Sharper and smaller error bounds for low precision scientific computing

Theo Mary, joint work with Nick Higham
University of Manchester, School of Mathematics

PEQUAN seminar, LIP6, Paris, 8 October 2019

M\crNA
Manchester Numerical Analysis

A quote from 2002

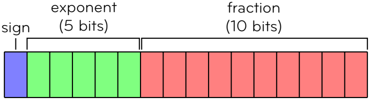
> *The constant terms in an error bound are the least important parts of error analysis. It is not worth spending much effort to minimize constants because the achievable improvements are usually insignificant.*
>
> *Nick Higham, ASNA 2ed (2002)*

Today: low precision arithmetics

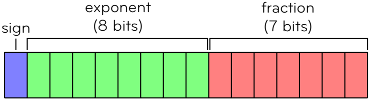
Type		Bits	Range	$u = 2^{-t}$
fp64	double	64	$10^{\pm 308}$	$2^{-53} \approx 1 \times 10^{-16}$
fp32	single	32	$10^{\pm 38}$	$2^{-24} \approx 6 \times 10^{-8}$
fp16	half	16	$10^{\pm 5}$	$2^{-11} \approx 5 \times 10^{-4}$
bfloat16	half	16	$10^{\pm 38}$	$2^{-8} \approx 4 \times 10^{-3}$

Half precision increasingly **supported by hardware**:

- Present: **NVIDIA** Pascal & Volta GPUs, **AMD** Radeon Instinct MI25 GPU, **Google** TPU, **ARM** NEON
- Near future: Fujitsu A64FX ARM, **IBM** AI chips, **Intel** Xeon Cooper Lake and Intel Nervana Neural Network



fp16



bfloat16

Today: low precision arithmetics

Type		Bits	Range	$u = 2^{-t}$
fp64	double	64	$10^{\pm 308}$	$2^{-53} \approx 1 \times 10^{-16}$
fp32	single	32	$10^{\pm 38}$	$2^{-24} \approx 6 \times 10^{-8}$
fp16	half	16	$10^{\pm 5}$	$2^{-11} \approx 5 \times 10^{-4}$
bfloat16	half	16	$10^{\pm 38}$	$2^{-8} \approx 4 \times 10^{-3}$

Designed for machine learning but offer interesting **opportunities for scientific computing**:

- Faster flops
- Less storage and communications
- Lower energy consumption

But need to deal with

- **Reduced range** (fp16)
- **Reduced precision** (large u)

Summation

Summation $s = \sum_{i=1}^{n} x_i$ is at the heart of NLA:

- Inner products $a^T b = \sum_{i=1}^{n} a_i b_i$
- Matrix–vector/matrix products $\equiv$ multiple inner products
- LU factorization and linear systems: $y = c - (\sum_{i=1}^{k-1} a_i b_i)/b_k$

Summation

Summation $s = \sum_{i=1}^{n} x_i$ is at the heart of NLA:

- Inner products $a^T b = \sum_{i=1}^{n} a_i b_i$
- Matrix–vector/matrix products $\equiv$ multiple inner products
- LU factorization and linear systems: $y = c - (\sum_{i=1}^{k-1} a_i b_i)/b_k$

Backward error analysis:

$$\widehat{s}_2 = (x_1 + x_2)(1 + \delta_2)$$

$$\widehat{s}_k = (\widehat{s}_{k-1} + x_k)(1 + \delta_k) = x_1 \prod_{j=2}^{k} (1 + \delta_j) + \ldots + x_k(1 + \delta_k)$$

$$\widehat{s}_n = \widehat{s} = \sum_{i=1}^{n} x_i \prod_{j=i}^{n} (1 + \delta_j), \qquad |\delta_j| \le u \qquad\qquad (\delta_1 := 0)$$

Summation

Summation $s = \sum_{i=1}^{n} x_i$ is at the heart of NLA:

- Inner products $a^T b = \sum_{i=1}^{n} a_i b_i$
- Matrix–vector/matrix products $\equiv$ multiple inner products
- LU factorization and linear systems: $y = c - (\sum_{i=1}^{k-1} a_i b_i)/b_k$

Backward error analysis:

$$\begin{aligned}
\widehat{s}_2 &= (x_1 + x_2)(1+\delta_2) \\
\widehat{s}_k &= (\widehat{s}_{k-1} + x_k)(1+\delta_k) = x_1 \prod_{j=2}^{k}(1+\delta_j) + \ldots + x_k(1+\delta_k) \\
\widehat{s}_n &= \widehat{s} = \sum_{i=1}^{n} x_i \prod_{j=i}^{n}(1+\delta_j), \qquad |\delta_j| \le u \qquad\qquad (\delta_1 := 0)
\end{aligned}$$

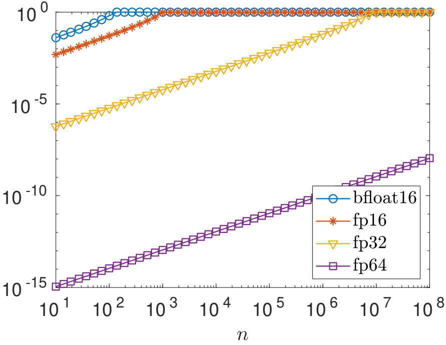
Fundamental lemma in backward error analysis

If $|\delta_i| \le u$ for $i = 1:n$ and $nu < 1$, then

$$\prod_{i=1}^{n}(1+\delta_i) = 1 + \theta_n, \quad |\theta_n| \le \gamma_n := \frac{nu}{1-nu} = nu + O(u^2)$$

With low precisions, backward stability is lost

Most backward error bounds in scientific computing $\propto \gamma_n \equiv nu$



In half precision, not a single correct digit guaranteed when $n > 1024$ (fp16) or $n > 128$ (bfloat16)

Classical algorithms can no longer be considered “backward stable”!

The emergence of low precisions has created a need for

- **Sharper bounds**, to maintain backward stability guarantees
- **Smaller bounds**, ideally $\propto cu$, for some modest $c = O(1)$
- Both important, as **sharp** $+$ **small bound** $\Rightarrow$ **small error**

Part 1: sharper bounds

- Traditional worst-case bounds are typically pessimistic because of statistical effects on the rounding errors

- Consider $E = \sum_{i=1}^{n} \delta_j$ for random independent δ_j of mean zero
$\Rightarrow$ central limit theorem: for $n \to \infty$, $E/\sqrt{n} \sim \mathcal{N}(0, u)$

> *In general, the statistical distribution of the rounding errors will reduce considerably the function of n occurring in the relative errors. We might expect in each case that this function should be replaced by something which is no bigger than its square root.*
>
> — *James Wilkinson, 1961*

Can we rigorously prove this rule of thumb for a wide class of algorithms?

Probabilistic model on the rounding errors

We seek an anologous result to the fundamental lemma by using the following model

Probabilistic model of rounding errors

In the computation of interest, the quantities δ in the model

$$\mathrm{fl}(a \text{ op } b) = (a \text{ op } b)(1+\delta), \quad |\delta| \le u, \quad \text{op} \in \{+,-,\times,/\}$$

associated with every pair of operands are independent random variables of mean zero.

> *There is no claim that ordinary rounding and chopping are random processes, or that successive errors are independent.* ***The question to be decided is whether or not these particular probabilistic models of the processes will adequately describe what actually happens.***
>
> *— Hull and Swenson, 1966*

Probabilistic backward error analysis: proof sketch

First step: transform the product in a sum by taking the logarithm

$$S = \log \prod_{i=1}^{n} (1+\delta_i) = \sum_{i=1}^{n} \log(1+\delta_i)$$

Probabilistic backward error analysis: proof sketch

First step: transform the product in a sum by taking the logarithm

$$S = \log \prod_{i=1}^{n} (1+\delta_i) = \sum_{i=1}^{n} \log(1+\delta_i)$$

Second step: apply Hoeffding's concentration inequality:

Hoeffding's inequality

Let X_1, ..., X_n be random independent variables satisfying $|X_i| \le c$. Then the sum $S = \sum_{i=1}^{n} X_i$ satisfies

$$\Pr(|S - \mathbb{E}(S)| \ge \lambda\sqrt{n}c) \le 2\exp(-\lambda^2/2)$$

to $X_i = \log(1+\delta_i)$ $\Rightarrow$ requires bounding $\log(1+\delta_i)$ and $\mathbb{E}\left(\log(1+\delta_i)\right)$ using Taylor expansions

Probabilistic backward error analysis: proof sketch

First step: transform the product in a sum by taking the logarithm

$$S = \log \prod_{i=1}^{n} (1+\delta_i) = \sum_{i=1}^{n} \log(1+\delta_i)$$

Second step: apply Hoeffding's concentration inequality:

Hoeffding's inequality

Let X_1, ..., X_n be random independent variables satisfying $|X_i| \le c$. Then the sum $S = \sum_{i=1}^{n} X_i$ satisfies

$$\Pr(|S - \mathbb{E}(S)| \ge \lambda\sqrt{n}c) \le 2\exp(-\lambda^2/2)$$

to $X_i = \log(1+\delta_i)$ ⇒ requires bounding $\log(1+\delta_i)$ and $\mathbb{E}\left(\log(1+\delta_i)\right)$ using Taylor expansions

Third step: retrieve the result by taking the exponential of S

Probabilistic backward error analysis: main result

Main result

Let δ_i, $i = 1 : n$, be independent random variables of mean zero such that $|\delta_i| \le u$. Then, for any constant $\lambda > 0$, the relation

$$\prod_{i=1}^{n}(1+\delta_i) = 1 + \theta_n, \quad |\theta_n| \le \widetilde{\gamma}_n(\lambda) := \exp\left(\lambda\sqrt{n}u + \frac{nu^2}{1-u}\right) - 1$$

$$\le \lambda\sqrt{n}u + O(u^2)$$

holds with probability $P(\lambda) = 1 - 2\exp\left(-\lambda^2(1-u)^2/2\right)$

Probabilistic backward error analysis: main result

Main result

Let δ_i, $i = 1:n$, be independent random variables of mean zero such that $|\delta_i| \le u$. Then, for any constant $\lambda > 0$, the relation

$$\prod_{i=1}^{n}(1+\delta_i) = 1+\theta_n, \quad |\theta_n| \le \widetilde{\gamma}_n(\lambda) := \exp\left(\lambda\sqrt{n}u + \frac{nu^2}{1-u}\right) - 1$$
$$\le \lambda\sqrt{n}u + O(u^2)$$

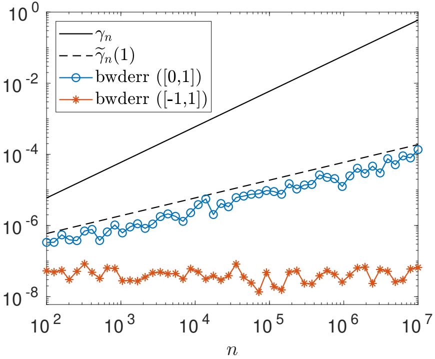
holds with probability $P(\lambda) = 1 - 2\exp\left(-\lambda^2(1-u)^2/2\right)$

Key features:

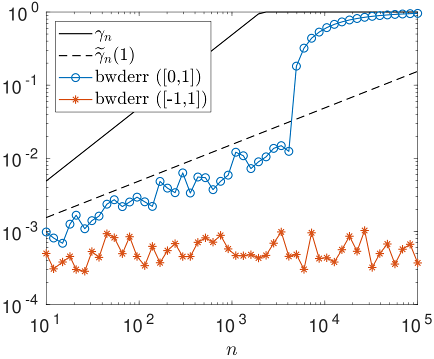
- Exact bound, not first order ($nu < 1$ not required)
- No "$n \to \infty$" assumption (CLT → Hoeffding's inequality)
- Small values of λ suffice: $P(1) \approx 0.73$, $P(5) \ge 1 - 10^{-5}$
- Can be applied **in a nearly systematic way:** $\gamma_n \to \widetilde{\gamma}_n(\lambda)$

Probabilistic backward error analysis: experiments

Single precision



Half precision



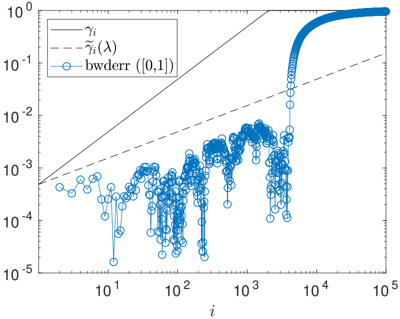
- Able to guarantee backward stability for a wider range of problems in a probabilistic sense
- With half precision and $[0,1]$ data, $\widetilde{\gamma}_n$ is not valid for large n
- Even $\widetilde{\gamma}_n$ is not sharp for $[-1,1]$ data

Stagnation leads to rounding errors with nonzero mean

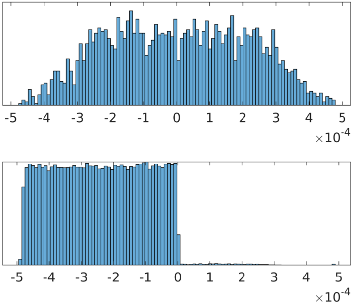
$$s_{i+1} = s_i + x_i \quad \Rightarrow \quad \widehat{s}_{i+1} = (\widehat{s}_i + x_i)(1+\delta_i)$$

Explanation: s_i keeps increasing, at some point, it becomes so large that $\widehat{s}_{i+1} = \widehat{s}_i \Rightarrow \delta_i = -x_i/(\widehat{s}_i + x_i) < 0$

Backward error at step i $\quad \dfrac{|\widehat{s}_i - s_i|}{\sum_{j=1}^{i} x_j}$



Distribution of the δ_i



Top: $1 \le i \le 3000$

Bottom: $3000 \le i \le 10^5$

Another summation error analysis

Recursive summation computes

$$\widehat{s}_{i+1} = (\widehat{s}_i + x_{i+1})(1+\delta_i), \quad i = 1:n \qquad \text{with } s_1 = x_1$$

$$\begin{aligned}\widehat{s} - s = \widehat{s}_n - s_n &= \widehat{s}_{n-1} - s_{n-1} + (\widehat{s}_{n-1} + x_n)\delta_n \\ &= \sum_{i=1}^{n-1} (\widehat{s}_i + x_{i+1})\delta_i = \sum_{i=1}^{n-1} \widehat{s}_{i+1}\delta_i/(1+\delta_i) = \sum_{i=1}^{n-1} s_{i+1}\delta_i + O(u^2)\end{aligned}$$

Another summation error analysis

Recursive summation computes

$$\widehat{s}_{i+1} = (\widehat{s}_i + x_{i+1})(1+\delta_i), \quad i = 1:n \qquad \text{with } s_1 = x_1$$

$$\begin{aligned}\widehat{s} - s = \widehat{s}_n - s_n &= \widehat{s}_{n-1} - s_{n-1} + (\widehat{s}_{n-1} + x_n)\delta_n \\ &= \sum_{i=1}^{n-1} (\widehat{s}_i + x_{i+1})\delta_i = \sum_{i=1}^{n-1} \widehat{s}_{i+1}\delta_i/(1+\delta_i) = \sum_{i=1}^{n-1} s_{i+1}\delta_i + O(u^2)\end{aligned}$$

Oettli-Prager backward error formula:

$$\varepsilon_{bwd} = \frac{|\widehat{s} - s|}{\sum_{i=1}^n |x_i|} = \frac{\left|\sum_{i=1}^{n-1} s_{i+1}\delta_i\right|}{\sum_{i=1}^n |x_i|} + O(u^2)$$

We recover worst-case bound:

$$\varepsilon_{bwd} \le \frac{u\sum_{i=1}^{n-1} |s_{i+1}|}{\sum_{i=1}^n |x_i|} \le \frac{u\sum_{i=1}^{n-1}\sum_{j=1}^{i} |x_j|}{\sum_{i=1}^n |x_i|} \le nu + O(u^2)$$

Probabilistic model of the data

We also recover probabilistic bound by applying

Hoeffding's inequality

Let X_1, ..., X_n be random independent variables satisfying $|X_j| \le c$. Then the sum $S = \sum_{i=1}^{n} X_j$ satisfies

$$\Pr(|S - \mathbb{E}(S)| \ge \lambda\sqrt{n}c) \le 2\exp(-\lambda^2/2)$$

to $X_j = s_{j+1}\delta_j$ with $c = u\sum_{i=1}^{n}|x_i|$

Probabilistic model of the data

We also recover probabilistic bound by applying

Hoeffding's inequality

Let X_1, ..., X_n be random independent variables satisfying $|X_j| \le c$. Then the sum $S = \sum_{i=1}^n X_j$ satisfies

$$\Pr(|S - \mathbb{E}(S)| \ge \lambda\sqrt{n}c) \le 2\exp(-\lambda^2/2)$$

to $X_j = s_{j+1}\delta_j$ with $c = u\sum_{i=1}^n |x_i|$

Our objective now is to obtain a **sharper** bound by taking into account the distribution of the x_i:

Probabilistic model of the data

The x_i, $i = 1 : n$, are independent random variables sampled from a given distribution of mean μ_x and satisfy $|x_i| \le C_x$.

Sharper probabilistic backward error analysis

- Hoeffding 1: $|s_j| \le \mu_x j + \lambda C_x \sqrt{j} \Rightarrow |X_j| \le c = (\mu_x n + \lambda C_x \sqrt{n})u$
- Hoeffding 2: $|\widehat{s} - s| = |\sum_{j=1}^{n-1} X_j| \le \lambda \sqrt{n} c = \big(\lambda \mu_x n^{3/2} + \lambda^2 C_x n\big)u$
- Technical difficulty: $X_j = s_{j+1}\delta_j$ are not independent since $s_j = \sum_{i=1}^{j} x_i$ depend on each other $\Rightarrow$ use martingales
- Hoeffding 3: $\sum_{i=1}^{n} |x_i| \ge n\mu_{|x|} - \lambda C_x \sqrt{n}$

Sharper probabilistic backward error analysis

- Hoeffding 1: $|s_j| \le \mu_x j + \lambda C_x \sqrt{j} \Rightarrow |X_j| \le c = (\mu_x n + \lambda C_x \sqrt{n})u$
- Hoeffding 2: $|\widehat{s} - s| = |\sum_{j=1}^{n-1} X_j| \le \lambda \sqrt{n} c = \big(\lambda \mu_x n^{3/2} + \lambda^2 C_x n\big) u$
- Technical difficulty: $X_j = s_{j+1}\delta_j$ are not independent since $s_j = \sum_{i=1}^{j} x_i$ depend on each other $\Rightarrow$ use martingales
- Hoeffding 3: $\sum_{i=1}^{n} |x_i| \ge n\mu_{|x|} - \lambda C_x \sqrt{n}$

Main result

Under the previously stated models of rounding errors and data,

$$\varepsilon_{bwd} = \frac{|\widehat{s} - s|}{\sum_{i=1}^{n} |x_i|} \le \frac{\lambda \mu_x \sqrt{n} + \lambda^2 C_x}{\mu_{|x|} - \lambda C_x / \sqrt{n}} \cdot u + O(u^2)$$

holds with probability $P(\lambda) = 1 - 2(n+1)\exp\left(-\lambda^2/2\right)$

- $\mu_x = O(1) \Rightarrow \varepsilon_{bwd} = O(\sqrt{n}u)$
- $\mu_x = 0$ or $\mu_x \ll 1 \Rightarrow \varepsilon_{bwd} = O(u)$

Sharper bounds: summary

	General δ_i	Probabilistic model on δ_i		
		General x_i	Probabilistic model on x_i	
			$\mu_x \neq 0$	$\mu_x = 0$
Backward	nu	$\sqrt{n}u$	$\sqrt{n}u$	u

By incorporating statistical effects on **both the rounding errors and the data** we obtained **sharp backward error bounds for any data**

Sharper bounds: summary

	General δ_i	Probabilistic model on δ_i		
		General x_i	Probabilistic model on x_i	
			$\mu_x \neq 0$	$\mu_x = 0$
Backward	nu	$\sqrt{nu}$	$\sqrt{nu}$	u
Forward	κnu	$\kappa\sqrt{nu}$	$\kappa\sqrt{nu} \equiv \sqrt{nu}$	$\kappa u \approx \sqrt{nu}$

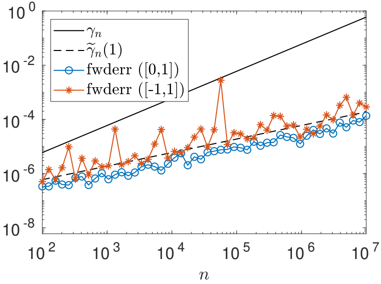
By incorporating statistical effects on **both the rounding errors and the data** we obtained **sharp backward error bounds for any data**

Forward = κ × Backward

$$\kappa = \frac{\sum_{i=1}^n |x_i|}{\left|\sum_{i=1}^n x_i\right|}$$

$\sqrt{nu}$ is still too large for large u and n

$\Rightarrow$ **we do need smaller bounds**



Part 2: smaller bounds

Existing algorithms to avoid error accumulation are expensive. For example, **compensated summation** [Kahan 1965]:

```
$s = 0$; $e = 0$;
**for** $i = 1 : n$ **do**
    $y = x_i + e$;
    $t = s$;    $s = t + y$;
    $e = (t - s) + y$;
**end for**
```

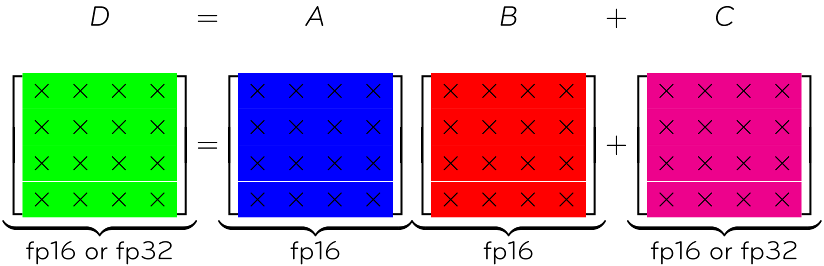
yields an **error bound** $2u$ but is $4\times$ **more expensive**

$\Rightarrow$ Not suited for low precisions: simply using higher precision would be cheaper!

Can we design more accurate algorithms while preserving high performance?

NVIDIA tensor cores (joint work with P. Blanchard, F. Lopez, S. Pranesh)

4×4 matrix multiplication **in 1 clock cycle**:



- Possibly, this is a **block fused multiply-add** (FMA): only one rounding error per element: $\widehat{D} = \mathrm{fl}_{16}(D)$ or $\mathrm{fl}_{32}(D)$
- Algorithms now become intrinsically **mixed precision**—and more complicated to analyze

Tensor cores: error analysis

Let $A, B \in \mathbb{R}^{n\times n}$. Computing $C = AB$ with a block FMA yields, for any row x of A and any column y of B

$$\widehat{s} = (x_1y_1 + \ldots + x_4y_4)\prod_{j=1}^{n/4}(1+\delta_j) \ + \ \ldots \ + \ (x_{n-3}y_{n-3} + \ldots + x_ny_n)(1+\delta_{n/4})$$

$$|\widehat{C} - C| \le \gamma_{n/4}^{FMA}|A||B|, \qquad u_{FMA} = u_{16} \text{ or } u_{32}$$

Tensor cores: error analysis

Let $A, B \in \mathbb{R}^{n\times n}$. Computing $C = AB$ with a block FMA yields, for any row x of A and any column y of B

$$\widehat{s} = (x_1y_1+\ldots+x_4y_4)\prod_{j=1}^{n/4}(1+\delta_j) \; + \; \ldots \; + \; (x_{n-3}y_{n-3}+\ldots+x_ny_n)(1+\delta_{n/4})$$

$$|\widehat{C}-C| \le \gamma_{n/4}^{FMA}|A||B|, \qquad u_{FMA} = u_{16} \text{ or } u_{32}$$

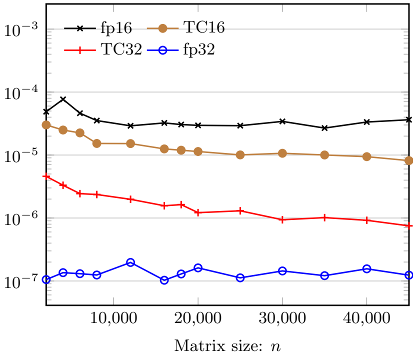
Standard fp16	Tensor core TC16	Tensor core TC32	Standard fp32
$(n+2)u_{16}$	$(n/4+2)u_{16}$	$2u_{16}+nu_{32}/4$	nu_{32}

- **fp16 → TC16**: factor 4 reduction thanks to block FMA
- **TC16 → TC32**: factor $n/8$ reduction by accumulating in fp32
- **TC32 → fp32**: in theory, reduction only if n is small

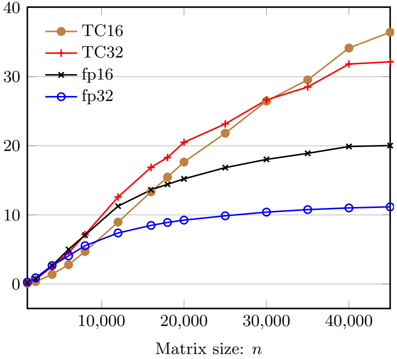
Tensor cores: experiments with LU factorization

Should we accumulate in single (**TC32**) or half (**TC16**) precision?

Backward error



Performance (TFlops/s)



- TC32 almost as fast as TC16, and much more accurate
- fp32 remains more accurate than TC32 in practice, but only by $\sim$ an order of magnitude

The FABsum algorithm (joint work with P. Blanchard)

Classical Blocked summation algorithm:

```
for $i = 1\colon n/b$ do
    Compute $s_i = \sum_{j=(i-1)b+1}^{ib} x_j$.
end for
Compute $s = \sum_{i=1}^{n/b} s_i$.
```

$$\underbrace{\underbrace{x_1 \cdots \; x_b}_{s_1} \quad \underbrace{\cdots\cdots\cdots\cdots}_{\cdots} \quad \underbrace{\cdots\cdots\cdots\cdots}_{\cdots} \quad \underbrace{\cdots\cdots\cdots\cdots}_{\cdots} \quad \underbrace{x_{n-b+1} \;\; \cdots \;\; x_n}_{s_{n/b}}}_{s}$$

- Widely used in NLA libraries (BLAS, LAPACK, ...)
- Error bound $nu \to (b + n/b)u$

The FABsum algorithm (joint work with P. Blanchard)

Fast Accurate Blocked summation algorithm (FABsum):

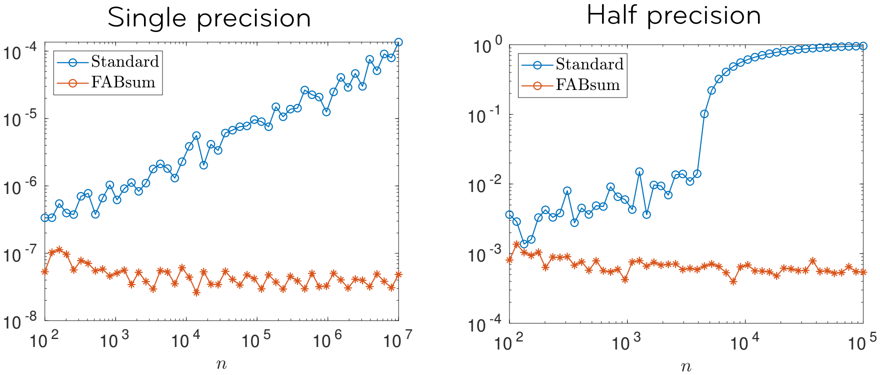
```
for $i = 1 \colon n/b$ do
    Compute $s_i = \sum_{j=(i-1)b+1}^{ib} x_j$ with FastSum.
end for
Compute $s = \sum_{i=1}^{n/b} s_i$ with AccurateSum.
```

$$\underbrace{\underbrace{x_1 \cdots x_b}_{s_1} \quad \underbrace{\cdots\cdots\cdots}_{\cdots} \quad \underbrace{\cdots\cdots\cdots}_{\cdots} \quad \underbrace{\cdots\cdots\cdots}_{\cdots} \quad \underbrace{x_{n-b+1} \cdots x_n}_{s_{n/b}}}_{s}$$

- Widely used in NLA libraries (BLAS, LAPACK, ...)
- Error bound nu → $(b + n/b)u$ → bu with FABsum
- Only $(1 + 1/b)\times$ **more expensive**

FABsum: numerical results

Backward error (for $[0, 1]$ data)


Single precision

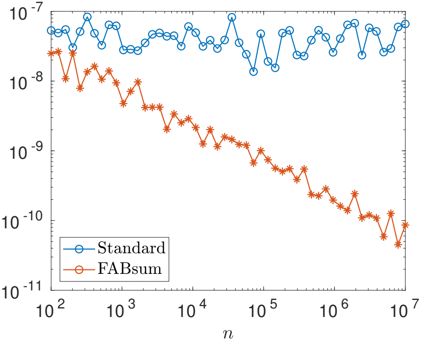
Half precision

- Implementation in **multicore library PLASMA** achieves high performance (less than 5% overhead)

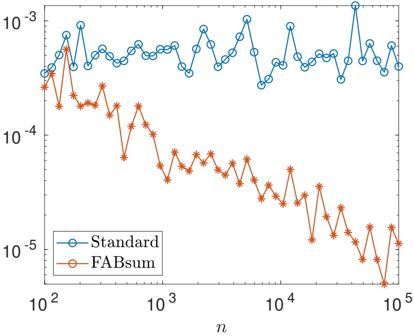
FABsum: numerical results

Backward error (for $[-1, 1]$ data)

Single precision



Half precision



- Implementation in **multicore library PLASMA** achieves high performance (less than 5% overhead)

One more idea: zeroing the summands mean

Idea: given x_i of mean $\mu_x \neq 0$, let $y_i = x_i - \mu_x$ and compute

$$s = \sum_{i=1}^{n} y_i + n\mu_x$$

$$\frac{|\widehat{s} - s|}{\sum_{i=1}^{n} |x_i|} \propto O(\sqrt{n}\mu_y u) + O(u) = O(u)$$

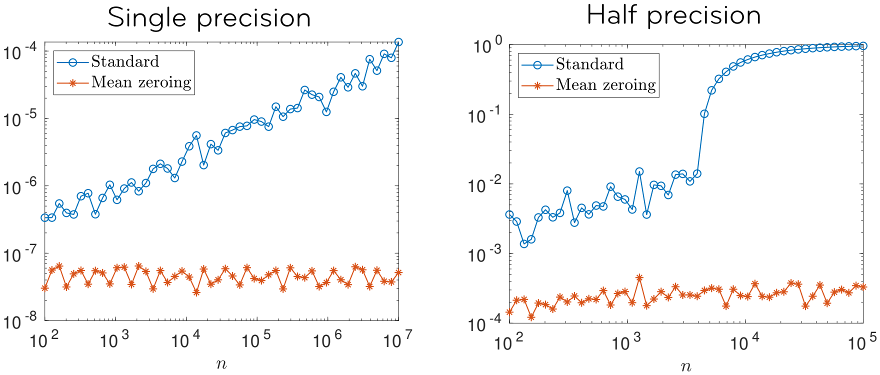
Cost: $2n$ flops but for $C = AB$, where $A, B, C \in \mathbb{R}^{n \times n}$ the cost of the algorithm below is $O(n^2) \ll O(n^3)$

$$\widetilde{A} \leftarrow A - xe^T$$
$$C \leftarrow \widetilde{A}B + x(e^T B)$$

where x_i = mean of ith row of A and e is the vector full of ones

Mean zeroing: numerical results

Backward error (for $[0, 1]$ data)



Single precision

Half precision

Smaller bounds: summary

Summation algorithm	Backward error	Cost
Compensated	$\propto u$	$\times 4$
Higher precision	$\propto u$	typically $\times 2$
Blocked*	$\propto (b+n/b)u$	
FABsum*	$\propto bu$	$\times(1+1/b)$
Mean zeroing**	$\propto u$	$\times(1+1/n)$
Tensor Cores	$\propto u$	$\div 4$

* worst case (probabilistic analogues: $\sqrt{b}u$ and $\sqrt{b+n/b}u$)

** under probabilistic model of the data

- Compensated: not suited for low precisions compared to use of higher precision
- Blocked: widely used in practice, dependence on n remains
- FABsum, mean zeroing: drop dependence on n for modest overhead
- Tensor Cores: nice, but hardware specific

Conclusion

With the emergence of low precision arithmetics,
classical analyses can no longer guarantee
the backward stability of classical algorithms

We need new analyses to obtain sharper bounds
⇒ probabilistic tools are both useful and timely

We need new algorithms to obtain smaller bounds
⇒ both high performance and high accuracy is possible!

Slides and papers available on my webpage

bit.ly/theomary