of the Caramba Research Group at Inria Nancy. Apart from the researchers who will be hired specifically to work on CORE-MATH, the PI will work in close collaboration with Jean-Michel Muller and Vincent Lefèvre at ENS Lyon. Jean-Michel Muller has been the leader of the computer arithmetic research group at ENS Lyon since 30 years, and Vincent Lefèvre is a member of this group who has made important contributions to the search for HR-cases, and to the implementation of correct rounding in the GNU MPFR library.

6 Curriculum Vitae

Personal Information. Paul Zimmermann, born 13/11/1964.
Email Paul.Zimmermann@inria.fr, home page https://members.loria.fr/PZimmermann/.

Education.
- 2001: Habilitation (highest French academic degree), Nancy, France.
- 1991: PhD in Computer Science, École Polytechnique, Palaiseau, France.
- 1988: Master in Computer Science, University Paris VII, France.
- 1987: Engineer from École Polytechnique (major French engineer school), Palaiseau, France.

Positions.
- 2019-present: Senior Research Director of "exceptional class" at Inria Nancy, France.
- 2008-2019: Senior Research Director at Inria Nancy, France.
- 1998-2008: Research Director (≈ Full Professor) at Inria Nancy, France.
- 1988-1998: Researcher at Inria (Rocquencourt near Paris until 1992, then Nancy).

Fellowships and Awards.
- 2012: Holder of "Prix La Recherche", France, for the record factorization of RSA-768.
- 2005: Winner of the Many Digits competition, Nijmegen, Netherlands.

Supervision of Graduate Students and Postdoctoral Fellows.
- 1994-present: supervised 5 PhD students (all as sole advisor). François Bertault now works for Facebook, Laurent Fousse for Google, and Damien Stehlé is Professor at ÉNS Lyon, France, and was PI of the LattAC (Lattices: Algorithms and Cryptography) ERC Starting Grant (2014-2018).

Teaching Activities. 1992-present: about 300 teaching hours at different levels (Master in Computer Science, engineering schools) and different topics (computer algebra, algorithmic number theory) in Paris and Nancy, France. In particular, the PI created a new course on Algorithmic Number Theory, Coding and Cryptography in the computer science Masters in Nancy (2000-2005), and in 2005-2006 he created a new course "Introduction to Cryptology" in this Masters.

Organization of Scientific Meetings. Co-organized a workshop on discrete tomography, Pont-à-Mousson, France, 1999; a workshop on open-source computer algebra in Lyon, France, 2002; the RNC'7 (Real Numbers and Computers) conference in Nancy, France, 2006; the Sage Days 10 and the CADO workshop on integer factorization in Nancy, France, 2008; and the Ninth Algorithmic Number Theory Symposium (ANTS-IX), Nancy, France, 2010. Organized the *Fast Algorithms* track at the workshop *Computing by the Numbers: Algorithms, Precision, and Complexity* for the 60th birthday of Richard Brent, Berlin, Germany, 2006.

Institutional Responsibilities and Research Leadership.
- 2013-2016: Head of Science of the Inria-Nancy research centre (21 research teams and 175 scientists).
- 2011-2014: Elected member of the Inria Scientific Board.
- 2011-2012: Head of a team of 8 engineers, Inria Nancy, France.
- 1999-2001 and 2005-2007: Elected member of the Inria Evaluation Committee.
- Since his arrival in Nancy in 1993, the PI was at the origin of several research teams on discrete mathematics and algorithmic number theory. Several full-time researchers were recruited in these teams, where more than twenty PhD or postdoctoral students were trained. The PI was in particular at the origin of a joint project involving several French research teams on

reliable computer arithmetic (1999-2000), and PI of a PRACE project (Partnership for Advanced Computing in Europe) which led to three world records in December 2019 and February 2020: the factorization of RSA-240, RSA-250, and the computation of a 240-digit discrete logarithm (see reference 9 in the ten-year track record).

Commissions of Trust.
- Member of the ARITH program committee in 2001 (ARITH'15), 2003 (ARITH'16), 2005 (ARITH'17), 2007 (ARITH'18) and 2009 (ARITH'19). ARITH is the major conference on computer arithmetic.
- Member of the program committee of AfricaCrypt in 2010, of ISSAC in 2013, of WAIFI in 2016, and of ANTS XIII in 2018.
- Program co-chair of the RNC'7 conference in Nancy, France, 2006.

International Recognition. Invited presentations (selection of) at:
- *Computational Number Theory* workshop at the *Foundations of Computer Mathematics* conference in Oxford, UK, 1999;
- SCAN conference in Paris, France, 2002 (main conference on interval arithmetic);
- PARI/GP workshop, Paris, France, 2004;
- IEEE 754 revision committee, Silicon Valley, USA, 2005 and 2006;
- *Grand Challenges of Informatics* conference Budapest, Hungria, 2006;
- *Algorithmic Number Theory* conference, Turku, Finland, 2007;
- colloquium in honor of Henri Cohen, Bordeaux, France, 2007;
- *Central European Conference on Cryptography*, Graz, Austria, 2008;
- MSR Talk Series, Microsoft Research, Redmond, USA, 2009;
- International Congress on Mathematical Software, Kobe, Japan, 2010;
- Euroscipy 2013 conference (advanced tutorial), Bruxelles, 2013;
- Number Theory Down Under conference, Newcastle, Australia, 2016;
- Dagstuhl seminar *Reliable Computation and Complexity on the Reals*, Germany, 2017;
- ICERM workshop for the 75 Years of Mathematics of Computation, Providence, 2018;
- ICERM workshop on Variable Precision in Mathematical and Scientific Computing, Providence, 2020 (virtual event).

Invited to write an entry on the Elliptic Curve Method in the Encyclopedia of Cryptography and Security, Springer, 2005; invited to write an article in the Notices of the American Mathematical Society, 2011, and in the Research Highlights of Communications of the ACM, 2019.

Major Collaborations.
- Coordinator in 1997 of a German-French project (Procope) with the MuPAD group in Paderborn, Germany, led by Prof. Benno Fuchssteiner.
- Head of an associate team co-funded by the University of Canberra (Australia) and Inria with the group of Richard Brent, 2008-2010.

Past Funding. The development of the MPFR library was supported by Inria in several forms (ARC Fiable 1999-2000, ARC AOC 2000-2002, engineer grants 2003-2005 and 2007-2009, post-doctoral grant 2009-2010) and by the "Conseil Régional de Lorraine" (2002). The PI was a main participant of the CADO and CATREL projects supported by the French National Research Agency (ANR). PI of the PRACE project *New Records for Integer Factorization and Discrete Logarithm* (2019-2020) which was awarded 32 million hours on the Juwels supercomputer.

7 Ten-Year Track-Record

The PI has two major research domains: computer arithmetic (where CORE-MATH belongs to), and applications of number theory to cryptography. The PI worked mainly in the second domain in the last years (publications 4, 5, 6, 7, 8, 9); the relevant publications for CORE-MATH are 1, 2, 3. A main publication related to CORE-MATH (*MPFR: A multiple-precision binary floating-point library with correct rounding*, with L. Fousse, G. Hanrot, V. Lefèvre, and P. Pélissier) is not mentioned here since it was published in 2007. The PI has been active in the dissemination of science through the book 10, for which he was both coordinator and main author.

Top 10 Paper Publications.

1. *Modern computer arithmetic*, R. P. Brent, P. Zimmermann, Cambridge University Press, 2010. This monograph has become a reference for algorithms and efficient implementations of arbitrary-precision computer arithmetic, both for integer and floating-point operations. Chapters 3 (Floating-point arithmetic) and 4 (Elementary and special function evaluation) are especially relevant to CORE-MATH.

2. *Optimized Binary64 and Binary128 Arithmetic with GNU MPFR*, V. Lefèvre, P. Zimmermann, 24th IEEE Symposium on Computer Arithmetic (ARITH), 2017. This article publishes new algorithms for addition, subtraction, multiplication, division and square root of floating-point numbers of up to 128-bit precision with correct rounding, yielding a speedup of a factor of 2 or more over previous state-of-the-art. A key feature of these algorithms is that they rely on integer operations: this will be essential for the speed of the CORE-MATH algorithms.

3. *On various ways to split a floating-point number*, C.-P. Jeannerod, J.-M. Muller, P. Zimmermann, 25th IEEE Symposium on Computer Arithmetic (ARITH), 2018. This article is an example of the close relations between the PI and the research group of Jean-Michel Muller.

4. *Factorization of a 768-bit RSA modulus*, T. Kleinjung, K. Aoki, J. Franke, A. K. Lenstra, E. Thomé, J. W. Bos, P. Gaudry, A. Kruppa, P. L. Montgomery, D. A. Osvik, H. te Riele, A. Timofeev, P. Zimmermann, 30th Annual International Cryptology Conference (Crypto), 2010.

5. *Finding Optimal Formulae for Bilinear Maps*, R. Barbulescu, J. Detrey, N. Estibals, P. Zimmermann, Intern. Workshop on the Arithmetic of Finite Fields (WAIFI), 2012.

6. *Discrete Logarithm in* $\mathrm{GF}(2^{809})$ *with FFS*, R. Barbulescu, C. Bouvier, J. Detrey, P. Gaudry, H. Jeljeli, E. Thomé, M. Videau, P. Zimmermann, International Conference on Practice and Theory of Public-Key Cryptography (PKC), 2014.

7. *Imperfect forward secrecy: How Diffie-Hellman fails in practice*, D. Adrian, K. Bhargavan, Z. Durumeric, P. Gaudry, M. Green, J. Alex Halderman, N. Heninger, D. Springfall, E. Thomé, L. Valenta, B. VanderSloot, E. Wustrow, S. Zanella-Béguelin, P. Zimmermann, ACM SIGSAC Conference on Computer and Communications, 2015.

8. *Better Polynomials for GNFS*, S. Bai, C. Bouvier, A. Kruppa, P. Zimmermann, Mathematics of Computation, volume 85, pages 861–873, 2016.

9. *Comparing the difficulty of factorization and discrete logarithm: a 240-digit experiment*, F. Boudot, P. Gaudry, A. Guillevic, N. Heninger, E. Thomé, P. Zimmermann, Proceedings of Advances in Cryptology (CRYPTO), LNCS 12171, pages 62-91, 2020. This work demonstrates the ability of the PI to run very large parallel and distributed computations, which will be crucial within CORE-MATH for the search of HR-cases.

10. *Mathematical Computation with SageMath*, P. Zimmermann, A. Casamayou, N. Cohen, G. Connan, Th. Dumont, L. Fousse, F. Maltey, M. Meulien, M. Mezzarobba, C. Pernet, N. Thiéry, E. Bray, J. Cremona, M. Cremona, A. Ghitza, and H. Thomas, SIAM, 478 pages, 2018. William Stein says about this book *"This fantastic and deep book about how to use Sage for learning and doing mathematics at all levels perfectly complements the existing Sage documentation. [...] Flip to almost any random page in this amazing book, and you will learn how to play with and visualize some beautiful part of mathematics."*

Top 3 Software Publications. As a computer scientist, the PI cannot consider his research without writing programs or libraries either to experiment with a new idea, as a proof-of-concept of a new algorithm, or as a general tool which will be useful to himself, to his research team, or to other researchers. Most of his software contributions are distributed under an open-source license to allow other people to use them in other tools, either free or commercial:

- main designer and main author of GNU MPFR, a library for multiple-precision floating-point arithmetic with correct rounding. Shipped within all Linux distributions. Prerequisite to build the GCC and Gfortran compilers. Used by Magma, SageMath, and the MPFI and MPC libraries. According to `openhub.net`, MPFR *has a well established, mature codebase maintained by a small development team* and *took an estimated 30 years of effort (COCOMO model)*. The experience of the PI when designing algorithms for GNU MPFR will be extremely relevant for CORE-MATH.
- main designer and main author of GNU MPC, a library for multiple-precision complex floating-point arithmetic with correct rounding. Shipped within all Linux distributions. Prerequisite to build the GCC compiler. According to `openhub.net`, MPC *has a well established, mature codebase* and *took an estimated 5 years of effort (COCOMO model).*
- main designer and main author of CADO-NFS, an integer factorization program using the Number Field Sieve. CADO-NFS holds the record of the largest integer factorization (RSA-250) and of the largest discrete logarithm computation (DLP-240) [2]. According to `openhub.net`, CADO-NFS *has a well established, mature codebase* and *took an estimated 115 years of effort (COCOMO model).*

Major contributions to the early careers of excellent researchers. Laurent Fousse, PhD student supervised by the PI, is now hired by Google at the Mountain View headquarters; Damien Stehlé, another PhD student supervised by the PI, is now full Professor in Lyon and obtained an ERC Starting Grant (2014-2018).

Contributions to GNU libc. While designing and writing the CORE-MATH project, the PI did several contributions to the GNU libc mathematical library. In particular he improved the exp10 `binary32` function, with a latency improving from 149 clock cycles to 77 on AArch64, and improved the accuracy of the Bessel j0 `binary32` function.

References

[1] Bailey, D. H. Variable precision computing: Applications and challenges. Slides presented at the ICERM workshop on Variable Precision in Mathematical and Scientific Computing, 2020. `https://www.davidhbailey.com/dhbtalks/dhb-icerm-2020.pdf`.

[2] Boudot, F., Gaudry, P., Guillevic, A., Heninger, N., Thomé, E., and Zimmermann, P. Comparing the difficulty of factorization and discrete logarithm: a 240-digit experiment. In *Proceedings of Advances in Cryptology (CRYPTO)* (2020), D. Micciancio and T. Ristenpart, Eds., vol. 12171 of *Lecture Notes in Computer Science*, pp. 62–91.

[3] Brisebarre, N., Hanrot, G., and Robert, O. Exponential sums and correctly-rounded functions. *IEEE Transactions on Computers 66*, 12 (2017), 2044–2057.

[4] Brunie, N., de Dinechin, F., Kupriianova, O., and Lauter, C. Code generators for mathematical functions. In *2015 IEEE 22nd Symposium on Computer Arithmetic* (2015), pp. 66–73.

[5] C floating point study group teleconference meeting notes. `http://www.open-std.org/jtc1/sc22/wg14/www/docs/n2473.pdf`, 2020.

[6] Chevillard, S., Joldes, M. M., and Lauter, C. Sollya: an environment for the development of numerical codes. In *Third International Congress on Mathematical Software - ICMS 2010* (Kobe, Japan, 2010), K. Fukuda, J. van der Hoeven, M. Joswig, and N. Takayama, Eds., vol. 6327 of *Lecture Notes in Computer Science*, Springer, pp. 28 – 31.

[7] Daramy, C., Defour, D., de Dinechin, F., and Muller, J.-M. CR-LIBM: a correctly rounded elementary function library. In *Advanced Signal Processing Algorithms, Architectures, and Implementations XIII* (2003), F. T. Luk, Ed., vol. 5205, International Society for Optics and Photonics, SPIE, pp. 458 – 464.

[8] de Dinechin, F., Ershov, A. V., and Gast, N. Towards the post-ultimate libm. In *Proceedings of the 17th IEEE Symposium on Computer Arithmetic (ARITH'17)* (2005), P. Montuschi and E. Schwarz, Eds., IEEE Computer Society, pp. 288–295.

[9] de Lassus Saint-Geniès, H., Brunie, N., and Revy, G. Exact lookup tables for the evaluation of trigonometric and hyperbolic functions. *IEEE Trans. Computers 66*, 12 (2017), 2058–2071.

[10] Gal, S. Computing elementary functions: A new approach for achieving high accuracy and good performance. In *Accurate Scientific Computations, Symposium, Bad Neuenahr, FRG, March 12-14, 1985, Proceedings* (1985), W. L. Miranker and R. A. Toupin, Eds., vol. 235 of *Lecture Notes in Computer Science*, Springer, pp. 1–16.

[11] Godunov, A. Algorithms for calculating correctly rounded exponential function in double-precision arithmetic. *IEEE Transactions on Computers 69*, 9 (2020), 1388–1400.

[12] Gustafson, J. L. The Minefield method: A uniformly fast solution to the Table-Maker's Dilemma. `https://bit.ly/2ZP4kHj`, 2020.

[13] Hanrot, G., Lefèvre, V., Stehlé, D., and Zimmermann, P. Worst cases of a periodic function for large arguments. In *Proceedings of the 18th IEEE Symposium on Computer Arithmetic (ARITH'18)* (Montpellier, France, 2007), P. Kornerup and J.-M. Muller, Eds., IEEE Computer Society Press, Los Alamitos, CA, pp. 133–140.

[14] Floating point case study: Intel and floating-point. `https://www.intel.com/content/dam/www/public/us/en/documents/case-studies/floating-point-case-study.pdf`. 11 pages.

[15] Lauter, C. Q. *Arrondi correct de fonctions mathématiques. Fonctions univariées et bivariées, certification et automatisation.* PhD thesis, Université de Lyon - École Normale Supérieure de Lyon, 2008.

[16] Le Maire, J., Brunie, N., de Dinechin, F., and Muller, J.-M. Computing floating-point logarithms with fixed-point operations. In *23rd IEEE Symposium on Computer Arithmetic* (Santa Clara, United States, 2016), P. Montuschi, M. J. Schulte, J. Hormigo, S. F. Oberman, and N. Revol, Eds., IEEE, pp. 156–163.

[17] Lefèvre, V. New Results on the Distance Between a Segment and Z^2. Application to the Exact Rounding. In *17th IEEE Symposium on Computer Arithmetic - Arith'17* (Cape Cod, MA, United States, 2005), P. Montuschi and E. Schwarz, Eds., IEEE Computer Society, pp. 68–75.

[18] Lefèvre, V., Muller, J.-M., and Tisserand, A. The Table Maker's Dilemma. Research Report LIP RR-1998-12, Laboratoire de l'informatique du parallélisme, 1998.

[19] Muller, J.-M., Brunie, N., de Dinechin, F., Jeannerod, C.-P., Joldes, M., Lefèvre, V., Melquiond, G., Revol, N., and Torres, S. *Handbook of Floating-Point Arithmetic, 2nd edition.* Birkhäuser Boston, 2018.

[20] Myklebust, T. G. J. Computing accurate Horner form approximations to special functions in finite precision arithmetic, 2015. `http://arxiv.org/abs/1508.03211`.

[21] Stehlé, D. On the Randomness of Bits Generated by Sufficiently Smooth Functions. In *Seventh Algorithmic Number Theory Symposium - ANTS 2006* (Berlin, Germany, 2006), F. Hess, S. Pauli, and M. Pohst, Eds., vol. 4076 of *Lecture Notes in Computer Science*, Springer, pp. 257–274.

[22] Stehlé, D., Lefèvre, V., and Zimmermann, P. Searching worst cases of a one-variable function using lattice reduction. *IEEE Transactions on Computers 54*, 3 (2005), 340–346.

[23] Torres, S. *Tools for the Design of Reliable and Efficient Functions Evaluation Libraries.* PhD thesis, Université de Lyon, 2016.

[24] Vuik, K. Some disasters caused by numerical errors. `http://ta.twi.tudelft.nl/users/vuik/wi211/disasters.html`.

[25] Zimmermann, P. Accuracy of mathematical functions in single precision. `https://members.loria.fr/PZimmermann/papers/accuracy.pdf`, 2020.

[26] Ziv, A. Fast evaluation of elementary mathematical functions with correctly rounded last bit. *ACM Trans. Math. Softw. 17*, 3 (1991), 410–423.

8 Appendix: on-going funding and submitted proposals

The PI is not currently involved in any specially funded project, and did not submit any other than the present one. The *New Records for Integer Factorization and Discrete Logarithm* project, which was allocated 32 million hours on the PRACE Juwels supercomputer, ended in March 2020 (the PI was principal investigator of this project).

ERC Advanced Grant
Research Proposal (Part B2)

CORE-MATH: Ensuring Correctly Rounded Mathematical Functions

Principal Investigator (PI):	**Dr Paul Zimmermann**
PI's host institution:	**Inria, France**
Proposal full title:	**Ensuring Correctly Rounded Mathematical Functions**
Proposal short name:	**CORE-MATH**
Project duration:	**60 months**
Targeted Review Panel:	**PE6 (Computer Science and Informatics)**

In 1985, the IEEE 754 standard defined for the first time what the result of a computation on floating-point numbers should be. Today, any program using floating-point additions, subtractions, multiplications and divisions yields bit-to-bit identical results, whatever the hardware, compiler, or operating system. This is because IEEE 754 requires the best possible result for these operations, called **correct rounding**.

However, most scientific or industrial applications, like the Large Hadron Collider software, developed by thousands of physicists and engineers over two decades, or Karplus' equation $J(\phi) = A\cos^2\phi + B\cos\phi + C$ in nuclear magnetic resonance spectroscopy, also require the evaluation of various **mathematical functions**: sine, exponential, logarithm, etc. These functions are provided by a **mathematical library**, which does not always provide correct rounding. As a resulting effect, a program using such mathematical functions might yield **wrong results**, which could have disastrous consequences [25]. Moreover, these results might **differ** depending on the mathematical library, hardware, compiler or operating system.

We strongly believe it is the right time to fix that numerical reproducibility issue **once and for all**. CORE-MATH will provide new numerical algorithms to evaluate mathematical functions, which will always yield **correct rounding** (i.e., the best possible result) with speed comparable to the best libraries currently available. This will require **clever algorithms** to identify the most difficult cases for correct rounding, and **innovative research** in the field of the evaluation of mathematical functions.

Thanks to CORE-MATH, scientists, engineers, researchers will obtain **correct results** and thus **bit-to-bit reproducible results** for their numerical computations, with the same efficiency as with currently available mathematical libraries, or even more.

Section A describes the state-of-the-art and the main objectives of CORE-MATH. To reach these objectives, Section B details the methodology and organization of the three Research Tracks and of the Validation Track.

A State-of-the-Art and Objectives

A.1 Introduction

Before describing the state-of-the art, let us introduce the CORE-MATH scientific context. To compute with real numbers, two main schemes exist: the RealRAM model, and floating-point numbers. The RealRAM model uses as much memory as needed to represent exactly real numbers, in consequence it is time and memory expensive, thus unusable for large applications. Floating-point numbers use a fixed amount of memory, and have been standardized through IEEE 754. For efficiency, IEEE 754 defines some fixed-precision formats (single, double, and quadruple precision). If arbitrary precision is needed, the GNU MPFR library is the current reference implementation.

Binary Floating-Point: the IEEE 754 Standard and Beyond. The IEEE 754 standard defines how binary and decimal floating-point arithmetic should be performed. Published in 1985, IEEE 754 was revised in 2008 and 2019 [11]. Scientific applications mainly use the binary formats, while the decimal formats are better suited for applications in finance. The standard defines three main binary formats for numerical computations: `binary32`, `binary64`, and `binary128`, with significands of 24 bits, 53 bits and 113 bits respectively. In this document, we focus on binary formats only:

IEEE 754 format	precision (bits)	$\lvert x\rvert_{\min}$	$\lvert x\rvert_{\max}$
`binary32` (single precision)	24	$1.4 \cdot 10^{-45}$	$3.4 \cdot 10^{38}$
`binary64` (double precision)	53	$4.9 \cdot 10^{-324}$	$1.8 \cdot 10^{308}$
`binary128` (quadruple precision)	113	$6.5 \cdot 10^{-4966}$	$1.2 \cdot 10^{4932}$

IEEE 754 requires *correct rounding* for the four arithmetic operations (addition, subtraction, multiplication, division), the fused multiply-add $\mathrm{FMA}(x, y, z) = xy + z$, and the square root. This means that for a given operation, say $x + t$, the implementation shall return the floating-point number y closest to the exact result according to the given rounding mode (to nearest, toward $-\infty$, toward zero, toward $+\infty$). Therefore, there is a *unique* possible answer y, which is called the *correct rounding* of $x+t$. For a mathematical function, e.g., exp, the correct rounding is the floating-point number y closest to the (infinite precision) exact value of $\exp(x)$. IEEE 754 *only recommends* (unlike for basic arithmetic operations) a set of correctly rounded mathematical functions.[1] Currently available mathematical libraries for the IEEE binary formats (for example GNU libc) do not provide correct rounding, and thus do not conform to IEEE 754 (see for example [27] for single precision).

Hardware and Software Support. Most current processors perform basic arithmetic operations ($+, -, \times, \div, \mathrm{FMA}$) in hardware (usually as micro-code for the division) for single and double precision (`binary32` and `binary64`), but not for quadruple precision, except the IBM Power9 processor, which also implements them in hardware. For quadruple precision, most compilers provide a data type (`__float128` in GCC) with basic arithmetic operations.

Mathematical functions are implemented in software, by a *mathematical library.* Current mathematical libraries do not guarantee correct rounding. Discrepancies from correct rounding range from one ulp (unit in the last place) to hundreds of thousands ulps, even in single precision [27]. For quadruple precision, mathematical functions are available for the `__float128` type since 2011 through GNU libc and/or the `libquadmath` library[2], which originates from the FDLIBM library developed by Sun Microsystems around 1993.

GNU MPFR. GNU MPFR (MPFR for short) is a C library performing arbitrary precision floating-point computations with correct rounding [5]. It thus extends IEEE 754 to arbitrary precision, with the main difference that correct rounding is guaranteed in MPFR not only for basic arithmetic operations, but also for all mathematical functions it implements. The development of MPFR has started in 1999, and it is continuously improved, mainly by the PI and Vincent Lefèvre. It is now a very mature library, which is available on all major operating systems (GNU/Linux, BSD, Windows, AIX, Solaris, MacOS), and is required to compile GCC and Gfortran. The original idea to create MPFR was due to the PI.

[1] Together with Jean-Michel Muller, the PI already argued in 2005 for correct rounding of mathematical functions during the first revision of IEEE 754 [26].

[2] The `libquadmath` code is automatically extracted from GNU libc.

A.2 State-of-the-Art

This section reviews the state-of-the-art of research relevant to CORE-MATH, for the search of Hard-to-Round cases (§A.2.1), and the numerical evaluation of mathematical functions (§A.2.2).

A.2.1 Search for HR-cases.

The search for HR-cases (Hard-to-Round cases) is a crucial step to design efficient algorithms that guarantee correct rounding. Indeed, in Ziv's onion-peeling strategy (which is explained in §A.2.2 below), the last step should always return the correct rounding, and the knowledge of HR-cases determines the minimum accuracy of this last step, i.e., its efficiency. Thus, apart from exact cases, one needs to determine the smallest distance between $f(x)$ and a floating-number in the target precision p (or $p + 1$ for rounding to nearest). This is known as the Table Maker's Dilemma [18]. For some algebraic functions, the HR-cases are known [12], but in general one has to resort to exhaustive search to find them. The first non-trivial algorithm for this task is due to Lefèvre, and the best algorithm currently known is the SLZ algorithm.

Lefèvre's Algorithm. Lefèvre's algorithm [14] was the first non-trivial algorithm to search HR-cases of mathematical functions. It is based on the "three-distance theorem" in a circle. This algorithm uses a first-order approximation of the function f: with target precision p, it splits the interval to check into subranges of roughly $p^{1/3}$ consecutive floating-point values if f is smooth enough, thus giving a complexity of roughly $p^{2/3}$ to check a whole *binade*[3].

The SLZ Algorithm. SLZ [21, 22] is a clever algorithm using modern lattice reduction techniques (due to Coppersmith) to find HR-cases of mathematical functions. It is in fact a family of algorithms, with parameters the degree d of the approximation polynomial of the function, and another parameter called α, the case $d = \alpha = 1$ corresponding to Lefèvre's algorithm. Let f be a mathematical function, p the target precision in bits, $N = 2^p$, M an integer, and assume one searches all integers $|t| \le T$ such that

$$|Nf(t/N) \operatorname{cmod} 1| < 1/M, \qquad (1)$$

where $u \operatorname{cmod} v$ denotes a centered modulus with value in $[-v/2, v/2]$. In a nutshell, SLZ first computes a degree-d approximation polynomial $P(t)$ of $Nf(t/N)$, then constructs polynomials $Q_{i,j} = M^{\alpha-j}(T\tau)^i(P(\tau) + y)^j$, reduces a matrix made from the $Q_{i,j}$ using the Lenstra-Lenstra-Lovász (LLL) algorithm, and if two reduced polynomials $q_1(\tau, y)$ and $q_2(\tau, y)$ are found with small enough coefficients, then for any integer t satisfying Eq. (1), $\tau = t/T$ will be a root of the resultant $\operatorname{Res}_y(q_1, q_2)$, which has integer coefficients.

For univariate functions, the asymptotic complexity of the SLZ algorithm, when the parameter α goes to infinity, is $N^{4/7}$ for $d = 2$, and $N^{1/\sqrt{d+1}}$ for $d \ge 3$ [20]. This yields $N^{0.5}$ for $d = 3$, $N^{0.45}$ for $d = 4$. When the degree d increases, SLZ allows one to consider larger ranges T, but the size of the matrix becomes larger, and the LLL reduction becomes more expensive. A reference implementation of SLZ is available in the BaCSeL software tool, written by Hanrot, Lefèvre, Stehlé and the PI. In practice, the SLZ algorithm starts to outperforms Lefèvre's algorithm for double precision, and gives a large speedup for quadruple precision (see Table 1).

In [20, Section 1.6], Stehlé extended SLZ to bivariate functions: with parameters $d = \alpha = 2$, the asymptotic complexity would be $N^{10/7} \sim N^{1.429}$, where N is the number of possible inputs for each variable. For single precision ($N = 2^{32}$), this corresponds to a complexity of about 2^{46}, and 2^{91} for double precision. However, these algorithms for bivariate functions have not been implemented and used for a real search, thus these figures should be handled with care.

[3] A binade is the set of all binary floating-point numbers between two consecutive powers of two.

format	N	M	T	est. time
`binary64`	2^{53}	2^{53}	2^{20}	1.1 days
`binary128`	2^{113}	2^{113}	2^{44}	420 Myears

Table 1: Best parameters for the SLZ algorithm and estimated time to check a binade of $N/2$ values for the 2^x function, using the BaCSeL tool on an Intel i5-4590 at 3.3 GHz (using one core).

A.2.2 Numerical Evaluation of Mathematical Functions

Once the HR-cases are known for a given function, it is possible to design an efficient correct rounding algorithm. The main ingredients are Ziv's strategy, argument reduction and reconstruction, and efficient polynomial evaluation.

Ziv's strategy. Ziv's strategy (also called onion-peeling strategy) consists in evaluating approximations of $f(x)$ with increasing working precisions $p_1 < p_2 < \cdots$ (see [30]). At step i with precision p_i, one gets an approximation y_i with an error bound ε_i: $f(x) \in [y_i - \varepsilon_i, y_i + \varepsilon_i]$. If both $y_i - \varepsilon_i$ and $y_i + \varepsilon_i$ round to the same number y in the target precision, then by monotonicity of the rounding function, y is the correct rounding of $f(x)$. Otherwise, one continues with a larger precision p_{i+1}.

Ziv's strategy will loop if $f(x)$ is exactly representable in the target precision p ($p + 1$ for rounding to nearest). It is thus mandatory to know these "exact" cases and be able to efficiently detect them. Fortunately for most functions the exact cases are quite rare, for example for the exponential function there is only $e^0 = 1$. A tricky case is the power function x^y, which admits plenty of exact cases, for example $x = 625$ and $y = 3/4$ yield $x^y = 125$.

If the HR-cases are not known, the only way to guarantee correct rounding is to implement Ziv's strategy with an unbounded number of steps, thus with unbounded working precision p_i. However, embarking an arithmetic with unbounded precision would be highly inefficient. This is why HR-cases are needed, or at least a tight bound for the maximal required precision. Ziv's original implementation uses a 3-step strategy for `binary64`: a first step using double precision, a second step using double-double arithmetic, and a final one using 32 digits in base 2^{24}, thus a total of 768 bits. For `binary64`, 768 bits is very likely large enough to guarantee correct rounding, but no proof was given by Ziv.

For a fixed target precision like in the CORE-MATH objectives, a close to optimal strategy is to use two precisions: a *fast path* with precision p_1 that will be able to return a correctly rounded value in almost all cases, and an *accurate path* with precision p_2 large enough to always return the correctly rounded value, as detailed in Figure 1. If t_1 (resp. t_2) is the average time of f_1 (resp. f_2), and ξ the probability that f_1 is not able to round correctly, the average time is:

fast path call a routine f_1, using a working precision $p_1 > p$, yielding an approximation y_1 such that $|y_1 - f(x)| < \varepsilon_1$;

rounding test if $\mathrm{round}_p(y_1 - \varepsilon_1) = \mathrm{round}_p(y_1 + \varepsilon_1)$, return that number;

accurate path otherwise call a routine f_2, using a working precision $p_2 > p_1$, yielding an approximation y_2, and return $\mathrm{round}_p(y_2)$.

Figure 1: The correct rounding algorithm for a univariate function $f(x)$ and target precision p.

$$t = t_1 + \xi t_2. \tag{2}$$

Once the HR-cases are known, we know the minimal precision p_2 required for f_2, and we can design such a function with the smallest t_2. Then different implementations of f_1 can be tried, with different values of t_1 and ξ, keeping the one giving the smallest average time t in Eq. (2). This approach has been successfully used by Godunov for the `binary64` exponential function [7].

Argument Reduction and Reconstruction. When a function $f(x)$ satisfies some mathematical properties, for example $\sin(x + 2\pi) = \sin(x)$, one can use these properties to reduce the evaluation to a small interval, usually around zero or one. This technique is called *argument reduction*. One distinguishes between *additive argument reduction*, like in $\sin(x + 2\pi) = \sin(x)$, and *multiplicative argument reduction*, like in $\log(2x) = \log(x) + \log(2)$. The typical workflow is thus the following: (i) reduce the input x to a reduced argument x', (ii) approximate $f(x')$, and (iii) recover $f(x)$ from $f(x')$. Step (iii) is called *argument reconstruction*, it can be trivial like in $\sin(x + 2\pi) = \sin(x)$.

Algorithms and Arithmetic. For the argument reduction/reconstruction and for the approximation of $f(x')$ (see above), different algorithms and arithmetic implementations are possible.

Once the argument has been reduced to a small interval $x' \in [a, b]$, a good polynomial approximation of the function f over $[a, b]$ is chosen. The state-of-the-art tool to choose this polynomial is the Sollya program [4], which in addition allows one to give constraints on the polynomial coefficients, so that they fit into the desired format. The range $[a, b]$ can also be split further into smaller sub-intervals, on which a polynomial of smaller degree can be used. The "middle" point of each sub-interval can be chosen according to Gal's "accurate table method", so that the constant coefficient of the polynomial yields some extra accuracy [6]. Very recently, Gustafson proposed a completely new approach [9], which however seems to be usable only for single precision.

Finally, the choice of the arithmetic implementation is crucial. Here three cases are distinguished according to the target precision. If the target precision is `binary32`, one can use `binary64` for the working precision of the fast path routine: it will yield $53 - 24 = 29$ extra bits of accuracy, and `binary64` is very efficient since implemented in hardware. For `binary64` target precision, one could use double extended variables, with a significand of 64 bits, but this format is available on some processors only. A better solution is to use a 64-bit integer type [13]. This approach has also demonstrated its efficiency for `binary128`, with multi-word integer types [29], where a speedup of more than 10 was obtained for the quadruple precision exponential function. Table 2 shows the maximal error in units in last place for some mathematical libraries in single precision, and the number of computer cycles needed by the reference GNU libc implementation (it is free, widely available, highly optimized, well tested, and contains benchmark utilities to measure average latency).

Summary. The main weakness of IEEE 754 is that correctly rounded mathematical functions are not mandatory. This has too major consequences: different mathematical libraries might give inaccurate results (and indeed they do [27]), and scientific computations are not bit-to-bit reproducible, as soon as they involve mathematical functions. When IEEE 754 was first published in 1985, it was too early to standardize mathematical functions. Nowadays, a lot of progress has been made by several researchers in the field, particularly by the PI and his co-authors; however, the issue of correctly rounded mathematical functions is far from being solved, since major algorithmic obstructions remain.

A.3 Grand Challenge and Scientific Objectives

It would be rather easy to provide correct rounding with an average factor of two slowdown with respect to current mathematical libraries (which do not yield correct rounding). However,

	GNU libc	Intel Math Library	AMD libm	Newlib	OpenLibm	Musl
asin	0.898	0.528	0.861	0.926	0.743	0.743
exp2	0.502	0.519	1.00	1.02	0.501	0.502
log2	0.752	0.508	0.586	1.65	0.865	0.752
sqrt	**0.500**	**0.500**	**0.500**	**0.500**	**0.500**	**0.500**

function	binary32	binary64	binary128
sin	71/79	306/299	3060/3361
exp	47/43	45/46	3546/3342
pow	82/76	115/108	9412/9027

Table 2: Top: maximal error in units in last place for some mathematical libraries in single precision [27]. Bottom: latency (in clock cycles) for some GNU libc 2.31 functions on an Intel Core i7-8750H (left) and an AMD Ryzen 5-2400G (right), for random inputs in $[-10, 10]$ for exp, in $[0, 10]^2$ for pow, and in $[2^{e-1}, 2^e]$ for sin, with $e = 128, 1024, 16384$ for `binary32`, `binary64` and `binary128` respectively.

to definitively convince users and members of the next IEEE 754 revision committee to adopt correctly rounded mathematical functions, we strongly believe in the following Grand Challenge:

Design correct rounding algorithms for mathematical functions, and corresponding IEEE 754 conforming implementations for single, double, and quadruple precision, with better efficiency than current non-conforming mathematical libraries.

Said otherwise, our Grand Challenge is to have all entries **0.500** in the top part of Table 2, which is the optimal maximal error in terms of units in last place for rounding to nearest, like in the sqrt row, while having better timings than in the bottom part of Table 2.

The scientific challenges to be solved depending on the format (single, double, quadruple), the Grand Challenge naturally splits into three Research Tracks: RT-1 for single precision (§A.3.1), RT-2 for double precision (§A.3.2), and RT-3 for quadruple precision (§A.3.3).

For each Research Track, some hard scientific challenges are identified, and corresponding success criteria are given. The Validation Track will ensure these scientific results will be made available to the scientific and research community.

A.3.1 Research Track 1: Single Precision

The IEEE 754 `binary32` format can represent numbers as small as $x_{\min} \approx 1.4 \cdot 10^{-45}$ (in absolute value), and as large as $x_{\max} \approx 3.4 \cdot 10^{38}$. This format being encoded on 32 bits, there are at most 2^{32} possible inputs. Searching all HR-cases for an univariate function is straightforward, even with a naive algorithm comparing each value to the one obtained with MPFR (which explains why nobody did bother publishing them). However, bivariate functions, for example the power function x^y, the hypot function $\sqrt{x^2 + y^2}$, or the atan2 function $\arctan(y/x)$, remain out of reach for an exhaustive HR-case search. The objective of Research Track 1 is to solve the Table Maker's Dilemma for these bivariate functions, and to provide corresponding correct rounding algorithms:

- **Track RT1-a: Search HR-cases for `binary32` bivariate functions**
- **Track RT1-b: Design efficient correct rounding algorithms for `binary32` bivariate functions**

Criterion of Success for Research Track 1: new algorithms and a reference IEEE 754-conforming implementation for the power function in single precision within 50

cycles on average (compared to 76-82 cycles for the current non-conforming GNU libc implementation)[4].

A.3.2 Research Track 2: Double Precision

For univariate functions in double precision, many HR-cases are known, thanks to the work of Lefèvre and Muller [15, 17]. One exception is the case of periodic functions. For example, the HR-cases of $\sin(x)$ are known only for $|x| \leq 12867/4096 \approx 3.1413$, the worst case in that range being (where the right-hand side is in binary)

$$\sin(8980155785351021 \cdot 2^{-54}) = 0.011110100110010101000001110011000011000100011010010101\underbrace{111...111}_{66}000...$$

HR-cases for larger absolute values (up to the largest `binary64` number $x \approx 1.8 \cdot 10^{308}$) are still unknown. In [10], a new algorithm was proposed for periodic functions, with an estimate of about 4 core-years for the $[2^{1023}, 2^{1024}]$ binade. This gives about 4000 core-years to find all HR-cases of $\sin(x)$ for the whole `binary64` format. One objective of this research track is to find new algorithms that will reduce that search time by a factor of 10 to make it feasible:

- **Track RT2-a: Search HR-cases for `binary64` periodic functions**
- **Track RT2-b: Design efficient correct rounding algorithms for `binary64` periodic functions**

Criterion of Success for Research Track 2: new algorithms and a reference IEEE 754-conforming implementation for the sine function within 100 cycles on average for the whole double precision exponent range (compared to about 300 cycles for the current non-conforming GNU libc implementation).

A.3.3 Research Track 3: Quadruple Precision

Quadruple precision (`binary128`) is the wider IEEE 754 format, which can represent up to 2^{128} different values, ranging from $6.5 \cdot 10^{-4966}$ to $1.2 \cdot 10^{4932}$ in absolute value. With the current state-of-the-art algorithms, it would take of the order of 420M core-years to find HR-cases for one binade (see Table 1), and the `binary128` format corresponds to about 2^{15} binades. No HR-cases are known (except for the square root [12]). CORE-MATH will provide major breakthroughs in two directions:

- **Track RT3-a: Search HR-cases for `binary128` functions**
- **Track RT3-b: Design efficient correct rounding algorithms for `binary128` functions**

For the design of efficient algorithms, an important difference with single and double precision is that basic arithmetic for quadruple precision (addition, subtraction, multiplication, division) is usually implemented in software, and it thus slow (with the already-mentioned exception of the IBM Power9 processor, see §A.1).

Criterion of Success for Research Track 3: new algorithms and a reference IEEE 754-conforming implementation for the exponential function in quadruple precision within 200 cycles on average (compared to 3300-3500 cycles for the current non-conforming GNU libc implementation).

[4] While CORE-MATH will target **all functions** of Annex F from the C language standard, within each research track we identify hard problems that remain currently unsolved, and give corresponding success criteria.

A.3.4 Validation Track

Using the results of RT-1, RT-2, and RT-3, the Validation Track will provide correct rounding implementations for the `binary32`, `binary64`, and `binary128` formats respectively, for all rounding modes. It will address all 28 functions of Annex F of the C language standard: trigonometric functions (acos, asin, atan, atan2, cos, sin, tan), hyperbolic functions (acosh, asinh, atanh, cosh, sinh, tanh), exponential and logarithmic functions (exp, exp2, expm1, log, log10, log1p, log2), power-like functions (cbrt, hypot, pow, sqrt), error and gamma functions (erf, erfc, lgamma, tgamma), together with the new functions planned in the C2X standard [3].

Criterion of Success for the Validation Track: ensure the correct rounding functions designed within CORE-MATH are integrated into at least one of the current mathematical libraries: GNU libc, Intel Math Library, etc.

B Methodology

We explain now how we will achieve our Grand Challenge, and which research plan we will set up for each research track. Some methodology of CORE-MATH is common to all three research tracks, in particular the dependency to the IEEE 754 rounding modes. The correct rounding algorithms designed within CORE-MATH will depend on the rounding mode in the very last step only: first the inputs will be converted to some internal representation, then all computations will be done within that internal representation (where the current rounding mode will have no effect), and the final approximation will be correctly rounded according to the current rounding mode.

For each research track, we detail the CORE-MATH methodology on the example function given in the corresponding criterion of success. As explained above, the other functions of Annex F of the C standard (see §A.3.4) will be considered too, but each example function represents well the main difficulties that will be encountered.

B.1 Research Track 1: Single Precision

Track RT1-a: Search HR-cases for `binary32` bivariate functions

Let us detail the CORE-MATH methodology on the power function x^y. A good news is that not all 2^{64} pairs of inputs yield a result in the `binary32` exponent range. Indeed, for $x = 17$ and $y = 42$, x^y overflows, thus there is no need to consider that pair for HR-cases. For the power function, the number of positive inputs such that x^y does not underflow or overflow is about 2^{61}. However, this number is still huge.

Some preliminary experiments with the bivariate SLZ algorithm in the SageMath computer algebra system [23] yield the following optimal settings for the x^y function, degree $d = 3$ and parameter $\alpha = 2$, where each call of the SLZ algorithm (see §A.2.1) deals with a rectangle of $2T$ consecutive floating-point values for x, and $2U$ for y:

format	T	U	estimated time
`binary32`	2^6	2^6	400 000 years
`binary64`	2^{14}	2^{14}	10^{17} years
`binary128`	2^{31}	2^{31}	10^{44} years

Apart from the fact that `binary64` and `binary128` are out of reach, the interesting figure is $T = U = 2^6$ for `binary32`. This means that each run of the algorithm deals with a square containing $2T = 128$ consecutive `binary32` x-values, and $2U = 128$ consecutive `binary32` y-values, thus a total of 16384 pairs (x, y). This corresponds to about 90 milliseconds per square of 16384 pairs for the SageMath toy implementation. We propose the following alternate algorithm:

- compute an order-1 expansion $f(x, y) \approx a+bt+cu$ around x_0, y_0, with a, b, c floating-point values, and t, u integers, $|t| < T$, $|u| < U$, assuming $1/2 \leq |f(x, y)| < 1$;
- deduce $a' = \mathrm{frac}(2^p a)$, $b' = \mathrm{frac}(2^p b)$, $c' = \mathrm{frac}(2^p c)$, corresponding to the least significant bits, with p the target precision;
- now one wants to find integers t, u such that $a' + b't + c'u \,\mathrm{cmod}\, 1$ is small, where a', b', c' are real numbers in $[0, 1)$, and cmod denotes the centered modulus.

A classical approach for the last step is the following: let a'' be the integer closest to $2^{64}a'$, and similarly for b'' and c'', then one is looking for $a'' + b''t + c''u \,\mathrm{cmod}\, 2^{64}$ small, say less than some bound d in absolute value. Another classical approach (already used for example in Lefèvre's algorithm [14]) is to compute instead $a''+b''t+c''u+d \bmod 2^{64}$ (now with the classical modulus, giving a number in $[0, 2^{64} - 1]$) and check whether it is smaller than $2d$. This could be done at the speed of one operation every clock cycle. On a 3 GHz processor, one should be able to check $3 \cdot 10^9$ `binary32` pairs (x, y) per second, and thus checking all the $\approx 2^{61}$ cases of x^y that do not yield underflow or overflow (see above) would take a few core-years, which becomes feasible.

Track RT1-b: Design correct rounding algorithms for `binary32` bivariate functions

The objective of this research track is to provide efficient correct rounding algorithms for mathematical functions in single precision, unlike current mathematical libraries [27]. The 28 functions of Annex F of the C language standard, detailed in §A.3.4, will be addressed.

The challenging functions will be the bivariate ones (atan2, pow, hypot), since on the one hand the HR-cases will be harder to compute (cf Track RT1-a), and on the other hand they are likely to require more accuracy for the accurate path. Indeed, for a format on p bits, the HR-cases for univariate functions are expected to require about $2p$ bits of accuracy, and those for bivariate functions are expected to require about $3p$ bits of accuracy, thus about 96 bits here. Instead of using Ziv's original strategy (see Figure 1), we propose to have an accurate path with a working precision of 64 bits, and to add a third "very accurate" path with a working precision sufficient to round correctly all HR-cases. This will require to implement an efficient arithmetic with about 96 bits of accuracy. Here the experience of the PI with MPFR will be extremely valuable [16].

B.2 Research Track 2: Double Precision

Track RT2-a: Search HR-cases for `binary64` periodic functions

Research Track RT2-a will first re-evaluate the estimate of 4000 core-years to find all HR-cases of $\sin(x)$ for `binary64` (see §A.3.2). A first research direction will be to investigate whether the algorithm from [10] can be parallelized. For the $[2^{1023}, 2^{1024}]$ binade, where two consecutive floating-point numbers are distant from $\mu = 2^{971}$, this algorithm considers arithmetic progressions of numbers distant of $q\mu$ from each other, where $q = 15\,106\,909\,301$ is chosen such that $q\mu$ is very small modulo 2π, here $q\mu \bmod (2\pi) \approx 4.41 \cdot 10^{-13}$. Then the original SLZ algorithm is used on each arithmetic progression (or Lefèvre's algorithm, which strangely was not even tried in [10]).

Within CORE-MATH, we will search for new ideas making obsolete the state-of-the-art algorithm from [10]. On the algorithmic side, one such idea to be experimented is the following. The algorithm from [10] deals independently with every binade. However, binades could be grouped together, for example if we group the $[2^{1022}, 2^{1023}]$ and $[2^{1023}, 2^{1024}]$ binades together, we have to consider inputs of the form $m \cdot 2^{970}$ for an integer m in the range $[2^{52}, 2^{54}]$. Since the algorithm from [10] is sublinear in the input size N, one can expect a smaller asymptotic complexity.

In the worst-case scenario, assuming only a factor of 4 will be gained, it will decrease the cost of the search of HR-cases for $\sin(x)$ from 4000 years to about 1000 years. This is tractable with university-level resources (using if needed the Grid5000 platform [8] or the PRACE Research Infrastructure [19]). Thus Track RT2-a should be able to determine and publish HR-cases of $\sin(x)$ for the whole `binary64` format, as well as of the other considered functions.

Track RT2-b: Design correct rounding algorithms for `binary64` periodic functions

We will provide correctly-rounded algorithms for `binary64` using Ziv's strategy with two steps: a fast path using 64-bit integer arithmetic, and an accurate path using 128-bit integer arithmetic, assuming it is enough for the HR-cases obtained by Track RT2-a. Indeed, since the advent of 64-bit processors, the clever use of integer operations can be faster than using hardware floating-point operations to implement mathematical functions [13, 29].

Efficient arithmetic will also be needed for the argument reduction step. In the case of $\sin(x)$ for x large, one first needs to compute $k = \lfloor x/(2\pi) \rceil$, then compute the reduced argument $x' = x - 2k\pi$, before approximating $\sin(x')$. For the largest possible x, the integer k will have up to 1022 bits. We will also try the following research direction: since every `binary64` number can be written $x = m \cdot 2^e$ for integers m and e, the idea is to precompute $\tau \approx 2^e \bmod (2\pi)$, then $x' \approx m\tau \bmod (2\pi)$. Using statistical considerations, an accuracy of about 128 bits should be enough for τ, therefore the argument reduction will require a smaller arithmetic and be faster, at the expense of more memory to store the table of the precomputed τ values. Track RT2-b will compare all these strategies and keep the best one.

B.3 Research Track 3: Quadruple Precision

Track RT3-a: Search HR-cases for `binary128` functions

The current cost estimates for the search of `binary128` HR-cases are huge: 420M core-years for one binade of the 2^x function (Table 1). A first research direction will be to revisit this estimate, using algorithmic and implementation ideas, as already detailed in Track RT2-a. On the other side, since the parameters are larger, this opens more room for new ideas.

In case the revised estimate is still too large, it will not be possible to actually *compute* the HR-cases for the target function. We will then implement the fallback solution of determining an *upper bound* for the required working precision. Indeed, the cost of the SLZ algorithm decreases when the number of sought identical bits after the rounding bit increases, i.e., when the parameter M increases in Equation (1). In his PhD thesis, Torres has shown that with $M = 2^{10p}$, degree $d = 45$, and parameter $\alpha = 10$, the cost of checking one quadruple-precision binade for the exponential function decreases to 66 core-years [24, Section 3.9.6.2]. The search then becomes tractable, very likely it will find no HR-case, nevertheless it will prove that a working precision of $113 + 10 \cdot 113 = 1243$ bits will be enough, i.e., about 20 words of 64 bits. Another research direction will be to re-evaluate this estimate, and similarly for smaller values of M (2^{9p}, 2^{8p}, ...), in order to determine the smallest value of M for which the HR-cases search (or more precisely bounding the HR-cases) is feasible. Indeed, for such large values of M, most of the time is spent in the LLL reduction, and one will search for a special-purpose reduction algorithm along the lines of [1].

In any case, Track RT3-a will provide for every function a bound $p_{\max}$, certifying that no solution to Equation (1) exists for $M = 2^{p_{\max}}$. This bound will be used in Track RT3-b.

Track RT3-b: Design correct rounding algorithms for binary128 functions

We will provide correctly-rounded algorithms for binary128 using Ziv's strategy with two or three steps: a fast path using 128-bit integer arithmetic, a second path using 256-bit integer arithmetic, and if needed a third path using larger integer arithmetic, using the bound $p_{\max}$ provided by Track RT3-a.

A preliminary study performed by the PI has shown that using integer-only arithmetic to implement quadruple precision mathematical functions can yield a speedup of 27% over GNU libc on platforms which support binary128 in hardware, and a factor of more than 10 on platforms without such hardware support [28]. These figures are very preliminary and are likely to be improved by CORE-MATH. This will be the main direction followed by Track RT3-b.

For each of the two or three steps of Ziv's strategy (128 bits, 256 bits, and up to about 1200 bits depending on the results of Track RT3-a), we will design efficient algorithms for the argument reduction and reconstruction, and the evaluation of the approximation polynomial itself. The binary128 instances of these algorithms will be automatically generated using the Meta-MPFR generator (see below).

B.4 Validation Track

The Validation Track will take care of efficiently implementing the algorithms designed in RT-1, RT-2, and RT-3. For this purpose, a meta-generator of efficient code (called Meta-MPFR) will be designed and tuned for the scientific objectives of CORE-MATH. This will greatly help disseminate and integrate the scientific results of CORE-MATH.

Track VT-a: Meta-MPFR

Meta-MPFR will be meta-generator, written in a high-level language like Python. It will generate efficient code for the C language with rigorous error bounds, that will be used to efficiently implement the algorithms designed in RT1-b, RT2-b, RT3-b. Meta-MPFR will have two layers:

- a lower layer providing low-level functions, in particular addition, subtraction and multiplication;
- a higher layer providing high-level functions, for example argument reduction or reconstruction, evaluation of an approximation polynomial.

The higher layer will be interfaced with the Sollya tool [4] to automatically compute approximation polynomials. The programs generated by Meta-MPFR will manipulate fixed-precision floating-point numbers stored on several computer words, using only integer operations (as in MPFR). Meta-MPFR will take as input the target precision, the bit size of the target processor (32 or 64), and other parameters like the maximal absolute value that can arise during the computations, the hardware configuration of the target processor (for example presence of a fused multiply-add operation, size of caches). Note that for a given step (fast or accurate path), the working precision will be determined by one of Tracks RT1-a, RT2-a, or RT3-a, then will be the same for all routines needed for that step, and can thus be implicit (contrary to MPFR where each floating-point variable stores its own precision).

Track VT-b: Dissemination and Integration

Track VT-b will take care of the dissemination of the CORE-MATH results toward the scientific community, and its full integration into existing mathematical libraries. For each function of

Annex F from the C language standard, a complete implementation with correct rounding will be published for public review. To assess the correctness of these reference implementations, a “CORE-MATH bugs bounty program” will be launched, with amounts of 1024 euros (single precision bounty), 2048 euros (double precision bounty), and 4096 euros (quadruple precision bounty) for the first individual to find a case that is not correctly rounded[5]. Apart from attracting public media, this will provide an excellent review of the work done in CORE-MATH. These implementations will be integrated into at least one of the main mathematical libraries (GNU libc for example) and thus available for every engineer, scientist or researcher.

B.5 CORE-MATH Roadmap

Table 3 summarizes the distribution of the work over the 5 years of CORE-MATH, according to the dependencies and relationships between the different research tracks. The first priorities for the HR-cases tracks (RT1-a, RT2-a, RT3-a) will be to compute upper bounds for the precision of the corresponding accurate paths, that will be needed by tracks RT1-b, RT2-b, RT3-b. Since Meta-MPFR is independent from the other tracks, its development can start at the beginning of CORE-MATH.

<table>
<tr><th></th><th>Year 1</th><th>Year 2</th><th>Year 3</th><th>Year 4</th><th>Year 5</th></tr>
<tr><td>RT1</td><td colspan="3">RT1-a (binary32 HR-cases)</td><td></td><td></td></tr>
<tr><td>RT1</td><td></td><td colspan="2">RT1-b (binary32 correct rounding)</td><td></td><td></td></tr>
<tr><td>RT2</td><td></td><td colspan="3">RT2-a (binary64 HR-cases)</td><td></td></tr>
<tr><td>RT2</td><td></td><td></td><td colspan="2">RT2-b (binary64 correct rounding)</td><td></td></tr>
<tr><td>RT3</td><td></td><td></td><td colspan="3">RT3-a (binary128 HR-cases)</td></tr>
<tr><td>RT3</td><td></td><td></td><td></td><td colspan="2">RT3-b (binary128 correct rounding)</td></tr>
<tr><td>VT</td><td colspan="2">VT-a (Meta-MPFR)</td><td colspan="3">VT-b (dissemination and integration)</td></tr>
<tr><td></td><td></td><td>Workshop 1</td><td></td><td>Workshop 2</td><td></td></tr>
</table>

Table 3: The CORE-MATH Roadmap (timeline in years).

Three PhD students will be hired to work on the HR-cases search (research tracks RT1-a, RT2-a, RT3-a), while three postdoctoral researchers will be hired to work on the efficient correct rounding algorithms (research tracks RT1-b, RT2-b, RT3-b). A confirmed researcher with 4-8 years of experience will be hired to work on the Validation Track.

B.6 High Risk, High Gain

The outcome of CORE-MATH will be new algorithms providing correct rounding for the three binary IEEE 754 formats, and the corresponding reference implementations. This will only be possible if we manage to do major algorithmic breakthroughs in the search for HR-cases, and in the accurate evaluation of mathematical functions.

High Risk. For Research Track 1, we are confident we will be able to determine the hard-to-round cases for x^y in the `binary32` format, by inventing an algorithm similar to SLZ for bivariate functions, if needed with the help of parallel computations.

For Research Track 2, saving a factor of 10 over the state-of-the-art search for HR-cases [10] entails a high risk. If we only save a smaller factor, for example 3, the total time will still be reachable using distributed computations, for which the PI has a very solid experience [2].

[5] The corresponding amounts will be paid by the Host Institution (Inria).

The risk for Research Track 3 is very high. Indeed, the current estimation of 420 Myears for the HR-cases search is huge (for just one binade), and here a factor of about one million should be saved to make it feasible. If the algorithmic and implementation improvements are not sufficient, another research direction would be to add a second rounding test in Figure 1 after the call to f_2, to check whether $\mathrm{round}_p(y_2 - \varepsilon_2) = \mathrm{round}_p(y_2 + \varepsilon_2)$, where ε_2 is the maximal error for y_2. If that is not the case, a third function f_3 will be called with a larger precision $p_3 > p_2$. Since the cost of determining HR-cases with the SLZ algorithm decreases with the precision, p_3 will be chosen such that this search becomes possible.

The Validation Track will consolidate the results obtained by Research Tracks 1-3. The main risk for this track is that the output of CORE-MATH will not be adopted by the scientific community. To mitigate this risk, contributions will be made to the main mathematical libraries used in scientific applications very early during CORE-MATH. (In addition, the PI is member of the IEEE 754 discussion list since 2001, and of the C Floating-Point group since early 2020.)

High Gain. Computer science achievements made IEEE 754 the most famous and successful industrial standard. However, it did not settle the case of correct rounding for mathematical functions, which has produced many incorrect results since 1985, and is still preventing bit-to-bit reproducibility of numerical computations. CORE-MATH will push the next revision of IEEE 754 to require correct rounding for mathematical functions. The timeline is perfect, since the next revision is due in 2029. CORE-MATH will open new possibilities for engineers and scientists from all domains. Firstly, scientific applications will yield the *best possible result* and become *bit-to-bit reproducible*, across hardware processors, compilers, operating systems. Secondly, since the roundoff error for every mathematical function will be bounded, it will become possible to compute rigorous error bounds for a whole computation. In particular, it will be possible to perform rigorous interval arithmetic with mathematical functions, and thus obtain a correct containment interval for the result of a whole computation. Last but not least, CORE-MATH will provide new algorithms for quadruple precision which will not only yield correct rounding, but also provide more than a tenfold speedup with respect to the best publicly available libraries. This will make quadruple precision really accessible for applications requiring it. In summary, CORE-MATH will allow to compute just right, and still fast!

The economic impact of CORE-MATH will be multiple. On the one hand, the cost of developing numerical applications will decrease, since it will no longer be required to test them on every different combination of hardware, compiler, operating system (or worse to tweak them so that the test suites run) and we will get for free *forward reproducibility*, i.e., a program written at year Y will still yield the same result at year $Y + 10$. This is not the case currently, since any tiny change in the mathematical library (either improving or degrading the accuracy) might change the final result. We also expect it will enable to join or share the development efforts of the different mathematical libraries currently available. Finally, *vendor lock-in* will no longer be possible, where the library designed by a vendor calls non-optimal routines on hardware from a different vendor.

B.7 Conclusion and Perspectives

As for the perspectives, one expectation is that CORE-MATH will motivate other researchers and numerical analysts to promote correct rounding for other domains of computation or other operations. For example, despite computations with (floating-point) complex numbers are standardized in the C language, there is currently no requirement for correct rounding at all, even when multiplying or dividing two complex numbers!

References

[1] Bi, J., Coron, J., Faugère, J., Nguyen, P. Q., Renault, G., and Zeitoun, R. Rounding and chaining LLL: finding faster small roots of univariate polynomial congruences. In *Public-Key Cryptography - PKC 2014 - 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings* (2014), H. Krawczyk, Ed., vol. 8383 of *Lecture Notes in Computer Science*, Springer, pp. 185–202.

[2] Boudot, F., Gaudry, P., Guillevic, A., Heninger, N., Thomé, E., and Zimmermann, P. Comparing the difficulty of factorization and discrete logarithm: a 240-digit experiment. In *Proceedings of Advances in Cryptology (CRYPTO)* (2020), D. Micciancio and T. Ristenpart, Eds., vol. 12171 of *Lecture Notes in Computer Science*, pp. 62–91.

[3] C Floating-point Group. IEC 60559 math functions for C2X. `http://www.open-std.org/jtc1/sc22/wg14/www/docs/n2373.pdf`, 2019.

[4] Chevillard, S., Joldes, M. M., and Lauter, C. Sollya: an environment for the development of numerical codes. In *Third International Congress on Mathematical Software - ICMS 2010* (Kobe, Japan, 2010), K. Fukuda, J. van der Hoeven, M. Joswig, and N. Takayama, Eds., vol. 6327 of *Lecture Notes in Computer Science*, Springer, pp. 28 – 31.

[5] Fousse, L., Hanrot, G., Lefèvre, V., Pélissier, P., and Zimmermann, P. MPFR: A multiple-precision binary floating-point library with correct rounding. *ACM Trans. Math. Softw. 33*, 2 (2007).

[6] Gal, S. Computing elementary functions: A new approach for achieving high accuracy and good performance. In *Accurate Scientific Computations, Symposium, Bad Neuenahr, FRG, March 12-14, 1985, Proceedings* (1985), W. L. Miranker and R. A. Toupin, Eds., vol. 235 of *Lecture Notes in Computer Science*, Springer, pp. 1–16.

[7] Godunov, A. Algorithms for calculating correctly rounded exponential function in double-precision arithmetic. *IEEE Transactions on Computers 69*, 9 (2020), 1388–1400.

[8] The Grid'5000 testbed for parallel and distributed computing. `https://www.grid5000.fr`.

[9] Gustafson, J. L. The Minefield method: A uniformly fast solution to the Table-Maker's Dilemma. `https://bit.ly/2ZP4kHj`, 2020.

[10] Hanrot, G., Lefèvre, V., Stehlé, D., and Zimmermann, P. Worst cases of a periodic function for large arguments. In *Proceedings of the 18th IEEE Symposium on Computer Arithmetic (ARITH'18)* (Montpellier, France, 2007), P. Kornerup and J.-M. Muller, Eds., IEEE Computer Society Press, Los Alamitos, CA, pp. 133–140.

[11] IEEE standard for floating-point arithmetic, 2019. 84 pages.

[12] Lang, T., and Muller, J.-M. Bounds on runs of zeros and ones for algebraic functions. In *Proceedings of the 15th IEEE Symposium on Computer Arithmetic* (2001), IEEE Computer Society, pp. 13–20.

[13] Le Maire, J., Brunie, N., de Dinechin, F., and Muller, J.-M. Computing floating-point logarithms with fixed-point operations. In *23rd IEEE Symposium on Computer Arithmetic* (Santa Clara, United States, 2016), P. Montuschi, M. J. Schulte, J. Hormigo, S. F. Oberman, and N. Revol, Eds., IEEE, pp. 156–163.

[14] Lefèvre, V. New Results on the Distance Between a Segment and Z^2. Application to the Exact Rounding. In *17th IEEE Symposium on Computer Arithmetic - Arith'17* (Cape Cod, MA, United States, 2005), P. Montuschi and E. Schwarz, Eds., IEEE Computer Society, pp. 68–75.

[15] Lefèvre, V., and Muller, J.-M. Worst Cases for Correct Rounding of the Elementary Functions in Double Precision. In *15th IEEE Symposium on Computer Arithmetic - ARITH 2001* (Vail, Colorado, 2001), N. Burgess and L. Ciminiera, Eds., pp. 111–118.

[16] Lefèvre, V., and Zimmermann, P. Optimized binary64 and binary128 arithmetic with GNU MPFR. In *24th IEEE Symposium on Computer Arithmetic, ARITH 2017, London, United Kingdom, July 24-26, 2017* (2017), N. Burgess, J. D. Bruguera, and F. de Dinechin, Eds., pp. 18–26.

[17] Lefèvre, V. Hardest-to-round cases – part 2. `http://tamadiwiki.ens-lyon.fr/tamadiwiki/images/c/c1/Lefevre2013.pdf`, 2013. Slides presented at the final TaMaDi meeting. 30 pages.

[18] Lefèvre, V., Muller, J.-M., and Tisserand, A. Toward correctly rounded transcendentals. *IEEE Transactions on Computers 47* (1998), 1235 – 1243.

[19] Partnership for Advance Computing in Europe. `https://prace-ri.eu`.

[20] Stehlé, D. *Algorithmique de la réduction de réseaux et application à la recherche de pires cas pour l'arrondi de fonctions mathématiques.* Theses, Université Henri Poincaré - Nancy I, 2005.

[21] Stehlé, D. On the Randomness of Bits Generated by Sufficiently Smooth Functions. In *Seventh Algorithmic Number Theory Symposium - ANTS 2006* (Berlin, Germany, 2006), F. Hess, S. Pauli, and M. Pohst, Eds., vol. 4076 of *Lecture Notes in Computer Science*, Springer, pp. 257–274.

[22] Stehlé, D., Lefèvre, V., and Zimmermann, P. Searching worst cases of a one-variable function using lattice reduction. *IEEE Transactions on Computers 54*, 3 (2005), 340–346.

[23] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.1)*, 2020. `https://www.sagemath.org`.

[24] Torres, S. *Tools for the Design of Reliable and Efficient Functions Evaluation Libraries.* PhD thesis, Université de Lyon, 2016.

[25] Vuik, K. Some disasters caused by numerical errors. `http://ta.twi.tudelft.nl/users/vuik/wi211/disasters.html`.

[26] Zimmermann, P. Why transcendentals and arbitrary precision? Invited talk at the IEEE 754 revision committee, Sun Menlo Park, 2005.

[27] Zimmermann, P. Accuracy of mathematical functions in single precision. `https://members.loria.fr/PZimmermann/papers/accuracy.pdf`, 2020.

[28] Zimmermann, P. Faster expf128. `https://sourceware.org/pipermail/libc-alpha/2020-June/115229.html`, 2020.

[29] Zimmermann, P. How slow is quadruple precision? Invited talk at the ICERM workshop on Variable Precision in Mathematical and Scientific Computing, Providence, 2020. `https://icerm.brown.edu/events/htw-20-vp/`.

[30] Ziv, A. Fast evaluation of elementary mathematical functions with correctly rounded last bit. *ACM Trans. Math. Softw. 17*, 3 (1991), 410–423.

Rocquencourt, le 28 juillet 2020

Commitment of the host institution for ERC Calls 2020

The Inria (Institut national de recherche en informatique et en automatique), which is the applicant legal entity, confirms its intention to sign a supplementary agreement with **Mr. Paul ZIMMERMANN** in which the obligations listed below will be addressed should the proposal entitled "***CORE-MATH: Ensuring Correctly Rounded Mathematical Functions***" be retained.

Performance obligations of the applicant legal entity that will become the beneficiary of the H2020 ERC Grant Agreement (hereafter referred to as the Agreement), should the proposal be retained and the preparation of the Agreement be successfully concluded:

The applicant legal entity commits itself to hosting and engaging the principal investigator for the duration of the grant to:

a) ensure that the work will be performed under the scientific guidance of the principal investigator who is expected to devote *in the case of an Advanced Grant at least 30% of his total working time* to the ERC-funded project (action) and spend at least 50% of his total working time in an EU Member State or Associated Country;

b) carry out the work to be performed, as it will be identified in Annex 1 of the Agreement, taking into consideration the specific role of the *principal investigator*;

c) enter — before signature of the Agreement — into a '*supplementary agreement*' with the *principal investigator*, that specifies the obligation of the *applicant legal entity* to meet its obligations under the Agreement;

d) provide *the principal investigator* with a copy of the signed Agreement;

e) guarantee the *principal investigator's* scientific independence, in particular for the:
   i. use of the budget to achieve the scientific objectives;
   ii. authority to publish as senior author and invite as co-authors those who have contributed substantially to the work;
   iii. preparation of scientific reports for the project (action);
   iv. selection and supervision of the other *team members* (hosted *[and engaged]* by the *applicant legal entity* or other legal entities), in line with the profiles needed to conduct the research and in accordance with the *applicant legal entity's* usual management practices;
   v. possibility to apply independently for funding;
   vi. access to appropriate space and facilities for conducting the research;

f) provide — during the implementation of the project (action) — research support to the *principal investigator* and the team members (regarding infrastructure, equipment, access rights, products and other services necessary for conducting the research);

g) support the *principal investigator* and provide administrative assistance, in particular for the:
   i. general management of the work and his team

ii. scientific reporting, especially ensuring that the team members send their scientific results to the *principal investigator*;
iii. financial reporting, especially providing timely and clear financial information;
iv. application of the *applicant legal entity's* usual management practices;
v. general logistics of the project (action);
vi. access to the electronic exchange system (see Article 52 of the Agreement);

h) inform the *principal investigator* immediately (in writing) of any events or circumstances likely to affect the Agreement (see Article 17 of the Agreement);

i) ensure that the *principal investigator* enjoys adequate:
   i. conditions for annual, sickness and parental leave;
   ii. occupational health and safety standards;
   iii. insurance under the general social security scheme, such as pension rights;

j) allow the transfer of the Agreement to a new beneficiary ('portability'; see Article 56a of the Agreement).

k) take all measures to implement the principles set out in the Commission Recommendation on the European Charter for Researchers and the Code of Conduct for the Recruitment of Researchers[1] - in particular regarding working conditions, transparent recruitment processes based on merit and career development – and ensure that the *principal investigator*, researchers and third parties involved in the project (action) are aware of them.

l) respect the fundamental principle of research integrity and ensure that persons carrying out research tasks follow the good research practices and refrain from the research integrity violations described in the European Code of Conduct for Research Integrity. If any such violations or allegations occur, verify and pursue them and bring them to the attention of the Agency.

For the host institution:

Inria

Marie-Hélène PAUTRAT

Director of European Partnerships

Marie-Helene.Pautrat@inria.fr

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE
Inria

[1] Commission Recommendation 2005/251/EC of 11 March 2005 on the European Charter for Researchers and on a Code of Conduct for the Recruitment of Researchers (OJ L 75, 22.3.2005, p. 67).

European Commission

FUNDING &
TENDERS PORTAL
E-RECEIPT

Digitally sealed by the European Commission
Date: 2020.08.26 09:52:43 CEST
Reason: Acknowledgement of Receipt