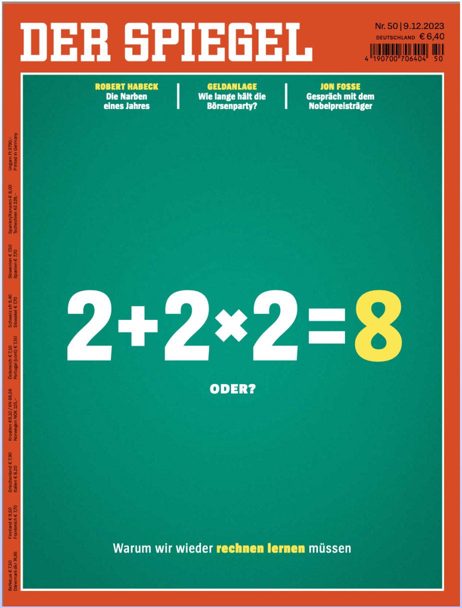
A tool-driven approach toward mixed-precision and sustainable solvers for CFD codes

Roman Iakymchuk[1]
joint work with Pablo Oliveira (Paris-Saclay)

Umeå University, Sweden
riakymch@cs.umu.se

CEEC Community Workshop
Energy-Efficient, Fault Resilient, and Scalable Solvers for CFD Codes
FAU, Erlangen, Germany
December 13th, 2023

[1]This research is partially funded by EuroHPC JU CoE CEEC (No. 101093393)


DER SPIEGEL
Nr. 50 | 9.12.2023
DEUTSCHLAND € 6,40
ROBERT HABECK
Die Narben eines Jahres
GELDANLAGE
Wie lange hält die Börsenparty?
JON FOSSE
Gespräch mit dem Nobelpreisträger
2+2×2=8
ODER?
Warum wir wieder rechnen lernen müssen

Numbers and counting

- Kids often count: one, two, three, many

Numbers and counting

- Kids often count: one, two, three, many
- We are used to the 10-base system

Source: Wikipedia

Numbers and counting

- Kids often count: one, two, three, many
- We are used to the 10-base system

Source: Wikipedia

- 20-base system was also common, e.g. Maya, and is still in use in France and Denmark
- But, computers use binary system

Outline

Binary numbers

- Computers usually store numbers in binary form:

$$(\overbrace{1101}^{\text{4 bit}})_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = (13)_{10}$$

- Fractional binary numbers:

$$\begin{aligned}(.1101)_2 &= 1 \cdot 2^{-1} + 1 \cdot 2^{-2} + 0 \cdot 2^{-3} + 1 \cdot 2^{-4} \\ &= \frac{1}{2} + \frac{1}{4} + 0 + \frac{1}{16} = \frac{13}{16} = (0.8125)_{10}\end{aligned}$$

- *Note:* The decimal fractions 0.1, 0.2, 0.3, 0.4, 0.6, 0.7, 0.8, 0.9 cannot be exactly represented as a fractional binary number!
  - But $(0.125)_{10} = \left(\frac{1}{8}\right)_{10} = 10^0 \cdot 2^{-3} = (.001)_2$ is fine.

Floating-point arithmetic (1/2)

- Computer arithmetic approximates real numbers with finite formats

Floating-point arithmetic (1/2)

- Computer arithmetic approximates real numbers with finite formats

- Floating-point operations $(+, \times)$ are commutative but non-associative
  - $(a+b)+c \neq a+(b+c)$
  - $(-1+1)+2^{-53} \neq -1+(1+2^{-53})$ in double precision

$$1 = (1.\underbrace{0000000000000000000000000000000000000000000000000000}_{1\ldots52})_2$$

$$2^{-53} = (0.\underbrace{0000000000000000000000000000000000000000000000000000}_{1\ldots52}1)_2$$

Floating-point arithmetic (1/2)

- Computer arithmetic approximates real numbers with finite formats

- Floating-point operations ($+, \times$) are commutative but non-associative
  - $(a + b) + c \neq a + (b + c)$
  - $(-1 + 1) + 2^{-53} \neq -1 + (1 + 2^{-53})$ in double precision

$$1 = (1.\underbrace{0000000000000000000000000000000000000000000000000000}_{1 \ldots 52})_2$$

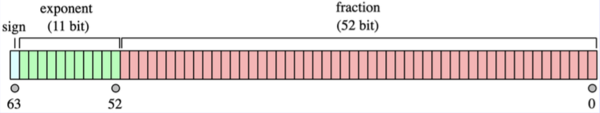
$$2^{-53} = (0.\underbrace{0000000000000000000000000000000000000000000000000000}_{1 \ldots 52}1)_2$$

- Another example is summation in ascending or descending orders
- Consequence: **results and accuracy** of floating-point computations depend on the order of computation especially in parallel

Floating-point arithmetic (2/2)

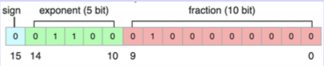
Almost all computer hardware and software support the **IEEE Standard for Floating-Point Arithmetic** IEEE 754-2019

- double - binary64

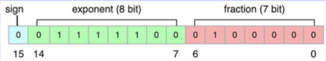


- single - binary32

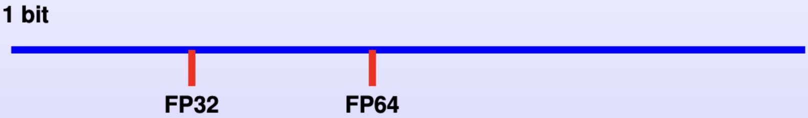
- half - binary16



- bfloat

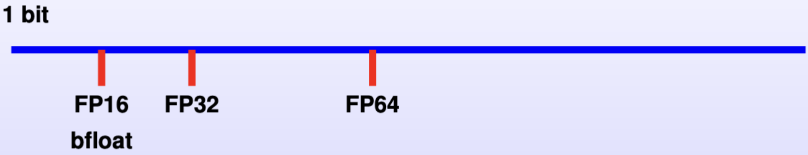


Mixed-precision arithmetic



- double-single plus iterative refinement

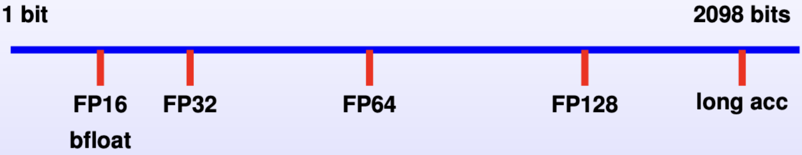
Mixed-precision arithmetic



- double-single plus iterative refinement
- double-single-half/ bfloat
- Over 100 works on mixed precision [a]

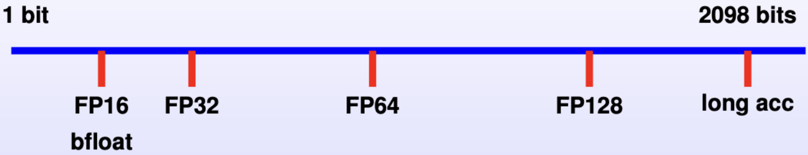
[a] Nicholas Higham and Théo Mary. 'Mixed Precision Algorithms in Numerical Linear Algebra'

Mixed-precision arithmetic



- double-single plus iterative refinement
- double-single-half/ bfloat
- Over 100 works on mixed precision
- Extending precision for exact computations: double plus FPEs or double plus long accumulator

Mixed-precision arithmetic



- double-single plus iterative refinement
- double-single-half/ bfloat
- Over 100 works on mixed precision
- Extending precision for exact computations: double plus FPEs or double plus long accumulator
- **Mixed-precision algorithm** is an algorithm that carefully, effectively and safely combines multiple precisions

Sustainable HPC → Energy-efficient HPC

- **Energy-efficient architectures** such as graphic processors (GPUs) and FPGAs – Green HPC computing
- PDC@KTH extracts the produced heat to **warm up the main campus**
- CSCS at Switzerland proposes **‘free cooling’** with the water from the lake of Lugano

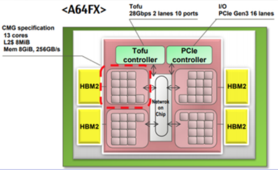
Precision & Sustainability

Exascale computing and linear algebra

- Exascale computing is constrained by **power consumption**

→ Power-efficient hardware

  - RIKEN's Fugaku w A64FX (FP64:FP32:FP16 = 1:2:4)
  - EPI (ARM, FPGA, RISC-V)



Source: Fujitsu

- Linear algebra is known to be dominant by **double precision**

→ Sustainable algorithms

math **Mixed and adaptive precision computing**

code **Communication hiding or avoiding**

tools **Numerical abnormalities and precision cropping**

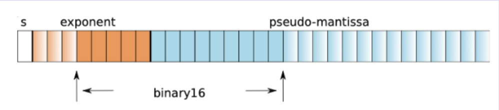
Sustainable algorithmic solutions

Inspection w tools $\rightarrow$ Strategy $\rightarrow$ Revision of algorithms

1. **Arithmetic tool** applied to **code** $\rightarrow$ manual/ automatic
2. if the reduction is possible, derive algorithmic solutions
3. conduct probabilistic (aka optimistic) error analysis
   - error bound with constant $\sqrt{n}\mu$ with high probability
4. implement on hardware with stochastic rounding support – randomly maps x to one of two bounds

Analysis with tools: VerifiCarlo

- Verificarlo – an automatic tool for debugging and assessing FP precision based on Monte Carlo Arithmetic
- Backends: debugging (MCA) and mixed-precision (Vprec)
- Eg setting $r = 5$ and $p = 10$, Vprec simulates a `binary16` embedded inside a `binary32`



- More details: `https://github.com/verificarlo` and InterFLOP project `https://www.interflop.fr/`

VerifiCarlo-Vprec Example

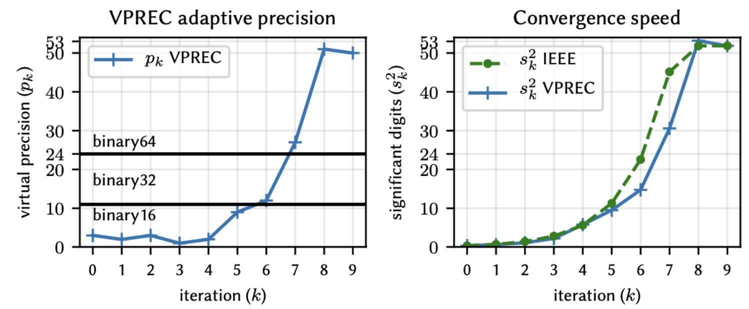
k	x_{k+1}	s_k^{10}	s_k^2
0	0.0690266447076745	0.11	0.37
1	0.1230846130203958	0.21	0.70
2	0.1985746566605835	0.43	1.43
3	0.2732703639721015	0.84	2.79
4	0.3119369815109966	1.79	5.95
5	0.3181822938100336	3.40	11.3
6	0.3183098350392471	6.79	22.6
7	0.3183098861837825	13.6	45.2
8	0.3183098861837907	15.6	51.8
9	0.3183098861837907	15.6	51.8

```
double newton(double x0) {
  double x_k, x_k1=x0, b=PI;
  do {
    x_k  = x_k1;
    x_k1 = x_k*(2-b*x_k);
  }while (fabs((x_k1-x_k)/x_k)
  >= 1e-15);
  return x_k1;
}
```

The Newton-Raphson method for inverse of π[a]

[a] Pablo Oliveira et al. *Automatic exploration of reduced floating-point representations in iterative methods.* Euro-Par 2019

VerifiCarlo-Vprec Example



The Newton-Raphson method for inverse of π[a]

[a]Pablo Oliveira et al. *Automatic exploration of reduced floating-point representations in iterative methods.* Euro-Par 2019

Test cases

- **NEK5000** is a high order, incompressible Navier-Stokes solver based on the spectral element method

→ **Nekbone** solves a Poisson equation using a Conjugate Gradient method with a simple or spectral element multigrid preconditioner

- **FLEXI** is high-order accurate, open source solver for general PDEs of hyperbolic/parabolic-type based on the DG-SEM

Nekbone w `Vprec`

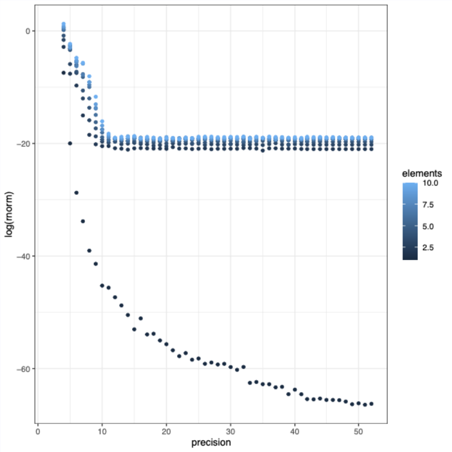
Basic example w/o preconditioner

$$\boxed{Ax = b}$$

while $(\tau > \tau_{\max})$

Step	Operation
$S1:$	$w := Ad$
$S2:$	$\rho := \beta / < d, w >$
$S3:$	$x := x + \rho d$
$S4:$	$r := r - \rho w$
$S5:$	$z := M^{-1} r$
$S6:$	$\beta := < z, r >$
$S7:$	$d := (\beta / \beta_{old}) d + z$
$S8:$	$\tau := < r, r >$

end while



Nekbone w `Vprec`

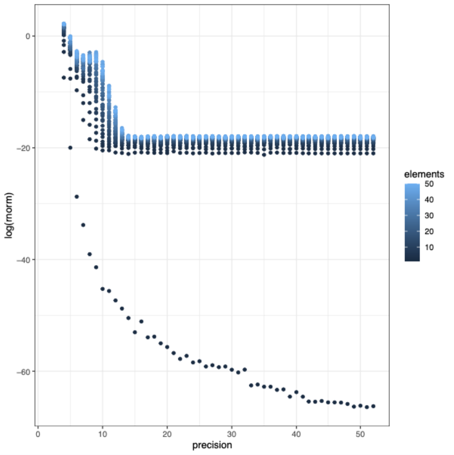
Multigrid Preconditioner Example

$$\boxed{Ax = b}$$

while $(\tau > \tau_{\max})$

Step	Operation
$S1:$	$w := Ad$
$S2:$	$\rho := \beta / < d, w >$
$S3:$	$x := x + \rho d$
$S4:$	$r := r - \rho w$
$S5:$	$z := M^{-1} r$
$S6:$	$\beta := < z, r >$
$S7:$	$d := (\beta / \beta_{old}) d + z$
$S8:$	$\tau := < r, r >$

end while



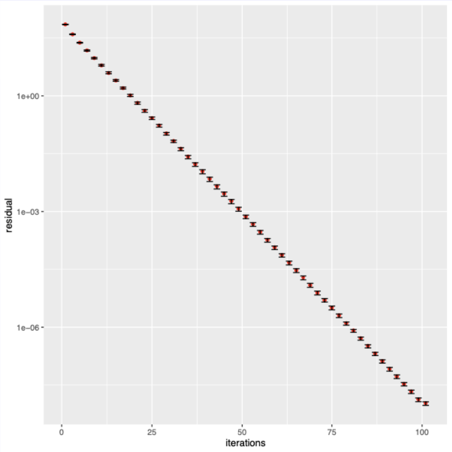
- **20 MCA samples** for the previously found Vprec configuration
- simulate `binary32`

$$\boxed{Ax = b}$$

while $(\tau > \tau_{\max})$

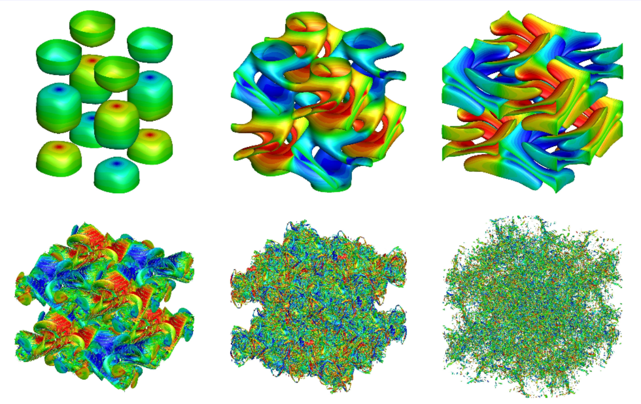
Step	Operation
$S1:$	$w := Ad$
$S2:$	$\rho := \beta / < d, w >$
$S3:$	$x := x + \rho d$
$S4:$	$r := r - \rho w$
$S5:$	$z := M^{-1} r$
$S6:$	$\beta := < z, r >$
$S7:$	$d := (\beta / \beta_{old}) d + z$
$S8:$	$\tau := < r, r >$

end while



FLEXI

- FLEXI finds the spatial solution of the Navier-Stokes equations and performs the time integration $\rightarrow$ a simple ODE in the form of $U_t = R(U)$
- The ODE in time is solved with **explicit Runge-Kutta**



Temporal evolution of the Taylor-Green vortex. Shown are contours of vorticity.
Courtesy of Andrea Beck

FLEXI results

First attempt

- Instrument the entire FLEXI application
- Run tests for various precisions 53, 52, ..., 20 bits for short time
- “ERROR: Legendre Gauss nodes could not be computed up to desired precision. Code stopped!"
  - $\rightarrow$ construction of the bases requires high precision

FLEXI results

First attempt

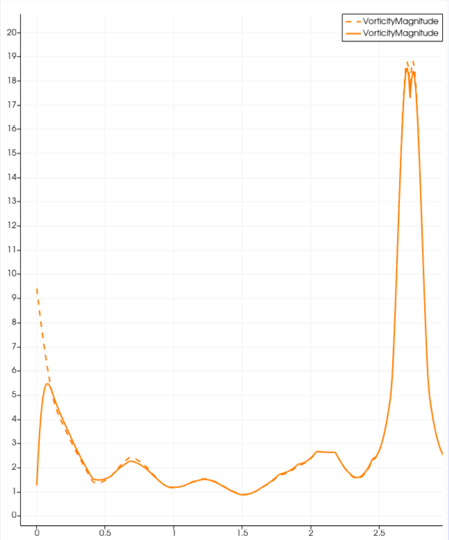
- Instrument the entire FLEXI application
- Run tests for various precisions 53, 52, ..., 20 bits for short time
- "ERROR: Legendre Gauss nodes could not be computed up to desired precision. Code stopped!"
  - $\rightarrow$ construction of the bases requires high precision

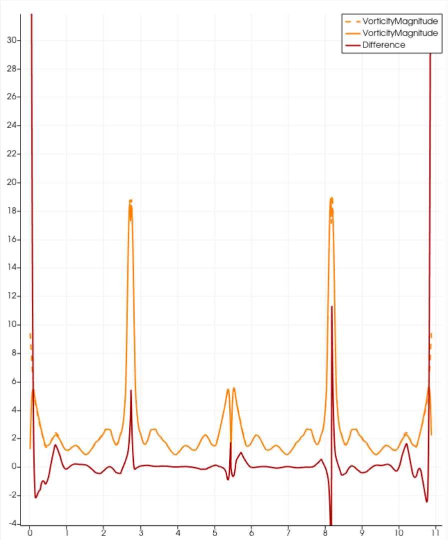
Refined attempt

- Focus on the compute heavy function that calculates the weak DG-SEM space operator from surface, volume and source contributions

FLEXI results

Focus on **velocity magnitude**: solid line is original in FP64 while dashed is VPREC w 23bits





Summary

- mixed-precision algorithmic solutions is a way toward sustainable computing
- computer arithmetic tools help to estimate precision needs
  - Per case base and even mixed-precision binary
- numerical techniques help to craft robust and adaptive solvers

Summary

- mixed-precision algorithmic solutions is a way toward sustainable computing
- computer arithmetic tools help to estimate precision needs
  - Per case base and even mixed-precision binary
- numerical techniques help to craft robust and adaptive solvers
- employ computer arithmetic tools for[a]
  - numerical abnormalities detection
  - precision inspection
  - numerical CI

[a] InterFLOP: `https://www.interflop.fr/`

Summary

- mixed-precision algorithmic solutions is a way toward sustainable computing
- computer arithmetic tools help to estimate precision needs
  - Per case base and even mixed-precision binary
- numerical techniques help to craft robust and adaptive solvers
- employ computer arithmetic tools for[a]
  - numerical abnormalities detection
  - precision inspection
  - numerical CI

[a] InterFLOP: `https://www.interflop.fr/`

Thank you for your attention !