Matrix multiplication over word-size modular rings using Bini's approximate formula

Brice Boyer Jean-Guillaume Dumas

JNCF 2014

3 Novembre 2014

Motivations/Goals

– matrix multiplication over word size $\mathbb{Z}/p\mathbb{Z}$ is a critical building block in exact linear algebra (matrix multiplication over $\mathbb{Z}$, over $\mathrm{GF}(q)$, Chinese remaindering,...)

Motivations/Goals

- matrix multiplication over word size $\mathbb{Z}/p\mathbb{Z}$ is a critical building block in exact linear algebra (matrix multiplication over $\mathbb{Z}$, over $\mathrm{GF}(q)$, Chinese remaindering,...)
- perform faster matrix multiplication over $\mathbb{Z}/p\mathbb{Z}$ (using fewer products)

Outline

Outline

Outline

Outline

Bini's approximate formula

Facts

– Computes $C_\varepsilon = A \times B + \varepsilon D(\varepsilon)$.

Bini's approximate formula

Facts

- Computes $C_\varepsilon = A \times B + \varepsilon D(\varepsilon)$.
- Multiplication of 3×2 and 2×2 matrices (noted $(3, 2, 2)$) using 10 products.

Bini's approximate formula

Facts

- Computes $C_\varepsilon = A \times B + \varepsilon D(\varepsilon)$.
- Multiplication of 3×2 and 2×2 matrices (noted $(3, 2, 2)$) using 10 products.
- Also (by duality): $(2, 3, 2)$ and $(2, 2, 3)$ multiplications.

Bini's approximate formula

Facts

- Computes $C_\varepsilon = A \times B + \varepsilon D(\varepsilon)$.
- Multiplication of 3×2 and 2×2 matrices (noted $(3, 2, 2)$) using 10 products.
- Also (by duality): $(2, 3, 2)$ and $(2, 2, 3)$ multiplications.
- Complexity of n^ω with $\omega \approx 2.780$ for $(12, 12, 12)$
  (compared to Strassen's $\omega \approx 2.807$)

Bini's approximate formula

Facts

- Computes $C_\varepsilon = A \times B + \varepsilon D(\varepsilon)$.
- Multiplication of 3×2 and 2×2 matrices (noted $(3, 2, 2)$) using 10 products.
- Also (by duality): $(2, 3, 2)$ and $(2, 2, 3)$ multiplications.
- Complexity of n^ω with $\omega \approx 2.780$ for $(12, 12, 12)$
  (compared to Strassen's $\omega \approx 2.807$)
- One call to Bini's algorithm saves roughly 4.5% operations (*vs.* Strassen's) on $3\,000 \times 3\,000$ matrices.

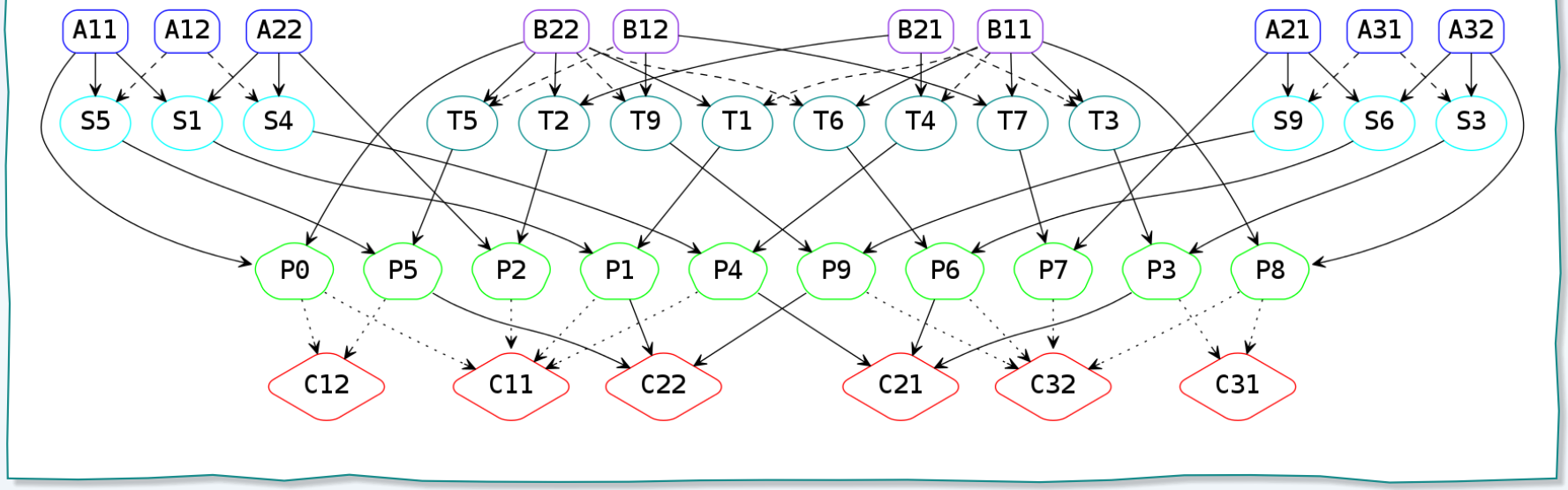
Bini's approximate formula

Algorithm

$$
\begin{array}{lll}
S_1 \leftarrow A_{11} + A_{22} & & T_1 \leftarrow B_{22} + \varepsilon \cdot B_{11} \\
 & & T_2 \leftarrow B_{21} + B_{22} \\
S_3 \leftarrow A_{32} + \varepsilon \cdot A_{31} & & T_3 \leftarrow B_{11} + \varepsilon \cdot B_{21} \\
S_4 \leftarrow A_{22} + \varepsilon \cdot A_{12} & & T_4 \leftarrow B_{21} - \varepsilon \cdot B_{11} \\
S_5 \leftarrow A_{11} + \varepsilon \cdot A_{12} & & T_5 \leftarrow B_{22} + \varepsilon \cdot B_{12} \\
S_6 \leftarrow A_{21} + A_{32} & & T_6 \leftarrow B_{11} + \varepsilon \cdot B_{22} \\
 & & T_7 \leftarrow B_{11} + B_{12} \\
S_9 \leftarrow A_{21} + \varepsilon \cdot A_{31} & & T_9 \leftarrow B_{12} - \varepsilon \cdot B_{22} \\
 & \rightsquigarrow & \\
P_0 \leftarrow A_{11} \times B_{22} & & P_1 \leftarrow S_1 \times T_1 \\
P_2 \leftarrow A_{22} \times T_2 & & P_3 \leftarrow S_3 \times T_3 \\
P_4 \leftarrow S_4 \times T_4 & & P_5 \leftarrow S_5 \times T_5 \\
P_6 \leftarrow S_6 \times T_6 & & P_7 \leftarrow A_{21} \times T_7 \\
P_8 \leftarrow A_{32} \times B_{11} & & P_9 \leftarrow S_9 \times T_9 \\
 & \rightsquigarrow & \\
C_{11} \leftarrow (P_1 - P_2 + P_4 - P_0)/\varepsilon & & C_{12} \leftarrow (P_5 - P_0)/\varepsilon \\
C_{21} \leftarrow P_4 - P_3 + P_6 & & C_{22} \leftarrow P_1 - P_5 + P_9 \\
C_{31} \leftarrow (P_3 - P_8)/\varepsilon & & C_{32} \leftarrow (P_6 - P_7 + P_9 - P_8)/\varepsilon
\end{array}
$$

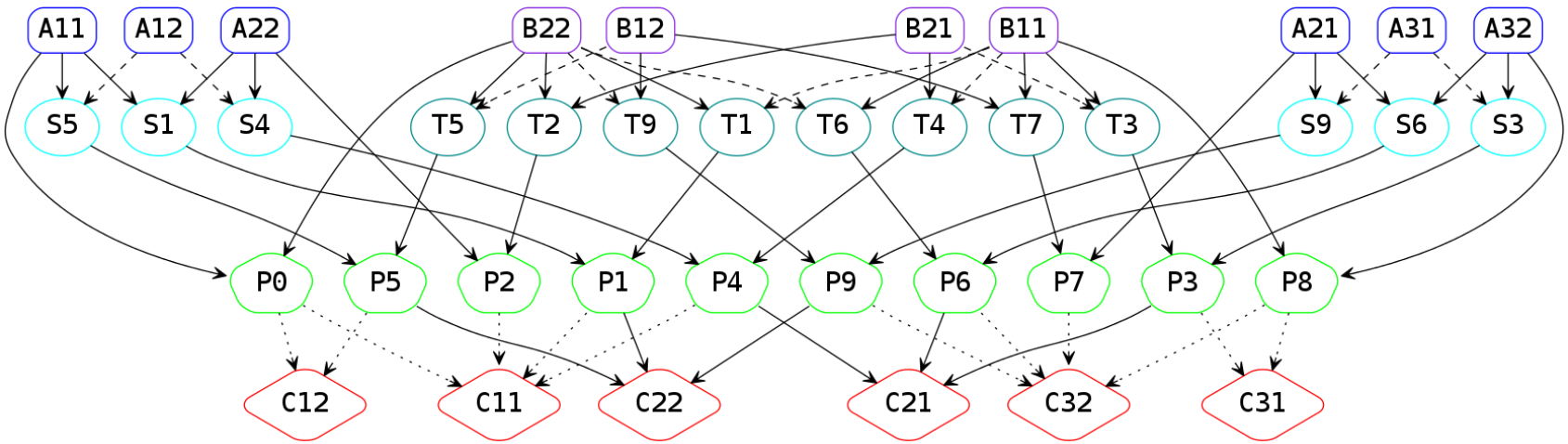
Bini's approximate formula

Dependencies



Bini's approximate formula

Dependencies



Symmetries !

Bini's approximate formula

Scheduling of the Algorithm

#	operation	var	#	operation	var
1	C11 := A11 * B22	P0	19	Y := B12 - e . B22	T9
2	X := A11 + e . A12	S5	20	C32 := X * Y	P9
3	Y := e . B12 + B22	T5	21	C22 := C22 + C32	C22
4	C22 := X * Y	P5	22	X := A21 + A32	S6
5	C12 := (C22 - C11)/e	C12	23	Y := B11 + e . B22	T6
6	Y := B21 + B22	T2	24	C31 := X * Y	P6
7	C31 := A22 * Y	P2	25	C21 := C21 + C31	C21
8	C11 := C11 + C31	C11	26	C32 := C32 + C31	C32
9	X := A11 + A22	S1	27	Y := B11 + B12	T7
10	Y := e . B11 + B22	T1	28	C31 := A21 * Y	P7
11	C21 := X * Y	P1	29	C32 := C32 - C31	C32
12	C22 := C21 - C22	C22	30	X := e . A31 + A32	S3
13	C11 := C21 - C11	C11	31	Y := B11 + e . B21	T3
14	X := e . A12 + A22	S4	32	C31 := X * Y	P3
15	Y := B21 - e . B11	T4	33	C21 := C21 - C31	C21
16	C21 := X * Y	P4	34	Y := A32 * B11	P8
17	C11 := (C21 + C11)/e	C11	35	C31 := (C31 - Y)/e	C31
18	X := A21 + e . A31	S9	36	C32 := (C32 - Y)/e	C32

Bini's approximate formula

Scheduling of the Algorithm

#	operation	var	#	operation	var
1	C11 := A11 * B22	P0	19	Y := B12 - e . B22	T9
2	X := A11 + e . A12	S5	20	C32 := X * Y	P9
3	Y := e . B12 + B22	T5	21	C22 := C22 + C32	C22
4	C22 := X * Y	P5	22	X := A21 + A32	S6
5	C12 := (C22 - C11)/e	C12	23	Y := B11 + e . B22	T6
6	Y := B21 + B22	T2	24	C31 := X * Y	P6
7	C31 := A22 * Y	P2	25	C21 := C21 + C31	C21
8	C11 := C11 + C31	C11	26	C32 := C32 + C31	C32
9	X := A11 + A22	S1	27	Y := B11 + B12	T7
10	Y := e . B11 + B22	T1	28	C31 := A21 * Y	P7
11	C21 := X * Y	P1	29	C32 := C32 - C31	C32
12	C22 := C21 - C22	C22	30	X := e . A31 + A32	S3
13	C11 := C21 - C11	C11	31	Y := B11 + e . B21	T3
14	X := e . A12 + A22	S4	32	C31 := X * Y	P3
15	Y := B21 - e . B11	T4	33	C21 := C21 - C31	C21
16	C21 := X * Y	P4	34	Y := A32 * B11	P8
17	C11 := (C21 + C11)/e	C11	35	C31 := (C31 - Y)/e	C31
18	X := A21 + e . A31	S9	36	C32 := (C32 - Y)/e	C32

- Only 2 temporaries! (X and Y)
- Easy to make it inplace while overwriting the left or right operand

Bini's approximate formula

Scheduling of the Algorithm

#	operation	var	#	operation	var
1	C11 := A11 * B22	P0	19	Y := B12 - e . B22	T9
2	X := A11 + e . A12	S5	20	C32 := X * Y	P9
3	Y := e . B12 + B22	T5	21	C22 := C22 + C32	C22
4	C22 := X * Y	P5	22	X := A21 + A32	S6
5	C12 := (C22 - C11)/e	C12	23	Y := B11 + e . B22	T6
6					P6
7					C21
8					C32
9					T7
10					P7
11					C32
12					S3
13					T3
14					P3
15					C21
16					P8
17					C31
18					C32

Brice Boyer, Jean-Guillaume Dumas, Clément Pernet, and Wei Zhou.
Memory efficient scheduling of Strassen-Winograd's matrix multiplication algorithm.
In Proceedings of the 2009 Internat. Symp. Symbolic Algebraic Comput., ISSAC '09, pages 55–62, New York, NY, USA, 2009. ACM.

- Only 2 temporaries! (`X` and `Y`)
- Easy to make it inplace while overwriting the left or right operand

Outline

Getting an exact algorithm

- Let $d = \deg_\varepsilon(\varepsilon D(\varepsilon))$.
- Find a $d+1$ scalars α_i, and $d+1$ pair-wise distinct scalars ε_i.
- Make sure $\sum_{i=1}^{d+1} \alpha_i = 1$ and for $j = 1, \ldots, d$ that $\sum_{i=1}^{d+1} \alpha_i \varepsilon_i^j = 0$.
- Then $\sum_{i=1}^{d+1} \alpha_i C_{\varepsilon_i} = A \times B$

Getting an exact algorithm

- Let $d = \deg_\varepsilon(\varepsilon D(\varepsilon))$.
- Find a $d+1$ scalars α_i, and $d+1$ pair-wise distinct scalars ε_i.
- M
- Th

D. Bini.
Relations between exact and approximate bilinear algorithms. applications.
Calcolo, 17:87–97, 1980.
10.1007/BF02575865.

Using Only One Call

Ideas

– Only “one recursive” call

Using Only One Call

Ideas

- Only “one recursive” call
- for a `double`: $\varepsilon = 2^{-27} \rightsquigarrow \varepsilon^2 \approx 0$.

Using Only One Call

Ideas

- Only “one recursive” call
- for a `double`: $\varepsilon = 2^{-27} \rightsquigarrow \varepsilon^2 \approx 0$.
- Modulo p: $\varepsilon = p = 0 \rightsquigarrow \varepsilon^2 = 0$.

Exact Algorithm 1

Case $\varepsilon = 2^{-27}$

Proposition ($\varepsilon = 2^{-27}$)

For an (m, k, n) matrix multiplication over $\mathbb{Z}/p\mathbb{Z}$, the rounding to the nearest integer of the output C_ε of one call to Bini's $(3, 2, 2)$—approximate algorithm using `double` floating point arithmetic, gives the exact result C, provided that:

$$2\lfloor k/2 \rfloor (p-1)^2 < \frac{1}{3} 2^{27}.$$

Exact Algorithm 1

Case $\varepsilon = 2^{-27}$

Proposition ($\varepsilon = 2^{-27}$)

For an (m, k, n) matrix multiplication over $\mathbb{Z}/p\mathbb{Z}$, the rounding to the nearest integer of the output C_ε of one call to Bini's $(3, 2, 2)$—approximate algorithm using `double` floating point arithmetic, gives the exact result C, provided that:

$$2\lfloor k/2 \rfloor (p-1)^2 < \frac{1}{3} 2^{27}.$$

Using balanced representation:

$$\lfloor k/2 \rfloor (p-1)^2 < \frac{1}{3} 2^{27}.$$

Exact Algorithm 2

Case $\varepsilon = p$

Proposition ($\varepsilon = p$)

For an (m, k, n) matrix multiplication over $\mathbb{Z}/p\mathbb{Z}$, the **reduction modulo** p of the output C_ε of one call to Bini's $(3, 2, 2)$—approximate algorithm with `double` floating point arithmetic, gives the exact result C, provided that:

$$\lfloor k/2 \rfloor (p-1)^2 (p+1)^2 < 2^{53}.$$

Exact Algorithm 2

Case $\varepsilon = p$

Proposition ($\varepsilon = p$)

For an (m, k, n) matrix multiplication over $\mathbb{Z}/p\mathbb{Z}$, the **reduction modulo** p of the output C_ε of one call to Bini's $(3, 2, 2)$—approximate algorithm with `double` floating point arithmetic, gives the exact result C, provided that:

$$\lfloor k/2 \rfloor (p-1)^2 (p+1)^2 < 2^{53}.$$

Using balanced representation:

$$1/2 \lfloor k/2 \rfloor (p-1)^2 p(p+1) < 2^{53}.$$

Example

Achievable size of p

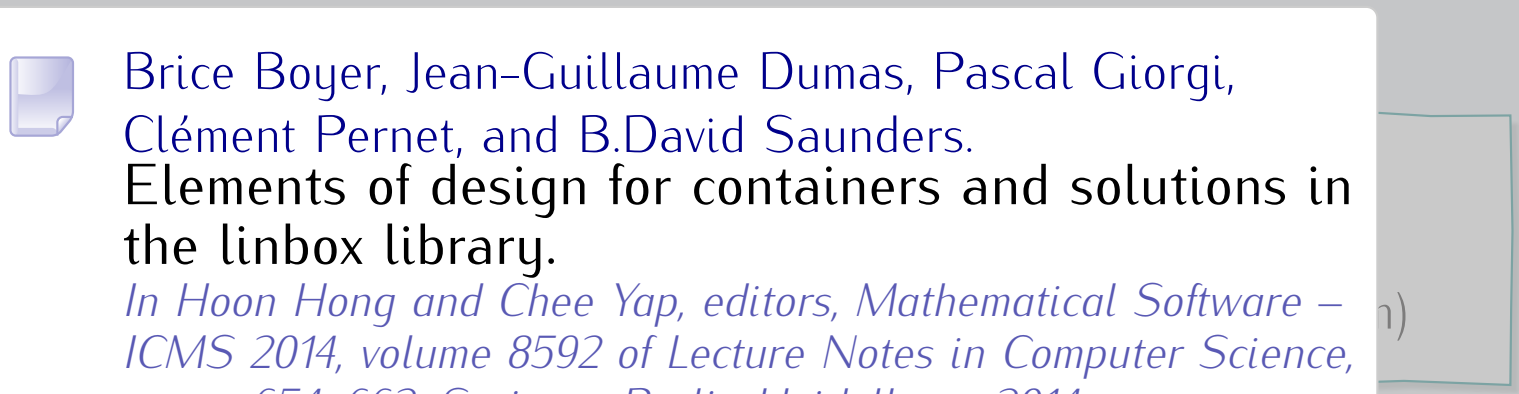
	$\varepsilon = 2^{-27}$		$\varepsilon = p$	
size (k)	Positive	Balanced	Positive	Balanced
1 000	212	599	2 060	2 450
2 000	150	424	1 732	2 060
3 000	123	346	1 565	1 861
4 000	106	300	1 456	1 732

Outline

Framework

- Use the FFLAS–FFPACK library and compare to existing implementations of Strassen–Winograd in `fgemm`.
- Use cascading designs (not new, but we automatise/generalise them)

Framework

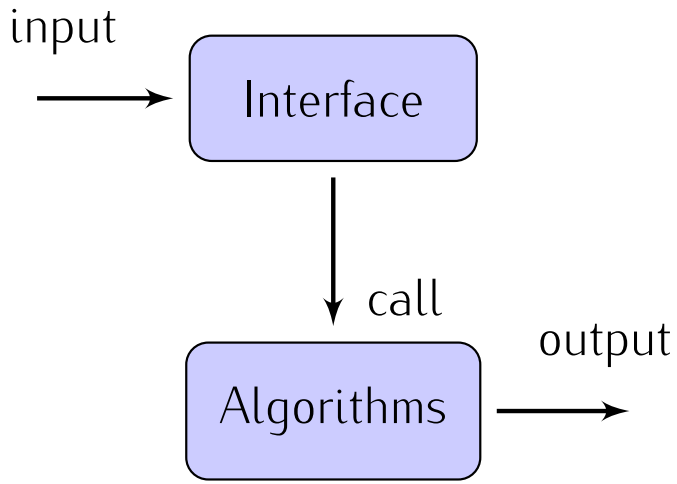


Brice Boyer, Jean-Guillaume Dumas, Pascal Giorgi, Clément Pernet, and B.David Saunders.
Elements of design for containers and solutions in the linbox library.
In Hoon Hong and Chee Yap, editors, Mathematical Software – ICMS 2014, volume 8592 of Lecture Notes in Computer Science, pages 654–662. Springer Berlin Heidelberg, 2014.

Jean-Guillaume Dumas, Pascal Giorgi, and Clément Pernet.
Dense linear algebra over word-size prime fields: the FFLASand FFPACK packages.
ACM Trans. Math. Softw., 35(3):1–42, 2008.

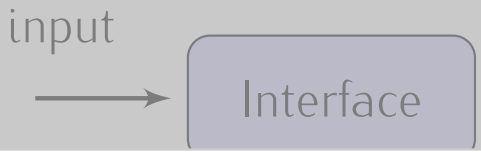
Design

Strategy Design Pattern



Design

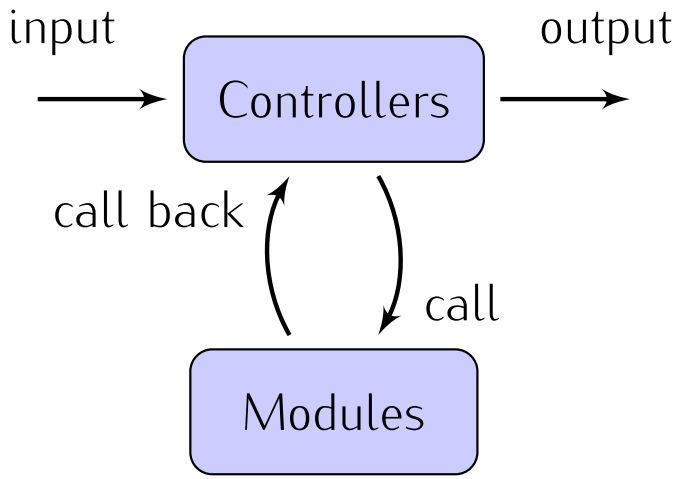
Strategy Design Pattern



E. Gamma.
Design Patterns: Elements of Reusable Object-Oriented Software.
Addison-Wesley Professional Computing Series.
Addison-Wesley, 1995.

Design

Controller/Module Design Pattern



Algorithms

Matrix Multiplication Algorithms

- Naïve algorithms (using BLAS)
- Strassen–Winograd (Recursive)
- Bini's algorithm (Non Recursive)
- In place fast algorithms

Algorithms

Matrix Multiplication Algorithms

- Naïve algorithms (using BLAS)
- Strassen–Winograd (Recursive)
- Bini's algorithm (Non Recursive)
- In place fast algorithms

- Thresholds (cut-off) when one algorithm performs better;
- Typically: a threshold for BLAS/Strassen–Winograd for $k \approx 1\,000$;
- Typically: BLAS on `float` are $\approx 2\times$ faster than on `double`.
- Any threshold/method has default strategy choices and they can be automatically tuned.

Algorithms

Matrix Multiplication Algorithms

- Naïve algorithms (using BLAS)
- Strassen–Winograd (Recursive)
- Bini's algorithm (Non Recursive)
- In place fast algorithms

- Thresholds (cut-off) when one algorithm performs better;
- Typically: a threshold for BLAS/Strassen–Winograd for $k \approx 1\,000$;
- Typically: BLAS on `float` are $\approx 2\times$ faster than on `double`.
- Any threshold/method has default strategy choices and they can be automatically tuned.

Where do we use Bini?

Automatic! Can be:

- Winograd+Bini+Winograd+BLAS
- Using our new `Helper` structures

Timings

dimensions (×1 000)	p	$\varepsilon = 2^{-27}$		$\varepsilon = p$		FFLAS `double`		FFLAS `float`		rel. change (%)	
		P	B	P	B	P	B	P	B	P	B
(1.5)	141	0.31	0.31	0.31	0.31	0.32	0.32	0.25	0.25	+21.4	+22.2
(1.5)	451	n/a	0.31	0.31	0.31	0.32	0.32	0.32	0.32	−2.99	−3.28
(1.5)	1 001	n/a	n/a	0.31	0.31	0.32	0.32	0.41	0.43	−2.95	−2.92
(1.5)	1 501	n/a	n/a	0.31	0.31	0.32	0.32	1.50	1.52	−2.76	−2.98
(2.7)	1 001	n/a	n/a	1.70	1.72	2.03	2.02	2.42	2.43	−16.3	−15.0
(2.7)	1 501	n/a	n/a	1.71	1.71	2.03	2.03	9.25	9.25	−15.9	−16.0
(3.3)	1 001	n/a	n/a	3.08	3.08	3.47	3.47	4.48	4.47	−11.1	−11.0
(3.3)	1 501	n/a	n/a	3.09	3.09	3.46	3.45	17.0	16.8	−10.7	−10.6
(3.9)	1 001	n/a	n/a	4.94	4.97	5.39	5.54	6.89	6.96	−8.34	−10.5
(3.9)	1 501	n/a	n/a	n/a	4.94	5.39	5.51	27.8	28.1	n/a	−10.4
(3.0, 2.7, 2.7)	1 001	n/a	n/a	1.88	1.89	2.00	2.00	2.75	2.71	−6.02	−5.62
(2.7, 3.0, 2.7)	1 001	n/a	n/a	1.83	1.85	2.16	2.16	2.69	2.73	−15.4	−14.7
(2.7, 2.7, 3.0)	1 001	n/a	n/a	1.86†	1.87†	2.26	2.25	2.73	2.76	−17.4	−16.9
(3.6, 2.7, 2.7)	1 001	n/a	n/a	2.26	2.28	2.39	2.38	3.18	3.15	−5.34	−4.43
(2.7, 3.6, 2.7)	1 001	n/a	n/a	2.20	2.20	2.55	2.55	3.16	3.18	−14.0	−13.6
(2.7, 2.7, 3.6)	1 001	n/a	n/a	2.23†	2.22†	2.65	2.65	3.15	3.14	−16.0	−16.2
(4.2, 2.7, 2.7)	1 001	n/a	n/a	2.62	2.63	2.76	2.76	3.68	3.64	−5.11	−4.65
(2.7, 4.2, 2.7)	1 001	n/a	n/a	2.54	2.59	2.98	2.98	3.79	3.68	−14.4	−13.6
(2.7, 2.7, 4.2)	1 001	n/a	n/a	2.58†	2.59†	3.08	3.09	3.71	3.70	−16.1	−16.1

Thank you
for your attention !

Matrix multiplication over word-size modular rings using Bini's approximate formula

Brice Boyer Jean-Guillaume Dumas

JNCF 2014

3 Novembre 2014