InterFLOP Project

AAPG 2020 PRCE

01/02/2021 – 01/02/2025

In cooperation with Society for Industrial and Applied Mathematics

2023 SIAM Conference on Computational Science and Engineering
MS177 Sustainable Computing with the Help of Tools, Mixed-Precision, and Optimistic Error Estimates

INTERFLOP: Consortium

David DEFOUR

El-Mehdi El Arar

Pablo OLIVEIRA

Devan SOHIER

Fabienne JEZEQUEL

Stef GRAILLAT

Jean-Luc LAMOTTE

Théo MARY

Franck VEDRINE

Julien SIGNOLES

Yves LHUILLIER

Bruno LATHUILIERE

Eric PETIT

Wilfried KIRSCHENMANN

François FEVOTTE

External Collaborators

Roman IAKYMCHUCK

InterFLOP: Motivations

“Floating-Point arithmetic is considered an esoteric subject by many people”.
D. Goldberg

1980 → 2000 → 2020

IEEE-754

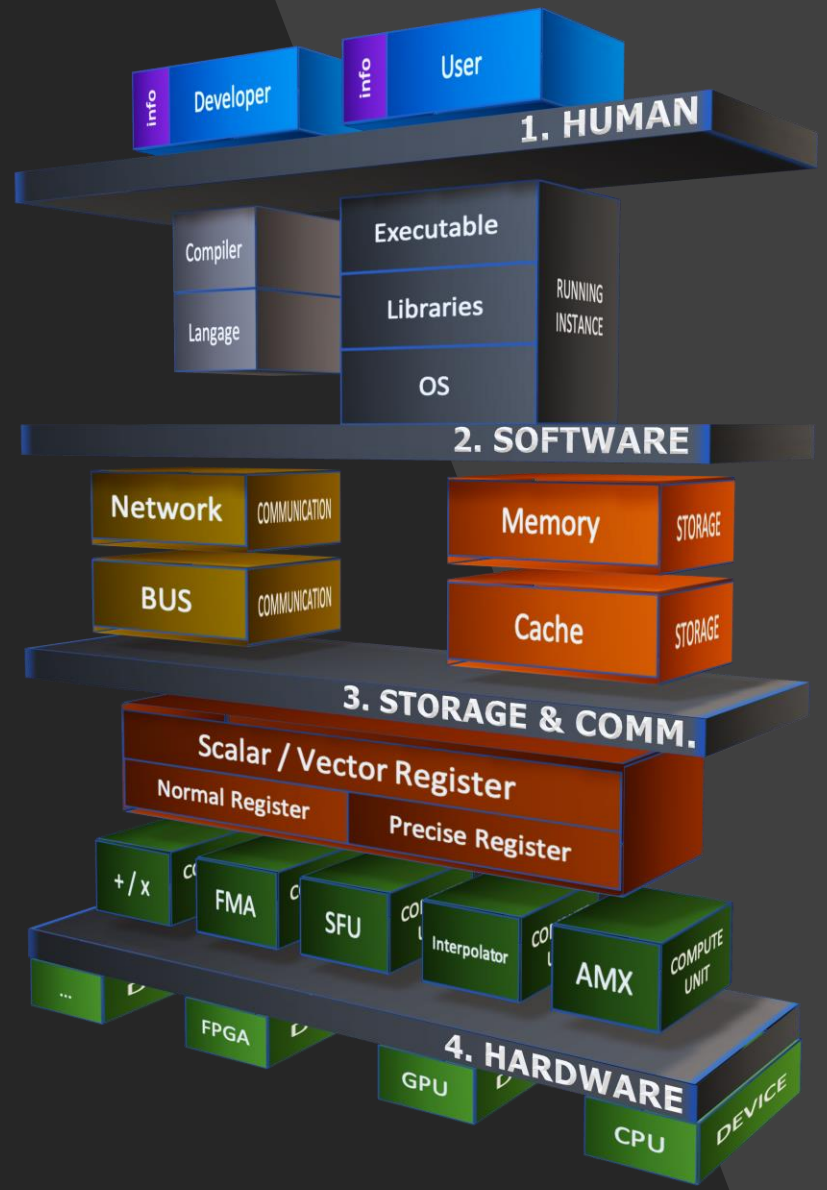
Numerical tools & methodology

InterFLOP

New apps:
AI, Drug simulation…

New Formats & behaviors:
BF16, FP16, FP24, FP128,
Unum, Posit, FlexPoint, FPANR…

New Units:
MatrixUnit, SpecialFunc Unit,
Interpolation, Compressor…



InterFLOP: Motivations

- Desired information to collect
  - Numerical statistics (min, max, avg, … for a given set of variables)
  - Numerical instability
    - Quantify propagation of rounding errors
    - Spot cancellation's during subtractions
  - Spot input values, line number, operations and timing incriminated
- Existing tools: CADNA, Verificarlo, VERROU, PROMISE, FLD-LIB …
  - Implemented technics
    - Operators overloading (C++, Python)
    - Compiler extension (LLVM)
    - Dynamic Binary Instrumentation (Pin, Valgrind)
    - Full System Emulation (UNISIM)
    - Various arithmetic
      - Synchronous / Asynchronous
      - Monte-Carlo, Stochastic (introducing small perturbation in inputs / results of operations)
      - Taylor Based, Affine, Floating-Point (propagation of errors, noises)
      - Static analysis
  - Take a large variety of inputs & executions platforms
    - Languages: Fortran, C, C++, Python, binary code, external library
    - Platform: Linux, Windows
    - Remark: work with **real code**

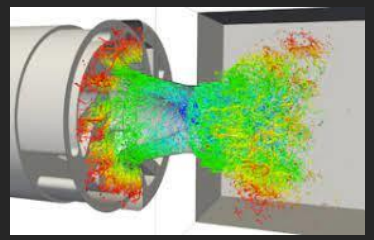
InterFLOP: main objectives

- Mutualize development among various numerical tools by
  1. Proposing a common front-end
     (Input specifications)
  2. Exchange methodology during execution
     (Functional specifications)
  3. Perform post-processing and statistical analysis
     (Provide useful and readable analysis)
  4. Suggest optimizations
     (Precision auto-tuning, verified computing)

- Extends existing tools & technics
  - *Shadow memory, chromatic analysis, probabilistic analysis ….*

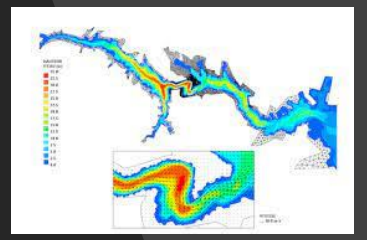
Application Cases

YALES2



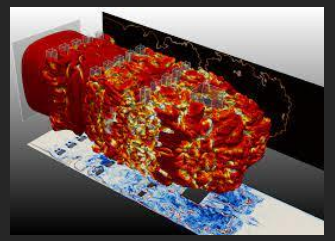
solving of two-phase combustion from primary atomization to pollutant prediction

code_aster



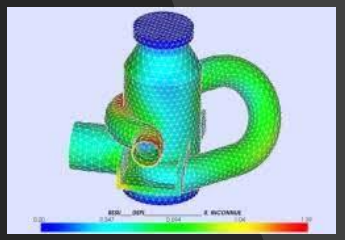
numerical modelling in mechanics

AVBP



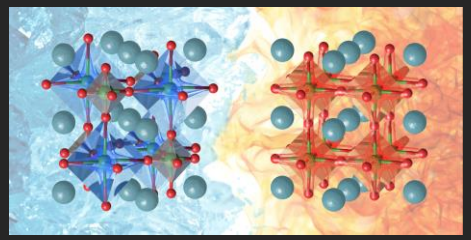
CFD tools to perform DNS and LES of compressible, reacting, turbulent, multispecies flows

TELEMAC

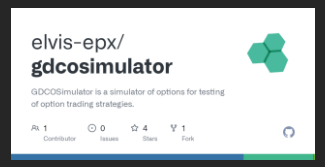


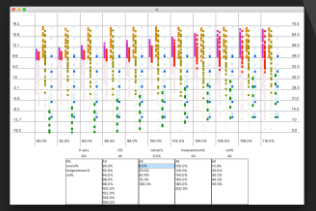
Numerical modelling of Free surface hydraulic, Sediment, Waves, in 1D, 2D or 3D

Abinit



Simulation for materials science





Option trading strategies

Examples of addressed issues

The variance of the error for stochastic rounding

Assume that $\delta_1, \delta_2, \ldots$ in that order are random errors on elementary operations obtained from SR-nearness. $\phi_j = \prod_{k=j}^{n}(1+\delta_k)$. For $i < j$ we have

$$\phi_i = \prod_{k=i}^{j-1}(1+\delta_k)\prod_{k=j}^{n}(1+\delta_k) = \prod_{k=i}^{j-1}(1+\delta_k)\phi_j.$$

Let K a subset of $\mathbb{N}$ of cardinal n and $\psi_K = \prod_{k\in K}(1+\delta_k)$. Since $|\delta_k| \leqslant u$ for all $k \in K$ we have

$$|\psi_K| \leqslant (1+u)^n.$$

Throughout this presentation, let $\gamma_n(u) = (1+u)^n - 1$ an

Lemma 2.

Under SR-nearness ψ_K satisfies

1. $E(\psi_K) = 1$.
2. *Let $K' \subset \mathbb{N}$ such that $|K \cap K'| = m$, under the assum $\forall j \in K \triangle K', k \in K \cap K', j < k$ we have*

$$0 \leqslant \mathrm{Cov}(\psi_K, \psi_{K'}) \leqslant$$

3. $V(\psi_K) \leqslant \gamma_n(u^2)$,

where $\gamma_n(u^2) = (1+u^2)^n - 1 = nu^2 + O(u^3)$.

El-Mehdi EL ARAR — Stochastic Rounding Variance and Probabilistic Bou

Correction Formule 2

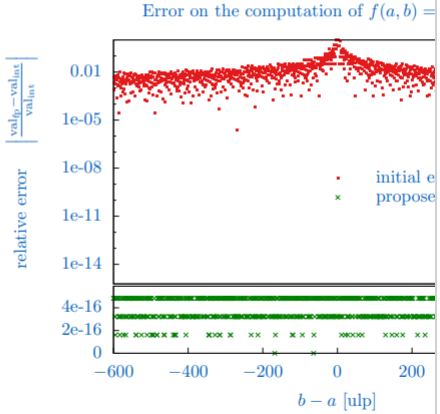
$$f(a,b) = \begin{cases} a & \text{si } a = b \\ 2\frac{ba^{2/3} - ab^{2/3}}{b^{2/3} - a^{2/3}} & \text{sinon} \end{cases}$$

wolfram alpha

$f(a,b) = 2$

Étude empirique

- en dehors du code
- autour du point problématique
- référence = arithmétique d'intervalles



Verrou/code_aster : travaux en cours

CHARACTERIZING NATIVE INSTRUCTION EXECUTION

Native execution of instruction tests raise issues

- Side effect may be unsafe for execution
- Random inputs may generate errors

Side effect	difficulty	Resolution
Memory access	May access memory randomly	a) Fix target address using input value control b) Run in an protected environment
Branch	May transfer control to any memory location	Step in debugger
Interrupts and Traps	Leave the program scope	Run in an hypervisor
Hypervisor ops	Hard to confine	Hardware debug ?
Access HW counters	Environment dependent values	No satisfying solution to our knowledge

InterFLOP

ANR-20-CE46-0009
(2020-2024)

Funded by The French National research Agency

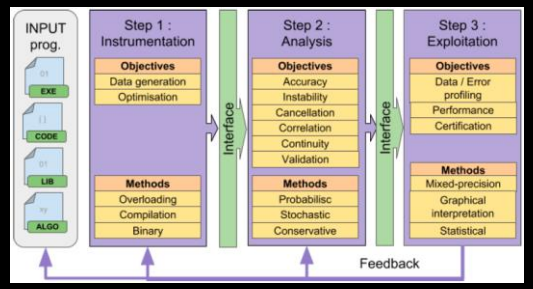
D. DEFOUR, F. FEVOTTE, S. GRAILLAT, F. JEZEQUEL, W. KIRSCHENMANN, J.-L. LAMOTTE, B. LATHUILIERE, Y. LHUILLIER, P. de OLIVEIRA, E. PETIT, J. SIGNOLES, D. SOHIER, F. VEDRINE.

1. MOTIVATIONS

"Floating-Point arithmetic is considered an esoteric subject by many people". *D. Goldberg*

The era is becoming increasingly complex as

- **News apps are rising (AI, Drug simulation)**
- **New formats are becoming available (BF16, FP16, FP24, FP128, Unum, Posit, FlexPoint, FPANR…)**
- **New units, new operators and implementations (MatrixUnit, SpecialFunc Unit, Interpolation, Compressor…)**



2. OBJECTIVES

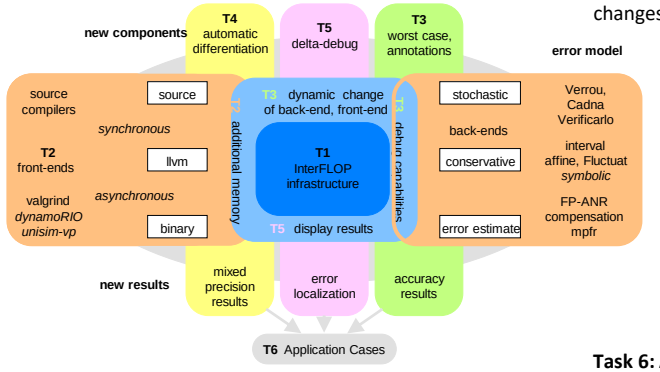
Offer a common platform integrating major tools of the French Floating-Point community to tackle the FP challenges and recent evolutions of software and hardware. We propose new analyses and combinations of existing ones to address the challenge of providing a quick and precise numerical diagnosis requiring little user expertise. InterFLOP will collect and combine information on numerical instabilities, catastrophic cancellations, unstable tests, build various statistical analyses of program executions at minimal overhead.

3. DESCRIPTION

Task 1: Specification of the platform
Propose an operational workflow. Define the type and the format of the data exchanged between each module to minimize bandwidth and memory usage while maximizing the amount of useful exchanged information. Promote a modular, sustainable and open platform with a common exchange specification between the modules while minimizing the impact on performance.

Task 2: Front-end and mechanism to collect information
Enable mixed analysis between the different FP arithmetic. The choice of the arithmetic will come from the availability of different back-ends: floating-point like FP-ANR, Monte-Carlo – MCA, stochastic – CESTAC and Taylor based or affine arithmetic.

Task 3: Models for error estimation and composite analysis
Define and implement composite analyses to illustrate the added value of the software chain. Promote an approach based on (i) an efficient search for instabilities (ii) guarantees of robustness / absence of instabilities on some parts of the code, and (iii) additions of generated code annotations and enabling dynamic changes of the analysis mode.



Task 4: Precision auto-tuning and verified computing
Validate the accuracy of numerical results, automatically tune the precision to achieve the desired accuracy for the result and propose new compressed format based.

Task 5: Post-processing and statistical analysis of the results
Tackles the problem of post-processing and analyzing the results of the InterFLOP chain through three axes: statistical analysis, instabilities tracking and visualization.

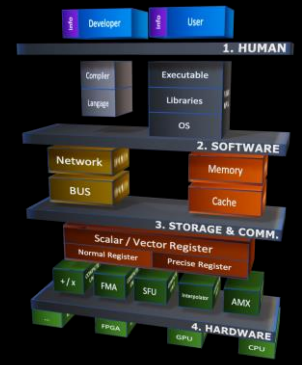
Task 6: Application Cases
Provide a feedback on the results from the other tasks with regards to their use in industrial applications and propose new analysis methodologies.
Considered applications: Yales2, AVBP, Abinit, Slang, EPX, Code_Aster, Telemac, quantitative analysis.

4. EXPECTED RESULTS

- **Common platform available at: https://github.com/interflop**
- **Adapt the granularity level of inspection and type of analyze to the application and user's need**
- **Automatic exploration of precision**
- **Statistical and visual analysis**
- **Validated on real applications**

5. CONSORTIUM

8 Complementary partners which bring their own expertise, tools and methodology in software analysis, compilation, Numerical Simulation, Statistics, Computer arithmetic, Parallelism, Computer architecture to make this project successful.



6. CONCLUSION

Takes the problem of numerical bug detection, software verification and validation to a new level, necessary to address issues that will be encountered with larger problems, new architectures, and new representation formats.
Numerical bug detection will be aided and guided through a unique interface at every step of the lifecycle of a software starting from its prototyping, testing, installation and operation.
Industrial and academic users could then evaluate basic compositions and develop customized ones for their own needs. Such composite analyses mixing execution (for speed on large codes), analysis (for precise diagnosis) and auto-tuning (to propose automatic enhancement) will be pioneer in the field and will be enriched with statistical and visual analysis.