Verificarlo applied to ABINIT: detecting numerical instability

Yohan Chatelain, Pablo De Oliveira, Eric Petit, Jordan Bieder and Marc Torrent

Introduction

Objective:

- Numerical debugger and analyzer of the floating-point model

Context:

- Complex HPC environment: heterogeneous parallel architecture, compiler optimization, parallelization paradigm
- ABINIT: large program with millions line of code

Proposal:

- Automatically pinpoint the impact of the floating-point model on the numerical stability of regions of code

Outline

- Verificarlo
  - How does it works ?
  - Estimating output error
  - An example: Tchebychev polynomial
- Application on ABINIT
  - Hydrogen test case
  - Perovskite test case
  - Numerical sensitive functions
- Conclusion & future prospects

Verificarlo

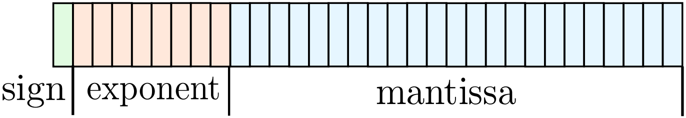
Verificarlo: What is the purpose ?

- Open Source Project under GPL licence, developped by University of Versailles and ENS Paris-Saclay
- Automatically analyses the numerical stability of applications
- Introduces a noise on each floating-point operation

Verificarlo: reminder about floating-point

IEEE-754 Single Precision 32-bit

$$f = (-1)^{sign} \times 2^{exponent-127} \times 1.M \,, M = \sum_{i=1}^{23} 2^{-i}$$

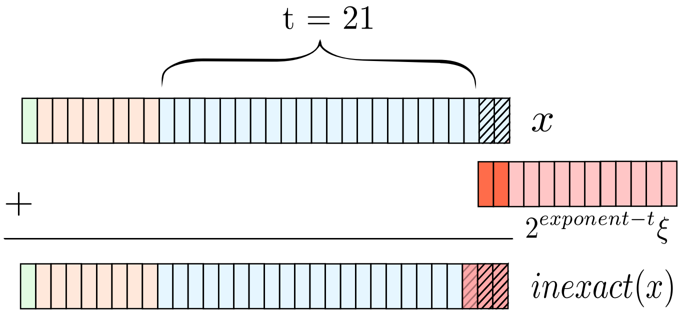


Precision	sign	exponent	mantissa	Total bits
Single	1	8	23	32
Double	1	11	52	64

Verificarlo: How does it works ?

Monte Carlo Arithmetic

$$inexact(x) = x + 2^{exponenent-t}\xi, \xi \in [-\frac{1}{2}, \frac{1}{2}] \text{ t virtual precision}$$



FP operations $\circ$ are replaced by:

$$mca(x) = round(inexact(inexact(x) \circ inexact(y)))$$

Verificarlo: Estimating output error

Rounding errors distribution:

- Estimates by using N Monte Carlo samples

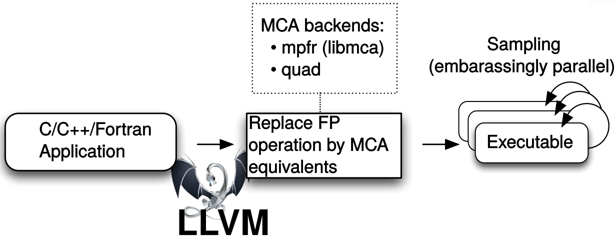
Significant digits number:

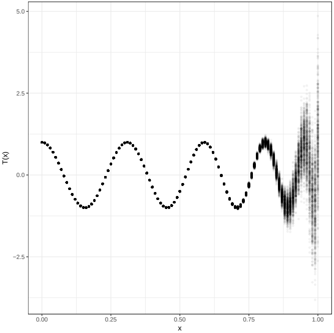
- $\tilde{s}(\chi) = -\log_{10}\left(\frac{\tilde{\sigma}}{\tilde{\mu}}\right) \xrightarrow[N\to\infty]{} s = -\log_{10}\left(\frac{\sigma}{\mu}\right)$

$\tilde{\mu}$: empirical mean
$\tilde{\sigma}$: empirical standard deviation

Estimate the number of correct significant digits

Verificarlo: An example



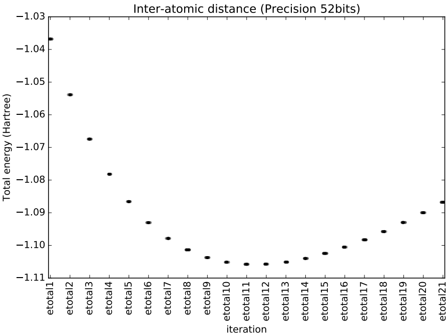


Tchebychev polynomial:

- $T_{10}(x), x \in [0, 1]$
- 100 points across $[0, 1]$
- 500 samples for each point evaluated
- Instability around 1

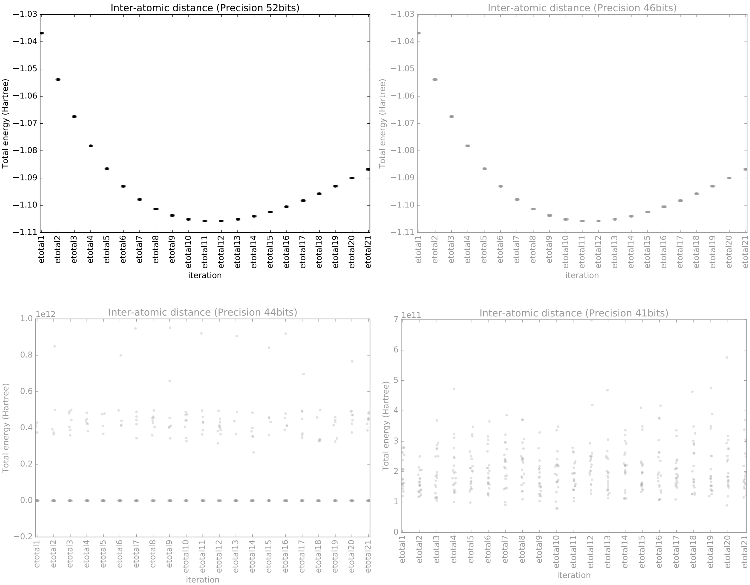
Application on ABINIT

Hydrogen test case

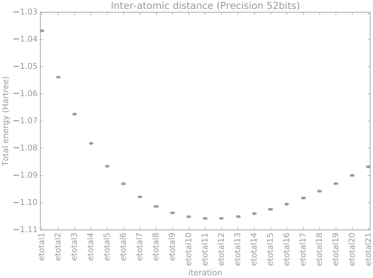


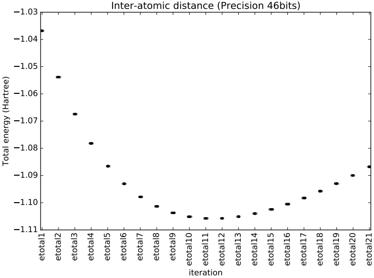
- The test case:
  - Find optimal inter-atomic distance for two hydrogen atoms
  - Simple example without non-local effects
  - Proof of concept to evaluate the cost of the method
  - Measure mean and standard deviation of MCA errors
  - Global analysis

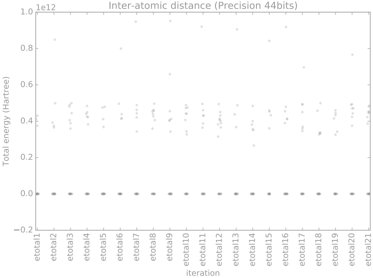
Verificarlo on whole ABINIT - Hydrogen test

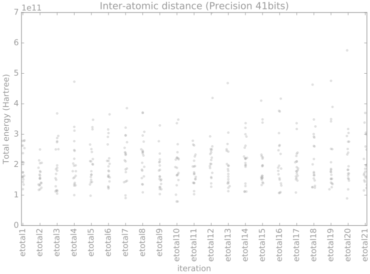


Verificarlo on whole ABINIT - Hydrogen test

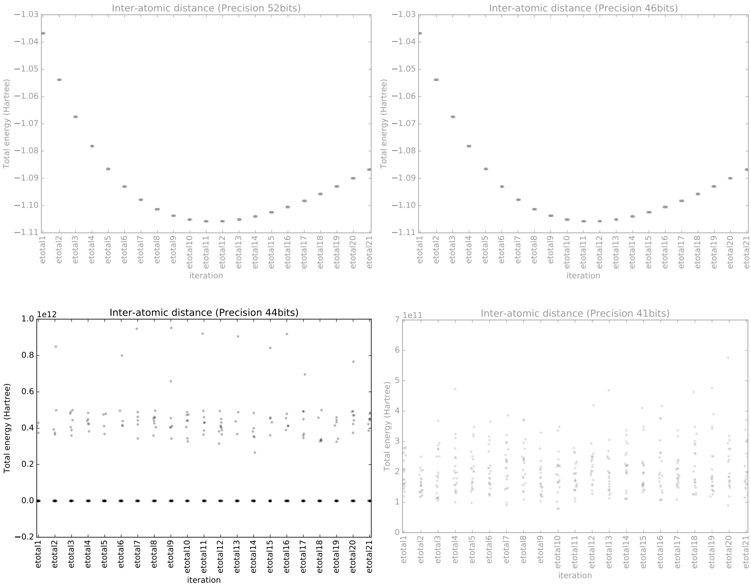




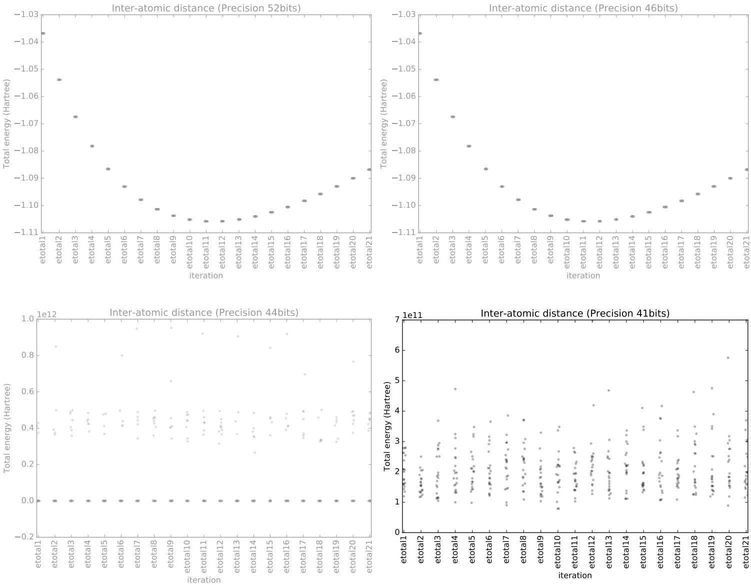




Verificarlo on whole ABINIT - Hydrogen test



Verificarlo on whole ABINIT - Hydrogen test



A more realistic use case

Hydrogen too weak ?

$\rightarrow$ Perovskite ($BaTiO_3$)

- Non-local physic
- Parallelization

Problem:

- Verificarlo is time consuming ($\times$400 overhead)
- Exhaustive analyse of the coupling of functions is impractible: 2^{88} functions to evaluate $\times$ 52 precisions $\times$ 32 samples

Idea:

- Reduce the set of function to test: some function does not impact the final result
- Modify the introduction of error: MCA is costly, low-order model (BITMASK backend)

Numerical sensitive functions

Function that impact the result:

- For each function, plot the minimal precision required to reach machine accuracy
- Only 88 functions among 3400 functions

Functions $<$ 23:

- One third of the functions are below 23bits
- Possible transformation:

double precision $\rightarrow$ single precision
memory scaled down, computation faster

Functions $\geq$ 23:

- Function requiring above 23bits are more sensible
- Require a fine-grained analysis

Conclusion & Future Prospects

Future prospects

- Reproduce with stochastic errors "realife" bugs such as when vectorizing
- Find smart coupling exploration to find errors correlation
- Build an errors graph-propagation model between functions
- Explore benefits of optimizations such as mixed-precision and approximate computing

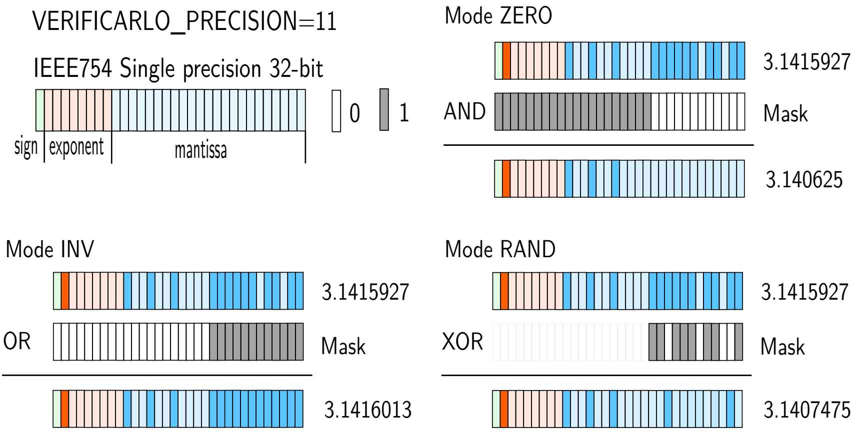
Conclusion

Verificarlo

https://github.com/verificarlo/verificarlo

- Verificarlo: a tool for automatically detect numerical instability
- Exposes noise tolerance and significant digits in number of bits
- Pinpoits functions causing global errors
- Reveals possible optimizations such as precision reduction

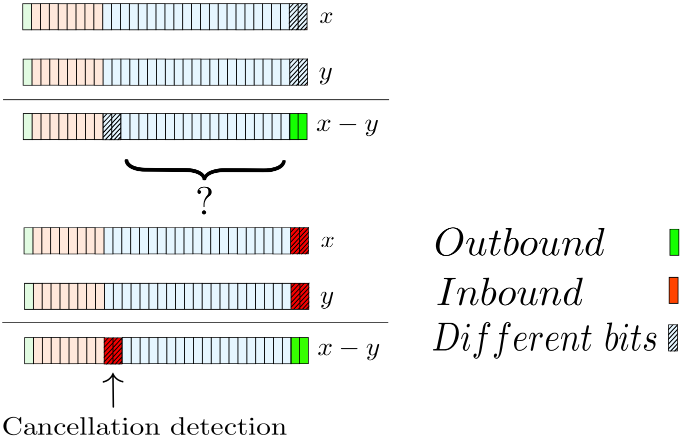
Backend BITMASK



Inbound / Outbound

$$x \sim y$$

$$mca(x) = round(inexact(inexact(x) - inexact(y)))$$



Sensitive functions - 1/6

Function name	Minimal precision
bound_	1
invcb_	1
getng_	5
prcref_	7
___m_lobpcgwf_MOD_lobpcgwf2	8
___m_pawfgr_MOD_pawfgr_init	10
xcmult_	11
initro_	12
___m_fftcore_MOD_kpgsph	13
___m_xg_MOD_xgblock_colwisecaxmy	13
moddiel_	13
getcut_	14
pawmkrho_	14
prep_fourwf_	14

Sensitive functions - 2/6

Function name	Minimal precision
___m_pawrhoij_MOD_symrhoij	16
dotprodm_v_	18
pawmknhat_	18
xcpot_	18
___m_pawxc_MOD_pawxcsum	19
symrhg_	19
newrho_	20
pawaccrhoij_	21
___m_lobpcgwf_MOD_getghc_gsc	22
___m_paw_sphharm_MOD_initylmr	22
___m_paw_numeric_MOD_paw_spline	23
invars2_	23
scfopt_	23
___m_special_funcs_MOD_abi_derfc	26
___m_xg_MOD_xgblock_add	26

Sensitive functions - 3/6

Function name	Minimal precision
getghc_	26
mkffnl_	26
__m_splines_MOD_splfit	27
__m_opernl_ylm_MOD_opernlb_ylm	28
__m_paw_numeric_MOD_paw_jbessel_4spline	28
__m_pawpsp_MOD_pawpsp_cg	28
pawdensities_	28
ph1d3d_	28
xcden_	28
__m_opernl_ylm_MOD_opernlc_ylm	29
__m_sgfft_MOD_sg_fftpx	29
__m_paw_sphharm_MOD_ass_leg_pol	30
__m_pawdij_MOD_pawdijhartree	30
__m_pawpsp_MOD_pawpsp_nl	30
__m_opernl_ylm_MOD_opernla_ylm	31

Sensitive functions - 4/6

Function name	Minimal precision
___m_pawdij_MOD_pawdijxcm	31
___m_sgfft_MOD_fftrisc_one_nothreadsafe	31
___m_special_funcs_MOD_phim	31
___m_paw_finegrid_MOD_pawrfgd_fft	32
___m_pawang_MOD_realgaunt	32
___m_pawdij_MOD_pawdij	32
___m_pawdij_MOD_symdij	32
___m_sgfft_MOD_sg_ctrig	32
___m_sgfft_MOD_sg_fftx	32
___m_sgfft_MOD_sg_ffty	32
mkkin_	32
scfcv_	32
___m_paw_atom_MOD_atompaw_vhnzc	33
___m_pawang_MOD_gaunt	33
___m_pawrad_MOD_nderiv_gen	33

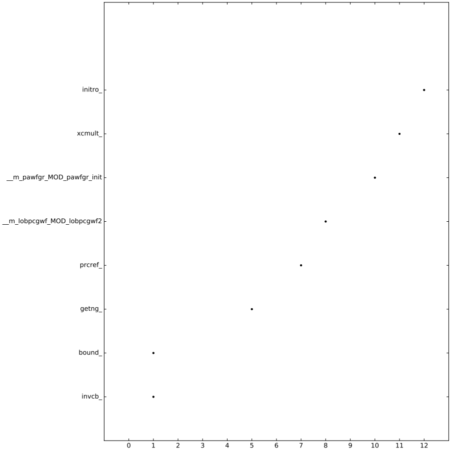
Sensitive functions - 5/6

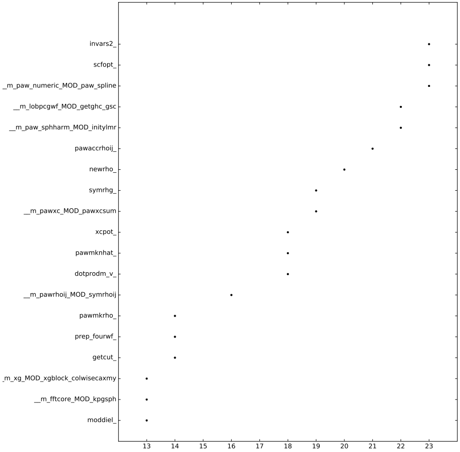
Function name	Minimal precision
getph_	33
pspcor_	33
rhotov_	33
__m_paw_atom_MOD_atompaw_shpfun	34
__m_pawxc_MOD_pawxcsph	34
__m_sgfft_MOD_sg_fft_rc	34
__m_sgfft_MOD_sg_fftz	34
metric_	34
pawinit_	34
__m_paw_atom_MOD_atompaw_dij0	35
__m_pawrad_MOD_nderiv_lin	35
atm2fft_	35
hartre_	35
__m_paw_atom_MOD_atompaw_shapebes	36
__m_paw_numeric_MOD_paw_jbessel	36

Sensitive functions - 6/6

Function name	Minimal precision
__m_pawpsp_MOD_pawpsp_lo	36
__m_pawdij_MOD_pawdijhat	37
__m_pawpsp_MOD_pawpsp_17in	37
__m_pawxc_MOD_pawxcm	37
etotfor_	37
pawdenpot_	37
vtorho_	37
xcpbe_	37
__m_pawrad_MOD_poisson	40
__m_pawrad_MOD_simp_gen	40
dotprod_vn_	40
rhohxc_	40
__m_ewald_MOD_ewald	44

Sensitive functions - 1/2





Sensitive functions - 2/2