RNC 7, 2007, Nancy, France

Accurate dot products with FMA

S. Graillat, Ph. Langlois and N. Louvet

{graillat, langlois, nicolas.louvet}@univ-perp.fr

DALI-LP2A Laboratory, University of Perpignan, France

Motivation: what is the best way to use FMA for accurate dot products?

- IEEE-754 floating point arithmetic + FMA:
  - $\mathbf{F}$ denotes the set of the floating point numbers,
  - $\mathbf{u}$ is the working precision: *e.g.*, $\mathbf{u} = 2^{-53} \approx 10^{-16}$ in IEEE-754 double precision.
- $x = (x_1, \dots, x_n)^T$ and $y = (y_1, \dots, y_n)^T$ belong to $\mathbf{F}^{n \times 1}$.
- Floating point Fused Multiply and Add (FMA):
  - given a, b and c in $\mathbf{F}$, FMA(a, b, c) equals $a \times b + c$ rounded to the nearest floating point value.
  - only one rounding error for two arithmetic operations!
- Available on Intel IA-64, IBM RS/6000 and PowerPC.

Accuracy of Classic Dot Product

- We consider dot products without/with FMA:

Classic dot product	**Dot product with FMA**
function $\widehat{s} =$ **Dot**(x, y)	function $\widehat{s} =$ **DotFMA**(x, y)
$\widehat{s} = x_1 \otimes y_1$	$\widehat{s} = x_1 \otimes y_1$
for $i = 2 : n$	for $i = 2 : n$
$\widehat{s} = \widehat{s} \oplus x_i \otimes y_i$	$\widehat{s} =$ FMA$(x_i, y_i, \widehat{s})$

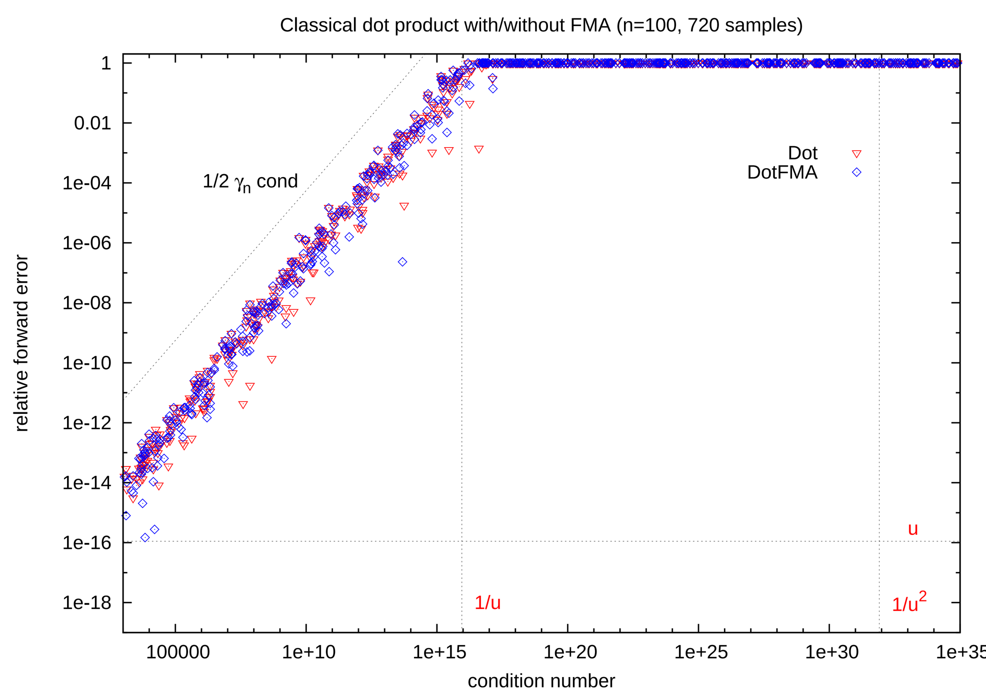
- The condition number for dot product computation is

$$\mathrm{cond}(x^Ty) = 2\frac{|x|^T|y|}{|x^Ty|}, \quad \text{with} \quad x^Ty \neq 0.$$

- **Worst case accuracy:** FMA does not improve the accuracy of computed dot product since **Dot** and **DotFMA** both verifies

$$\frac{|\widehat{s} - x^Ty|}{|x^Ty|} \le \frac{1}{2}\underbrace{\gamma_n}_{\approx n\,\mathbf{u}} \mathrm{cond}(x^Ty).$$

- **Practical accuracy:** FMA only slightly improves the actual accuracy.



- Classic dot product is not accurate enough when applied to ill-conditioned dot products *e.g.*, when computing residuals for ill-conditioned linear systems.
- **Question:** How can we obtain more accurate dot products?

Compensated Dot Products

- More accuracy can be achieved thanks to
  - double-double computations (see algorithm **DotXBLAS** below),
  - or with **compensated algorithms**: the forward error in the floating point evaluation of x^Ty is

$$c = x^Ty - \mathrm{computed}(x^Ty).$$

  The main idea is to compute an approximate $\widehat{c}$ of the global error c thanks to **Error Free Transformations** (EFT). Then a **compensated result** $\overline{r}$ is provided correcting the computed x^Ty as follows,

$$\overline{r} = \mathrm{computed}(x^Ty) \oplus \widehat{c}.$$

- From **Dot** and **DotFMA**, we derive two compensated algorithms using EFT (**2Sum**, **2ProdFMA** and **3FMA** are presented in the EFT frame below),
  - **CompDot**: correcting $+$ and $\times$ in **Dot** with **2Sum** and **2ProdFMA** (see [2]).
  - **CompDotFMA**: correcting FMA in **DotFMA** with **3FMA**.

Compensated Dot	**Compensated DotFMA**
function $\overline{r} =$ **CompDot**(x, y)	function $\overline{r} =$ **CompDotFMA**(x, y)
$[\widehat{s}, \widehat{c}] =$ **2ProdFMA**(x_1, y_1)	$[\widehat{s}, \widehat{c}] =$ **2ProdFMA**(x_1, y_1)
for $i = 2 : n$	for $i = 2 : n$
$[\widehat{p}, \pi] =$ **2ProdFMA**(x_i, y_i)	$[\widehat{s}, \alpha, \beta] =$ **3FMA**$(x_i, y_i, \widehat{s})$
$[\widehat{s}, \sigma] =$ **2Sum**$(\widehat{s}, \widehat{p})$	$\widehat{c} = \widehat{c} \oplus (\alpha \oplus \beta)$
$\widehat{c} = \widehat{c} \oplus (\pi \oplus \sigma)$	end
end	$\overline{r} = \widehat{s} \oplus \widehat{c}$
$\overline{r} = \widehat{s} \oplus \widehat{c}$	

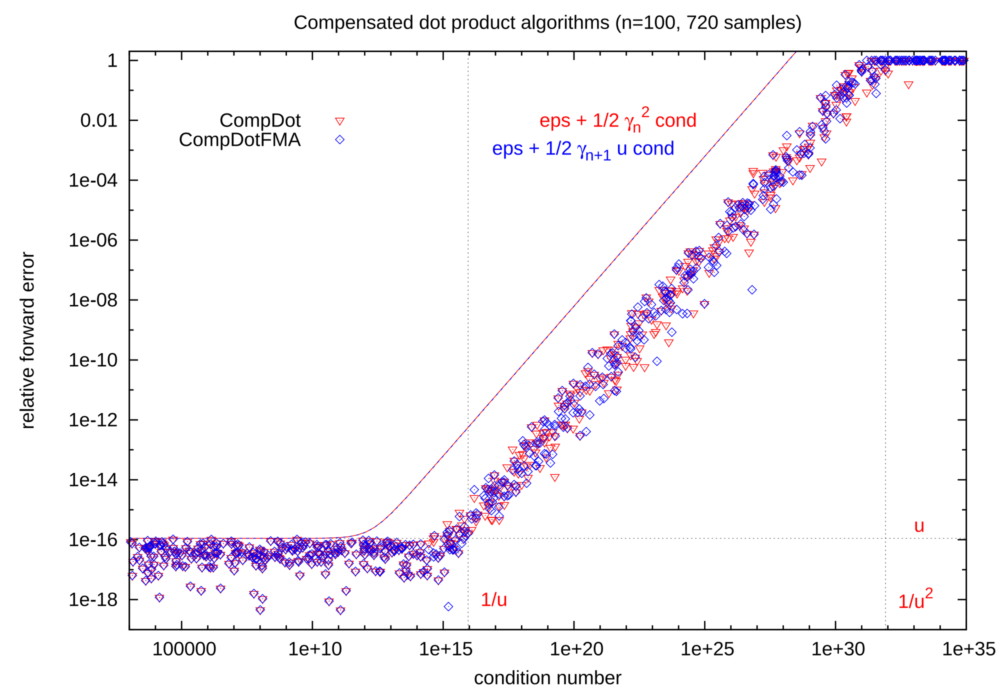
- The relative accuracy of the compensated result now verifies:

$$\frac{|\overline{r} - x^Ty|}{|x^Ty|} \le \begin{cases} \mathbf{u} + \frac{1}{2}\underbrace{\gamma_n^2}_{\approx n^2\,\mathbf{u}^2} \mathrm{cond}(x^Ty), & \text{with } \textbf{CompDot}, \\ \mathbf{u} + \frac{1}{2}\underbrace{\gamma_{n+1}\,\mathbf{u}}_{\approx (n+1)\,\mathbf{u}^2} \mathrm{cond}(x^Ty), & \text{with } \textbf{CompDotFMA}. \end{cases}$$

- **CompDot** and **CompDotFMA** are as accurate as classic dot product computed in doubled working precision $\mathbf{u}^2$.

Experimental results

- Algorithms **CompDot** and **CompDotFMA** as accurate as the classic dot product performed in doubled working precision means:
  - while $\mathrm{cond}(x^Ty) \lesssim \mathbf{u}^{-1}$, the accuracy is about the working precision $\mathbf{u}$,
  - when $\mathbf{u}^{-1} \lesssim \mathrm{cond}(x^Ty) \lesssim \mathbf{u}^{-2}$, the accuracy decreases from 16 digits to 0,
  - for $\mathrm{cond}(x^Ty) \gtrsim \mathbf{u}^{-2}$, no correct digit is returned.



- What is the running time over-cost compared to **DotFMA**?

n	**DotFMA**	**CompDot**	**CompDotFMA**	**DotXBLAS**
50	1.0	1.63	2.61	9.87
100	1.0	1.35	2.43	9.65
1000	1.0	1.26	2.6	10.86
10000	1.0	1.25	2.62	10.97
100000	1.0	1.25	2.35	9.8

Measured computing times on Intel Itanium 2 (1.6 GHz, ICC v9.0, IEEE-754 double precision)

- **Observations:**
  1. **CompDot** and **CompDotFMA** run both faster than **DotXBLAS**,
  2. **CompDot** is the most efficient alternative to **DotXBLAS**.

Error Free Transformations (EFT)

- Error Free Transformations are properties and algorithms to compute the generated rounding errors at the working precision $\mathbf{u}$. The following table sums up the EFT for $+$, $\times$ and FMA.

$+$	$(x, y) =$ **2Sum**(a, b) such that $a + b = x + y$ and $x = a \oplus b$	6 flops	Knuth (74)
$\times$	$(x, y) =$ **2ProdFMA**(a, b) such that $a \times b = x + y$ and $x = a \otimes b$ Indeed $y = a \times b - x =$ FMA$(a, b, -x)$	2 flops	
FMA	$(x, y, z) =$ **3FMA**(a, b, c) such that $x =$ FMA(a, b, c) and $a \times b + c = x + y + z$	17 flops	Boldo Muller (05)

- Remark: x, y and z belongs to $\mathbf{F}$ when a, b and c are in $\mathbf{F}$.
- Algorithm **2ProdFMA** is used instead of the classic **2Prod** (by Dekker and Veltkamp, 17 flops) to benefit from FMA.

XBLAS Dot Product

- **XBLAS** = BLAS + Bailey's double-doubles = eXtended and mixed precision BLAS [1].
- A double-double number = unevaluated sum of two IEEE-754 double precision numbers = at least 106 significand bits.
- **DotXBLAS** = Classic dot product (**Dot**) + double-doubles.
- **DotXBLAS** also benefits from the availability of FMA.

Conclusions

- FMA only slightly improves the accuracy of the classic dot product.
- Nevertheless FMA is useful for designing accurate algorithms: **CompDot** and **CompDotFMA** are very efficient for doubling the working precision.
- In particular **CompDot** is about 6 times faster than XBLAS algorithm **DotXBLAS** in our experiments.

[1] X. S. Li, J. W. Demmel, D. H. Bailey, G. Henry, Y. Hida, J. Iskandar, W. Kahan, S. Y. Kang, A. Kapur, M. C. Martin, B. J. Thompson, T. Tung and D. J. Yoo. Design, implementation and testing of extended and mixed precision BLAS. *ACM Transactions on Mathematical Software*, 28(2), 2002.

[2] T. Ogita, S. M. Rump, and S. Oishi. Accurate sum and dot product. *SIAM Journal on Scientific Computing*, 26(6), 2005.