

Arithmétique modulaire astucieuse

Réduction modulaire simultanée et substitution de Kronecker pour les petits corps finis

Jean-Guillaume Dumas¹ **Laurent Fousse**¹ Bruno Salvy²

¹Laboratoire J. Kuntzmann
Université Grenoble 1

²INRIA Rocquencourt

28 octobre 2009 (RAIM'09)



MATHÉMATIQUES APPLIÉES - INFORMATIQUE

Plan

- 1 Problématique
- 2 Deux idées magiques
 - Substitution de Kronecker
 - REDQ
- 3 Division euclidienne en flottants
- 4 Applications
 - Multiplication polynomiale
 - Multiplication matricielle compressée
- 5 Conclusion



Plan

- 1 Problématique
- 2 Deux idées magiques
 - Substitution de Kronecker
 - REDQ
- 3 Division euclidienne en flottants
- 4 Applications
 - Multiplication polynomiale
 - Multiplication matricielle compressée
- 5 Conclusion



Besoins en arithmétique exacte

Justification métaphysique

- Arithmétique efficace pour les petits corps finis ;
- En particulier pour l'algèbre linéaire.



Besoins en arithmétique exacte

Justification métaphysique

- Arithmétique efficace pour les petits corps finis ;
- En particulier pour l'algèbre linéaire.

Théorie des graphes

Pseudo-Paley graphs and skew Hadamard difference sets from presemifields
(Des. Codes Cryptogr. (2007) 44 :49 62)
G. Weng, W. Qiu, Z. Wang, Q. Xiang.



Besoins en arithmétique exacte

Justification métaphysique

- Arithmétique efficace pour les petits corps finis ;
- En particulier pour l'algèbre linéaire.

Théorie des graphes

Pseudo-Paley graphs and skew Hadamard difference sets from presemifields
(Des. Codes Cryptogr. (2007) 44 :49 62)
G. Weng, W. Qiu, Z. Wang, Q. Xiang.

Calculs de rang

Efficient Matrix Rank Computation with Application to the Study of Strongly Regular Graphs
John P. May, David Saunders, Zhendong Wan.
ISSAC 2007.

Problème

Un CPU généraliste, donc qui ne sait rien faire

Votre CPU ne sait pas calculer nativement :

- dans $\mathbb{Z}/pq\mathbb{Z}$;



Problème

Un CPU généraliste, donc qui ne sait rien faire

Votre CPU ne sait pas calculer nativement :

- dans $\mathbb{Z}/pq\mathbb{Z}$;
- dans $\mathbb{Z}/17\mathbb{Z}$;



Problème

Un CPU généraliste, donc qui ne sait rien faire

Votre CPU ne sait pas calculer nativement :

- dans $\mathbb{Z}/pq\mathbb{Z}$;
- dans $\mathbb{Z}/17\mathbb{Z}$;
- dans \mathbb{F}_{35} .



Problème

Un CPU généraliste, donc qui ne sait rien faire

Votre CPU ne sait pas calculer nativement :

- dans $\mathbb{Z}/pq\mathbb{Z}$;
- dans $\mathbb{Z}/17\mathbb{Z}$;
- dans \mathbb{F}_{35} .

Deux philosophies :

- On le jette et on se fait un circuit *ad hoc* en FPGA ;



Problème

Un CPU généraliste, donc qui ne sait rien faire

Votre CPU ne sait pas calculer nativement :

- dans $\mathbb{Z}/pq\mathbb{Z}$;
- dans $\mathbb{Z}/17\mathbb{Z}$;
- dans \mathbb{F}_{35} .

Deux philosophies :

- On le jette et on se fait un circuit *ad hoc* en FPGA ;
- On lui apprend.



Problématique (suite)

Leviers

Choix de représentation :

- des éléments du corps ;
- de la matrice/des polynômes.



Problématique (suite)

Leviers

Choix de représentation :

- des éléments du corps ;
- de la matrice/des polynômes.

Trucs à considérer

En vrac :

- Taille des matrices/polynômes manipulées ;
- Taille des entiers/flottants machine ;
- Gestion du cache ;
- Division 10 à 100 fois plus lente qu'une addition/multiplication ;
- Les réductions modulaires coûtent.



Idées/solutions existantes

- BLAS numériques performantes ;
- Routines optimisés pour le cas mod 2 (notamment dans NTL) ;
- Réduction flottante dans NTL ;
- Algorithmes *cache-aware* (FFLAS/FFPACK).



Plan

1 Problématique

2 Deux idées magiques

- Substitution de Kronecker
- REDQ

3 Division euclidienne en flottants

4 Applications

- Multiplication polynomiale
- Multiplication matricielle compressée

5 Conclusion



Plan

- 1 Problématique
- 2 Deux idées magiques
 - Substitution de Kronecker
 - REDQ
- 3 Division euclidienne en flottants
- 4 Applications
 - Multiplication polynomiale
 - Multiplication matricielle compressée
- 5 Conclusion



Le morphisme à tout faire

Substitution de Kronecker (DQT)

Pour $q \in \mathbb{Z}$, l'application

$$\begin{aligned}\varphi : \mathbb{Z}[X] &\rightarrow \mathbb{Z} \\ P(X) &\mapsto P(q)\end{aligned}$$

est un morphisme d'anneaux (et avec les bons paramètres, plus ou moins inversible).



Exemple

$$q = 100.$$

$$a(X) = X + 1$$

$$b(X) = X + 2$$

$$\varphi(a) = 100 + 1 = 101$$

$$\varphi(b) = 100 + 2 = 102$$

$$\varphi(a) \cdot \varphi(b) = 10302$$

$$a \cdot b = X^2 + 3X + 2$$

$$\varphi(a \cdot b) = 100^2 + 3 \cdot 100 + 2 = 10302.$$



Cas général

Plus généralement :

$$a(X) = \sum_{i=0}^{k-1} a_i X^i$$

$$b(X) = \sum_{i=0}^{k-1} b_i X^i$$

$$\varphi(a \cdot b) = \sum_{j=0}^{2k-2} \left(\sum_{i=0}^j a_i b_{j-i} \right) q^j.$$

q assez grand : le coefficient de q^j ne débordera pas dans le terme suivant.



DQT et produit scalaire polynomial (Dumas et al. 2002)

Require: $v_1 = [P_1(X), P_2(X), \dots, P_n(X)]$ with $P_i \in \mathbb{Z}/p\mathbb{Z}[X]$.

Require: $\deg(P_i) < k$.

Require: $\deg(R_i) < k$.

Require: $v_2 = [R_1(X), R_2(X), \dots, R_n(X)]$ with $R_i \in \mathbb{Z}/p\mathbb{Z}[X]$.

Ensure: $R \in \mathbb{Z}/p\mathbb{Z}[X]$ with $R = \langle v_1, v_2 \rangle$.

- 1: $\tilde{v}_1 \leftarrow [P_1(q), P_2(q), \dots, P_n(q)]$
- 2: $\tilde{v}_2 \leftarrow [R_1(q), R_2(q), \dots, R_n(q)]$
- 3: $\tilde{r} \leftarrow P_1(q)R_1(q) + \dots + P_n(q)R_n(q)$
- 4: $\tilde{r} = \sum_{i=0}^{2k-2} \tilde{\mu}_i q^i$.
- 5: $\mu_i \leftarrow \tilde{\mu}_i \pmod{p}$.
- 6: Return $R = \sum_{i=0}^{2k-2} \mu_i X^i$.



DQT et produit scalaire polynomial (Dumas et al. 2002)

Théorème

Si β est la taille en bit des mots machines et

$$q > nk(p - 1)^2 \text{ et } (2k - 1) \log_2(q) \leq \beta$$

alors l'algorithme précédent est correct.



Plan

- 1 Problématique
- 2 Deux idées magiques
 - Substitution de Kronecker
 - REDQ
- 3 Division euclidienne en flottants
- 4 Applications
 - Multiplication polynomiale
 - Multiplication matricielle compressée
- 5 Conclusion



Réduction modulaire simultanée (REDQ)

Exemple : Dans $\mathbb{Z}/5\mathbb{Z}$:

$$a = X^2 + 2X + 3$$

$$b = 4X^2 + 5X + 6$$

Alors

$$a \times b = 4X^4 + 3X^3 + 3X^2 + 2X + 3.$$

Prenons $q = 10000$.

$$\tilde{r} := \tilde{a} \times \tilde{b} = \mathbf{40013002800270018}$$

→ il reste encore à réduire !



Réduction modulaire simultanée

$$\tilde{r} := \tilde{a} \times \tilde{b} = \mathbf{40013002800270018}$$

Approche naïve :

$$4 = 0 \times 5 + 4, \quad 13 = 2 \times 5 + 3, \quad 28 = 5 \times 5 + 3, \quad 27 = 5 \times 5 + 2, \quad 18 = 3 \times 5 + 3.$$



Réduction modulaire simultanée

$$\tilde{r} := \tilde{a} \times \tilde{b} = \mathbf{40013002800270018}$$

Approche naïve :

$$4 = 0 \times 5 + 4, \quad 13 = 2 \times 5 + 3, \quad 28 = 5 \times 5 + 3, \quad 27 = 5 \times 5 + 2, \quad 18 = 3 \times 5 + 3.$$

Approche simultanée :

$$s := \lfloor \tilde{r}/p \rfloor = \mathbf{08002600560054003}$$

Puis

$$u_i := \lfloor \tilde{r}/q^i \rfloor - p \lfloor s/q^i \rfloor, \text{ pour } i = 0, \dots, 4.$$



Réduction modulaire simultanée

Prenons $R = 1234X^3 + 5678X^2 + 9123X + 4567$ à réduire modulo $p = 23$.
On choisit $q = 10^6$.

$$R \bmod p = 15X^3 + 20X^2 + 15X + 13$$

$$\tilde{r} = 1234005678009123004567$$

$$s := \lfloor \tilde{r}/p \rfloor = 53652420783005348024$$

On obtient comme avant :

$$u_0 = \lfloor \tilde{r}/q^0 \rfloor - 23 \lfloor s/q^0 \rfloor = 15,$$

$$u_1 = \lfloor \tilde{r}/q^1 \rfloor - 23 \lfloor s/q^1 \rfloor = 8,$$

...

Valeurs incorrectes ! Il faut une étape de correction : $\mu_3 = u_3$
et $\mu_i = u_i - qu_{i+1} \bmod p$ pour $i = 0, 1, 2$.



Algorithme REDQ

Require: Two integers p and q .

Require: $\tilde{r} = \sum_{i=0}^d \tilde{\mu}_i q^i \in \mathbb{Z}$.

Ensure: $\rho \in \mathbb{Z}$, with $\rho = \sum_{i=0}^d \mu_i q^i$ where $\mu_i = \tilde{\mu}_i \bmod p$.

REDQ COMPRESSION

- 1: $s = \left\lfloor \frac{\tilde{r}}{p} \right\rfloor$;
- 2: **for** $i = 0$ to d **do**
- 3: $u_i = \left\lfloor \frac{\tilde{r}}{q^i} \right\rfloor - p \left\lfloor \frac{s}{q^i} \right\rfloor$;
- 4: **end for**

REDQ CORRECTION {when $p \nmid q$ }

- 5: $\mu_d = u_d$
- 6: **for** $i = 0$ to $d - 1$ **do**
- 7: $\mu_i = u_i - q u_{i+1} \bmod p$;
- 8: **end for**
- 9: Return $\rho = \sum_{i=0}^d \mu_i q^i$.



Algorithme REDQ

Théorème

L'algorithme REDQ est correct.

Remarque : quand q est une puissance de 2 on gagne sur les divisions (et même sur le nombre d'itérations de la boucle l. 2-4).



Algorithme REDQ

Théorème

L'algorithme REDQ est correct.

Remarque : quand q est une puissance de 2 on gagne sur les divisions (et même sur le nombre d'itérations de la boucle l. 2-4).

Et peut-on gagner sur la division de la ligne 1 ?



Plan

- 1 Problématique
- 2 Deux idées magiques
 - Substitution de Kronecker
 - REDQ
- 3 Division euclidienne en flottants
- 4 Applications
 - Multiplication polynomiale
 - Multiplication matricielle compressée
- 5 Conclusion



Division euclidienne

$$r = kp + u$$

r, p entiers.

Comment calculer k efficacement ?



Division euclidienne

$$r = kp + u$$

r, p entiers.

Comment calculer k efficacement ?

- La division directe r/p coûte cher.



Division euclidienne

$$r = kp + u$$

r, p entiers.

Comment calculer k efficacement ?

- La division directe r/p coûte cher.
- Idée : calculer $1/p$ en flottant une seule fois et utiliser des multiplications.



Division euclidienne

$$r = kp + u$$

r, p entiers.

Comment calculer k efficacement ?

- La division directe r/p coûte cher.
- Idée : calculer $1/p$ en flottant une seule fois et utiliser des multiplications.

$$k = \lfloor r \cdot (1/p) \rfloor \quad ?$$



Division euclidienne

$$r = kp + u$$

r, p entiers.

Comment calculer k efficacement ?

- La division directe r/p coûte cher.
- Idée : calculer $1/p$ en flottant une seule fois et utiliser des multiplications.

$$k = \lfloor r \cdot (1/p) \rfloor \quad ?$$

$$k = \lfloor \circ(r \cdot \circ(1/p)) \rfloor \quad ?$$



Modèle flottant

- IEEE 754 ;
- $\text{ulp}(z)$ (*unit in the last place*) :

$$2^{\beta-1} \text{ulp}(z) \leq |z| \leq (2^{\beta} - 1) \text{ulp}(z) ;$$

- Trois modes d'arrondi : $\Delta(\cdot)$, $\nabla(\cdot)$ et $\diamond(\cdot)$;
- Un changement de mode d'arrondi peut être coûteux.



Algorithme FDIV

Require: One integer r such that $0 \leq r \leq 2^\beta - 1$;

Require: one integer p such that $1 \leq p \leq 2^\beta - 1$;

Require: two choices of rounding-modes \circ_1 and \circ_2 .

Ensure: $\lfloor \frac{r}{p} \rfloor$.

1: $invp \leftarrow \circ_1(1/p)$

2: $x \leftarrow \circ_2(r \cdot invp)$

3: Return $\lfloor x \rfloor$.



Résultats

| Cas | \circ_1 | \circ_2 | Résultat naïf | Borne sur r | Bits perdus |
|-----|--------------------|--------------------|---|------------------------------------|-------------|
| 1 | $\triangle(\cdot)$ | $\triangle(\cdot)$ | $k \leq \lfloor x \rfloor \leq k + 1$ | $2^\beta / (4 + 2^{2-\beta})$ | 3 |
| 2 | $\triangle(\cdot)$ | $\diamond(\cdot)$ | $k \leq \lfloor x \rfloor \leq k + 1$ | $2^\beta / (3 + 2^{1-\beta})$ | 2 |
| 3 | $\triangle(\cdot)$ | $\nabla(\cdot)$ | $k \leq \lfloor x \rfloor \leq k + 1$ | $2^\beta / 2$ | 1 |
| 4 | $\diamond(\cdot)$ | $\triangle(\cdot)$ | $k \leq \lfloor x \rfloor \leq k + 1$ | $2^\beta / (3 + 2^{1-\beta})$ | 2 |
| 5 | $\diamond(\cdot)$ | $\diamond(\cdot)$ | $k - 1 \leq \lfloor x \rfloor \leq k + 1$ | $2^{\beta-1} / (1 + 2^{-1-\beta})$ | 2 |
| 6 | $\diamond(\cdot)$ | $\nabla(\cdot)$ | $k - 1 \leq \lfloor x \rfloor \leq k$ | - | 0 |
| 7 | $\nabla(\cdot)$ | $\triangle(\cdot)$ | $k - 1 \leq \lfloor x \rfloor \leq k + 1$ | $2^\beta / 2$ | 1 |
| 8 | $\nabla(\cdot)$ | $\diamond(\cdot)$ | $k - 1 \leq \lfloor x \rfloor \leq k$ | - | 0 |
| 9 | $\nabla(\cdot)$ | $\nabla(\cdot)$ | $k - 1 \leq \lfloor x \rfloor \leq k$ | - | 0 |



Preuves

On appelle ϵ_1 and ϵ_2 les erreurs d'arrondi telles que :

$$\text{inv}p = \frac{1}{p}(1 + \epsilon_1), \quad x = (r \cdot \text{inv}p)(1 + \epsilon_2).$$



Preuves

On appelle ϵ_1 and ϵ_2 les erreurs d'arrondi telles que :

$$\text{invp} = \frac{1}{\rho}(1 + \epsilon_1), \quad x = (r \cdot \text{invp})(1 + \epsilon_2).$$

Donc la valeur calculée est

$$\begin{aligned} x &= \frac{r}{\rho}(1 + \epsilon_1)(1 + \epsilon_2) \\ &= k + \frac{u}{\rho} + (\epsilon_1 + \epsilon_2 + \epsilon_1\epsilon_2)\frac{r}{\rho} =: k + R \end{aligned}$$

où R est la valeur à borner (dans un sens ou l'autre).



Preuves

On appelle ϵ_1 and ϵ_2 les erreurs d'arrondi telles que :

$$\text{invp} = \frac{1}{\rho}(1 + \epsilon_1), \quad x = (r \cdot \text{invp})(1 + \epsilon_2).$$

Donc la valeur calculée est

$$\begin{aligned} x &= \frac{r}{\rho}(1 + \epsilon_1)(1 + \epsilon_2) \\ &= k + \frac{u}{\rho} + (\epsilon_1 + \epsilon_2 + \epsilon_1\epsilon_2)\frac{r}{\rho} =: k + R \end{aligned}$$

où R est la valeur à borner (dans un sens ou l'autre).

$$|\epsilon_i| \leq 2^{1-\beta}, \quad i \in \{1, 2\}.$$



Preuves sur $\lfloor x \rfloor$.

Lemme

On a toujours :

$$k - 1 \leq \lfloor x \rfloor \leq k + 1.$$



Preuves sur $\lfloor x \rfloor$.

Lemme

On a toujours :

$$k - 1 \leq \lfloor x \rfloor \leq k + 1.$$

Premier résultat : $R < 2$

$$\begin{aligned} R &\leq \frac{p-1}{p} + (2^{2-\beta} + 2^{2-2\beta}) \frac{r}{p} & (1) \\ &\leq 1 - \frac{1}{p} + (2^{2-\beta} + 2^{2-2\beta}) \frac{2^\beta - 1}{p} \\ &= 1 - \frac{1}{p} (3 - 2^{2-2\beta}). \end{aligned}$$



Preuves sur $\lfloor x \rfloor$.

Lemme

On a toujours :

$$k - 1 \leq \lfloor x \rfloor \leq k + 1.$$

Premier résultat : $R < 2$

$$\begin{aligned} R &\leq \frac{p-1}{p} + (2^{2-\beta} + 2^{2-2\beta}) \frac{r}{p} & (1) \\ &\leq 1 - \frac{1}{p} + (2^{2-\beta} + 2^{2-2\beta}) \frac{2^\beta - 1}{p} \\ &= 1 - \frac{1}{p} (3 - 2^{2-2\beta}). \end{aligned}$$

Donc $R < 2$ pour $p \geq 3$.



Preuves sur $\lfloor x \rfloor$.

Deuxième résultat : $R \geq -1$

$$\begin{aligned} R &\geq -(2^{2-\beta} + 2^{2-2\beta}) \frac{r}{p} \\ &\geq -\frac{1}{p}(4 - 2^{2-2\beta}) \end{aligned}$$



Preuves sur $\lfloor x \rfloor$.

Deuxième résultat : $R \geq -1$

$$\begin{aligned} R &\geq -(2^{2-\beta} + 2^{2-2\beta}) \frac{r}{p} \\ &\geq -\frac{1}{p}(4 - 2^{2-2\beta}) \end{aligned}$$

et $R \geq -1$ pour $p \geq 4$. Pour $p = 3$, on analyse plus précisément ϵ_1 :
 $\frac{1}{3} = 0,01010101010_2 \dots$ implique

$$\circ\left(\frac{1}{3}\right) = \begin{cases} \frac{1}{3}(1 + 2^{-\beta-1}) & \text{si } \beta \text{ est pair et } \circ(\cdot) \in \{\Delta(\cdot), \diamond(\cdot)\}, \\ \frac{1}{3}(1 + 2^{-\beta-1}) & \text{si } \beta \text{ est impair et } \circ(\cdot) \in \{\diamond(\cdot), \nabla(\cdot)\}, \\ \frac{1}{3}(1 + 2^{-\beta}) & \text{sinon.} \end{cases}$$



Preuves sur $\lfloor x \rfloor$.

Deuxième résultat : $R \geq -1$

$$\begin{aligned} R &\geq -(2^{2-\beta} + 2^{2-2\beta}) \frac{r}{p} \\ &\geq -\frac{1}{p}(4 - 2^{2-2\beta}) \end{aligned}$$

et $R \geq -1$ pour $p \geq 4$. Pour $p = 3$, on analyse plus précisément ϵ_1 :
 $\frac{1}{3} = 0,01010101010_2 \dots$ implique

$$\circ\left(\frac{1}{3}\right) = \begin{cases} \frac{1}{3}(1 + 2^{-\beta-1}) & \text{si } \beta \text{ est pair et } \circ(\cdot) \in \{\Delta(\cdot), \diamond(\cdot)\}, \\ \frac{1}{3}(1 + 2^{-\beta-1}) & \text{si } \beta \text{ est impair et } \circ(\cdot) \in \{\diamond(\cdot), \nabla(\cdot)\}, \\ \frac{1}{3}(1 + 2^{-\beta}) & \text{sinon.} \end{cases}$$

Et dans tous les cas $|\epsilon_1| \leq 2^{-\beta}$.

Preuves sur $\lfloor x \rfloor$.

Lemme

Dans les cas 1, 2, 3 et 4, $\lfloor x \rfloor \geq k$.



Preuves sur $\lfloor x \rfloor$.

Lemme

Dans les cas 1, 2, 3 et 4, $\lfloor x \rfloor \geq k$.

Cas 1, 2, et 3

Dans ces cas, $invp$ est arrondi vers le haut et donc

$$r \cdot invp = k(p \cdot invp) + u \cdot invp \geq k.$$



Preuves sur $\lfloor x \rfloor$.

Lemme

Dans les cas 1, 2, 3 et 4, $\lfloor x \rfloor \geq k$.

Cas 1, 2, et 3

Dans ces cas, $invp$ est arrondi vers le haut et donc

$$r \cdot invp = k(p \cdot invp) + u \cdot invp \geq k.$$

En particulier $\circ(r \cdot invp) \geq k$ car les arrondis sont monotones et k est représentable.



Preuves sur $\lfloor x \rfloor$.

Cas 4 : $x = \lfloor \Delta(r \cdot \diamond(1/p)) \rfloor$

Puisque $|\epsilon_1| \leq 2^{-\beta}$, on a

$$\text{inv}p > (1 - 2^{-\beta}) \frac{1}{p}.$$



Preuves sur $\lfloor x \rfloor$.

Cas 4 : $x = \lfloor \Delta(r \cdot \diamond(1/p)) \rfloor$

Puisque $|\epsilon_1| \leq 2^{-\beta}$, on a

$$\text{inv}p > (1 - 2^{-\beta}) \frac{1}{p}.$$

Donc

$$r \cdot \text{inv}p = (kp + u)\text{inv}p \geq kp \cdot \text{inv}p > k(1 - 2^{-\beta}).$$



Preuves sur $\lfloor x \rfloor$.

Cas 4 : $x = \lfloor \Delta(r \cdot \diamond(1/p)) \rfloor$

Puisque $|\epsilon_1| \leq 2^{-\beta}$, on a

$$\text{inv}p > (1 - 2^{-\beta}) \frac{1}{p}.$$

Donc

$$r \cdot \text{inv}p = (kp + u)\text{inv}p \geq kp \cdot \text{inv}p > k(1 - 2^{-\beta}).$$

k est un entier donc exactement représentable. On appelle k^- le plus grand nombre flottant $< k$:

$$k^- = \begin{cases} k - \frac{1}{2}\text{ulp}(k) & \text{si } k \text{ est une puissance de } 2, \\ k - \text{ulp}(k) & \text{sinon.} \end{cases}$$



Preuves sur $\lfloor x \rfloor$.

Cas 4 (suite)

Si k est une puissance de 2, alors on arrondit $k(1 - 2^{-\beta})$ comme on arrondit $1 - 2^{-\beta}$ car k ne change que l'exposant.



Preuves sur $\lfloor x \rfloor$.

Cas 4 (suite)

Si k est une puissance de 2, alors on arrondit $k(1 - 2^{-\beta})$ comme on arrondit $1 - 2^{-\beta}$ car k ne change que l'exposant.

$$\Delta(1 - 2^{-\beta} + \delta) \geq 1 \text{ pour tout } \delta > 0.$$



Preuves sur $\lfloor x \rfloor$.

Cas 4 (suite)

Si k est une puissance de 2, alors on arrondit $k(1 - 2^{-\beta})$ comme on arrondit $1 - 2^{-\beta}$ car k ne change que l'exposant.

$$\Delta(1 - 2^{-\beta} + \delta) \geq 1 \text{ pour tout } \delta > 0.$$

$$x = \Delta(r \cdot invp) \geq k.$$



Preuves sur $\lfloor x \rfloor$.

Cas 4 (suite)

Si k est une puissance de 2, alors on arrondit $k(1 - 2^{-\beta})$ comme on arrondit $1 - 2^{-\beta}$ car k ne change que l'exposant.

$$\Delta(1 - 2^{-\beta} + \delta) \geq 1 \text{ pour tout } \delta > 0.$$

$$x = \Delta(r \cdot invp) \geq k.$$

Si k n'est pas une puissance de 2 :

$$k^- = k - \text{ulp}(k) < k(1 - 2^{-\beta}) < r \cdot invp$$

et alors $x = \Delta(r \cdot invp) \geq k$.



Preuves (bornes sur r)

Cas 1 : $x = \lfloor \Delta(r \cdot \Delta(1/p)) \rfloor$

Dans ce cas $0 \leq \epsilon_i \leq 2^{1-\beta}$ pour $i \in \{1, 2\}$.



Preuves (bornes sur r)

Cas 1 : $x = \lfloor \Delta(r \cdot \Delta(1/p)) \rfloor$

Dans ce cas $0 \leq \epsilon_i \leq 2^{1-\beta}$ pour $i \in \{1, 2\}$.

$$R \leq \frac{p-1}{p} + (2^{2-\beta} + 2^{2-2\beta}) \frac{r}{p}$$

et il suffit pour $|R| < 1$ que

$$r < 2^\beta \frac{1}{4 + 2^{2-\beta}}.$$



Preuves (bornes sur r)

Cas 2 : $x = \lfloor \diamond(r \cdot \Delta(1/p)) \rfloor$

Dans ce cas $|\epsilon_2| \leq 2^{-\beta}$



Preuves (bornes sur r)

Cas 2 : $x = \lfloor \diamond(r \cdot \Delta(1/p)) \rfloor$

Dans ce cas $|\epsilon_2| \leq 2^{-\beta}$

$$R \leq \frac{p-1}{p} + (3 \cdot 2^{-\beta} + 2^{1-2\beta}) \frac{r}{p}.$$



Preuves (bornes sur r)

Cas 2 : $x = \lfloor \diamond(r \cdot \Delta(1/p)) \rfloor$

Dans ce cas $|\epsilon_2| \leq 2^{-\beta}$

$$R \leq \frac{p-1}{p} + (3 \cdot 2^{-\beta} + 2^{1-2\beta}) \frac{r}{p}.$$

On a $R < 1$ dès que

$$3 \cdot 2^{-\beta} + 2^{1-2\beta} < \frac{1}{r}$$

c'est-à-dire $r < 2^\beta / (3 + 2^{1-\beta})$.



Preuves (bornes sur r)

Cas 3 : $x = \lfloor \nabla(r \cdot \Delta(1/p)) \rfloor$

Dans ce cas $-2^{1-\beta} \leq \epsilon_2 \leq 0$, $|\epsilon_1 + \epsilon_2| \leq 2^{1-\beta}$ et $\epsilon_1 \epsilon_2 \leq 0$.



Preuves (bornes sur r)

Cas 3 : $x = \lfloor \nabla(r \cdot \Delta(1/p)) \rfloor$

Dans ce cas $-2^{1-\beta} \leq \epsilon_2 \leq 0$, $|\epsilon_1 + \epsilon_2| \leq 2^{1-\beta}$ et $\epsilon_1 \epsilon_2 \leq 0$.

$$R \leq 1 - \frac{1}{p} + \frac{r}{p} 2^{1-\beta}$$

et la condition $R < 1$ est assurée par

$$r < \frac{1}{2} 2^\beta.$$



Preuves (bornes sur r)

Cas 4 : $x = \lfloor \Delta(r \cdot \diamond(1/p)) \rfloor$

Même preuve que pour le cas 2 car $\Delta(\cdot)$ et $\diamond(\cdot)$ jouent un rôle symétrique dans l'analyse.



Optimalité des bornes

- Certaines bornes sur r sont asymptotiquement optimales (famille de cas paramétrées par β);



Optimalité des bornes

- Certaines bornes sur r sont asymptotiquement optimales (famille de cas paramétrées par β);
- Le meilleur pire cas générique pour le choix d'arrondis 3 atteint seulement $\frac{3}{8}$ au lieu de $\frac{1}{3}$:

$$\beta = 2n + 1$$

$$p = 2^n - 1$$

$$r = (3 \cdot 2^{n-2} + 3)p - 1;$$



Optimalité des bornes

- Certaines bornes sur r sont asymptotiquement optimales (famille de cas paramétrées par β);
- Le meilleur pire cas générique pour le choix d'arrondis 3 atteint seulement $\frac{3}{8}$ au lieu de $\frac{1}{3}$:

$$\beta = 2n + 1$$

$$p = 2^n - 1$$

$$r = (3 \cdot 2^{n-2} + 3)p - 1;$$

- Pas de pire cas générique pour le cas 4.



Utiliser FDIV

Require: r , integer such that $0 \leq r \leq 2^\beta - 1$;

Require: p , integer such that $1 \leq p \leq 2^\beta - 1$;

Ensure: $\lfloor \frac{r}{p} \rfloor$.

Constants

1: $B_\Delta \leftarrow 2^\beta / (3 + 2^{1-\beta})$

2: $B_\diamond \leftarrow 2^\beta / (3 + 2^{1-\beta})$

3: $B_\nabla \leftarrow 2^{\beta-1}$

Precomputation

4: $invp_\Delta \leftarrow \diamond(1/p)$

5: $invp_\diamond \leftarrow \Delta(1/p)$

6: $invp_\nabla \leftarrow \Delta(1/p)$



Utiliser FDIV (suite)

- 1: $x \leftarrow \circ(r \cdot invp_o)$
- 2: $y \leftarrow \lfloor x \rfloor$

Possible correction

- 3: **if** $r \geq B_o$ **then**
- 4: $z \leftarrow p \cdot y$
- 5: **if** $z > r$ **then**
- 6: $y \leftarrow y - 1$
- 7: **end if**
- 8: **end if**
- 9: Return y .



Plan

- 1 Problématique
- 2 Deux idées magiques
 - Substitution de Kronecker
 - REDQ
- 3 Division euclidienne en flottants
- 4 Applications
 - Multiplication polynomiale
 - Multiplication matricielle compressée
- 5 Conclusion



Plan

- 1 Problématique
- 2 Deux idées magiques
 - Substitution de Kronecker
 - REDQ
- 3 Division euclidienne en flottants
- 4 Applications
 - Multiplication polynomiale
 - Multiplication matricielle compressée
- 5 Conclusion



Stratégie classique

$P(X) = \sum_i a_i X^i$, $Q(X) = \sum_i b_i X^i$ polynômes mod p .

Réduction retardée

- On accumule $\sum_i a_i b_{k-i}$ que l'on réduit une fois.



Stratégie classique

$P(X) = \sum_i a_i X^i$, $Q(X) = \sum_i b_i X^i$ polynômes mod p .

Réduction retardée

- On accumule $\sum_i a_i b_{k-i}$ que l'on réduit une fois.
- Accumulation jusqu'à n_d produits valide tant que

$$n_d(p-1)^2 < 2^{\beta+1}$$

en représentation centrée.

- On appelle REDC l'algorithme de réduction.



Fast Q-adic Transform

On représente $P(X) = \sum_{i=0}^N a_i X^i$ sous la forme :

$P(X) = \sum P_i(X) \cdot (X^{d+1})^i$ où les $P_i(X)$ sont des polynômes de degré d stockés dans un seul entier (en q -adique).

Alors un produit PQ s'écrit

$$PQ = \sum \left(\sum P_i Q_{t-i} \right) (X^{d+1})^t,$$

où chaque multiplication $P_i Q_{t-i}$ est calculé par l'algorithme DQT.
La réduction finale est assurée par un appel tabulé à REDQ.



Complexités

P de degré N en X . Avec une FQT d'ordre d , P devient un polynôme de degré D_q en $Y = X^{d+1}$, avec

$$D_q = \left\lceil \frac{N+1}{d+1} \right\rceil - 1.$$

$$n_d = \left\lfloor \frac{2^{\beta+1}}{(p-1)^2} \right\rfloor, \quad n_q = \left\lfloor \frac{q}{(d+1)(p-1)^2} \right\rfloor.$$

| | Mul & Add | Reductions |
|---------|-------------------------|---|
| Delayed | $(2N+1)^2$ | $(2N+1) \left\lceil \frac{2N+1}{n_d} \right\rceil$ REDC |
| d-FQT | $\frac{1}{2}(2D_q+1)^2$ | $(2D_q+1) \left\lceil \frac{2D_q+1}{n_q} \right\rceil$ REDQ $_{2d+1}$ |



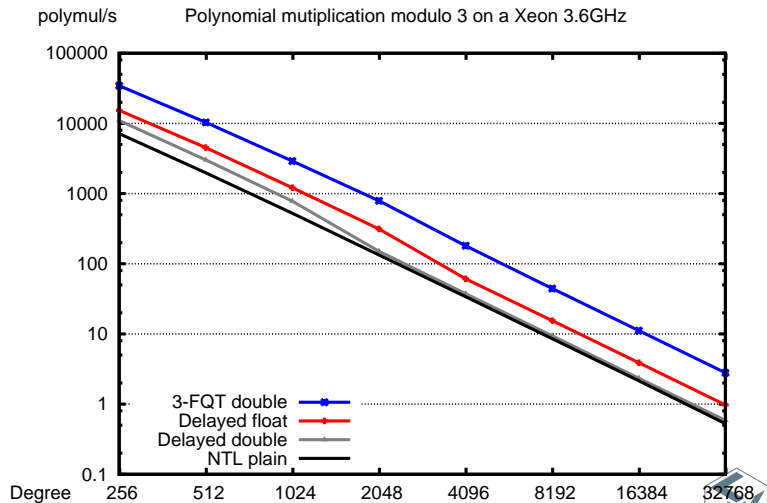
Complexités

Exemple : $p = 3$, $N = 500$.

| Algorithme | Mul & Add | Reductions |
|--------------------------|----------------|----------------|
| Classique | 10^6 | 10^3 |
| 4-FQT (flottant, tabulé) | $5 \cdot 10^4$ | $4 \cdot 10^3$ |

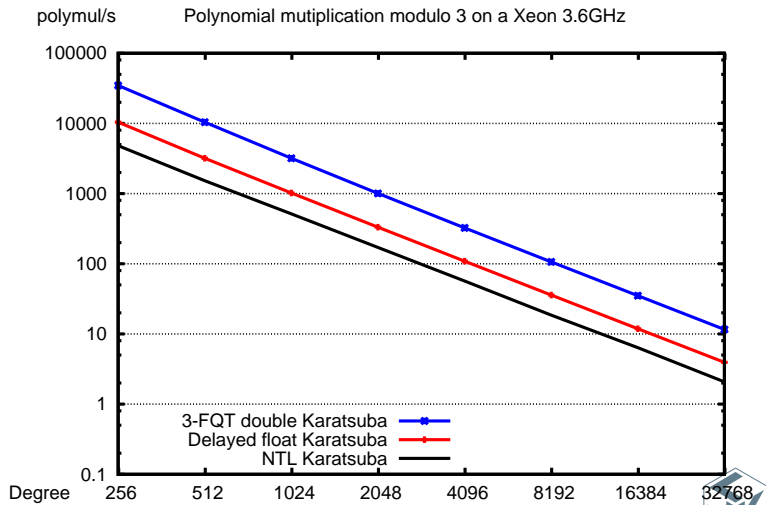


Résultats exp



MATHÉMATIQUES APPLIQUÉES - INFORMATIQUE

Résultats exp



Plan

- 1 Problématique
- 2 Deux idées magiques
 - Substitution de Kronecker
 - REDQ
- 3 Division euclidienne en flottants
- 4 Applications
 - Multiplication polynomiale
 - **Multiplication matricielle compressée**
- 5 Conclusion



Prélude : comment faire un produit scalaire

$$a = (a_0, a_1, \dots, a_k)$$

$$b = (b_0, b_1, \dots, b_k)$$

$$\langle a^T, b \rangle = \sum_{i=0}^k a_i b_{k-i}$$

$$A(X) = a_0 + a_1 X + \dots + a_k X^k$$

$$B(X) = b_0 + b_1 X + \dots + b_k X^k$$

$$A \cdot B = \sum_{i=0}^{2k} X^i \left(\sum_{l+j=i} a_l b_j \right)$$

$$[X^k] A \cdot B = \langle a^T, b \rangle$$



Cas dimension 2

Le produit

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix}$$

se déduit de

$$\begin{bmatrix} Qa + b \\ Qc + d \end{bmatrix} \times [e + Qg \quad f + Qh] = \begin{bmatrix} * + (ae + bg)Q + * Q^2 & * + (af + bh)Q + * Q^2 \\ * + (ce + dg)Q + * Q^2 & * + (cf + dh)Q + * Q^2 \end{bmatrix}$$

où * indique des coefficients inutiles.



Multiplication de matrices compressées (CMM)

En général :

- La matrice A de dimension $m \times k$ est compressée en une matrice $m \times \lceil k/(d+1) \rceil$;
- La matrice B de dimension $k \times n$ est compressée en une matrice de dimension $\lceil k(d+1) \rceil \times n$;
- Gain d'un facteur jusqu'à $d+1$ en nombre d'opérations arithmétiques ;
- Seule la partie haute des coefficients calculés est nécessaire.



Multiplication de matrices compressées (CMM)

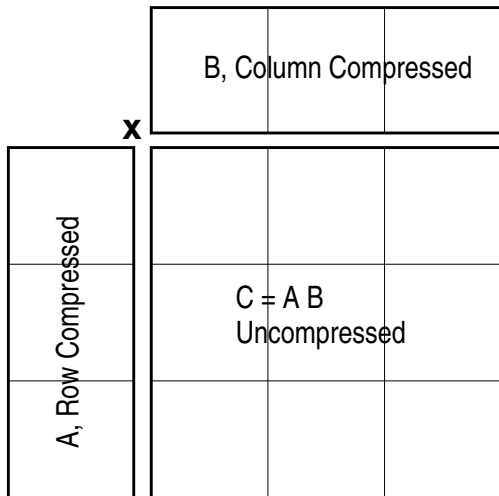


FIG.: Multiplication de matrice compressée (CMM).



Performances

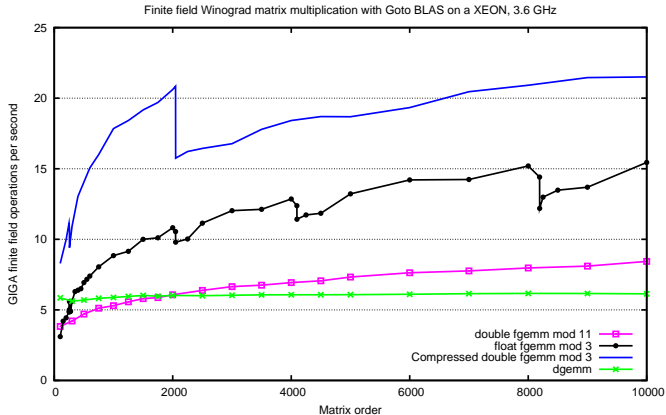


FIG.: Comparaison entre la multiplication de matrices compressée et $dgemm$ (la multiplication de matrices flottants double précision de GotoBlas) et $fgemm$ (la routine exacte de FFLAS) en simple ou double précision.



Performances

| | | | | | | | | |
|-------------|-------|----------|----------|-----------|-----------|-----------|------------|-------------|
| Compression | 2 | 3..4 | 5..8 | 8 | 7 | 6 | 5 | 4 |
| Degré d | 1 | 5 | 9 | 7 | 6 | 5 | 4 | 3 |
| Q-adic | 2^3 | 2^4 | 2^5 | 2^6 | 2^7 | 2^8 | 2^{10} | 2^{13} |
| Dimensions | 2 | ≤ 4 | ≤ 8 | ≤ 16 | ≤ 32 | ≤ 64 | ≤ 256 | ≤ 2048 |

TAB.: Facteurs de compression pour différentes dimensions de matrice modulo 3, avec 53 bits de mantisse et Q une puissance de 2.



Compression gauche/droite

Il est possible de compresser différemment :

- Conserver A non compressée $m \times k$;



Compression gauche/droite

Il est possible de compresser différemment :

- Conserver A non compressée $m \times k$;
- Compresser B par lignes $k \times n/(d + 1)$.



Compression gauche/droite

Il est possible de compresser différemment :

- Conserver A non compressée $m \times k$;
- Compresser B par lignes $k \times n/(d + 1)$.

Par exemple en dimension 2 :

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \times \begin{bmatrix} e + Qf \\ g + Qh \end{bmatrix} = \begin{bmatrix} (ae + bg) + Q(af + bh) \\ (ce + dg) + Q(cf + dh) \end{bmatrix}$$



Compression gauche/droite

- Différence dans la réduction : tous les coefficients sont nécessaires (REDQ).
- Il faut donc

$$k(p-1)^2 < Q$$

et

$$Q^{d+1} < 2^\beta.$$



Toutes les compressions

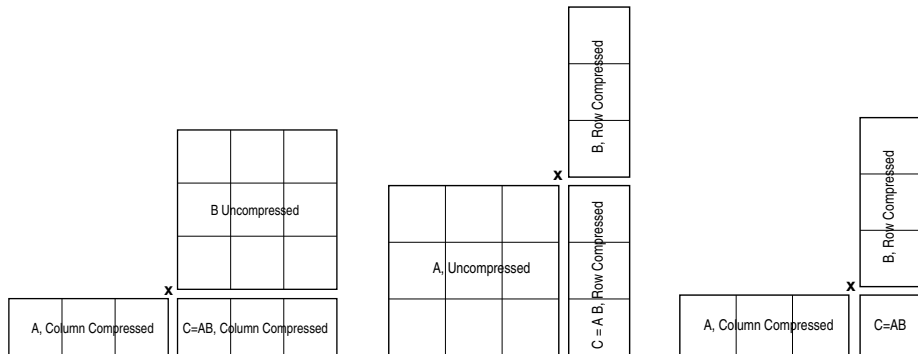


FIG.: Compressions gauche, droite et complète.



Comparaison des compressions

| Algorithme | Opérations | Réductions | Conversions |
|-------------|--|--|--------------------------------------|
| CMM | $\mathcal{O}\left(mn \left(\frac{k}{e}\right)^{\omega-2}\right)$ | $m \times n$ REDC | $\frac{1}{e}mn$ INIT _e |
| Right Comp. | $\mathcal{O}\left(mk \left(\frac{n}{e}\right)^{\omega-2}\right)$ | $m \times \frac{n}{e}$ REDQ _e | $\frac{1}{e}mn$ EXTRACT _e |
| Left Comp. | $\mathcal{O}\left(nk \left(\frac{m}{e}\right)^{\omega-2}\right)$ | $\frac{m}{e} \times n$ REDQ _e | $\frac{1}{e}mn$ EXTRACT _e |
| Full Comp. | $\mathcal{O}\left(k \left(\frac{mn}{e}\right)^{\frac{\omega-1}{2}}\right)$ | $\frac{m}{\sqrt{e}} \times \frac{n}{\sqrt{e}}$ REDQ _e | $\frac{1}{e}mn$ INIT _e |

TAB.: Nombre d'opérations arithmétiques pour les différents algorithmes.



Comparaison des compressions

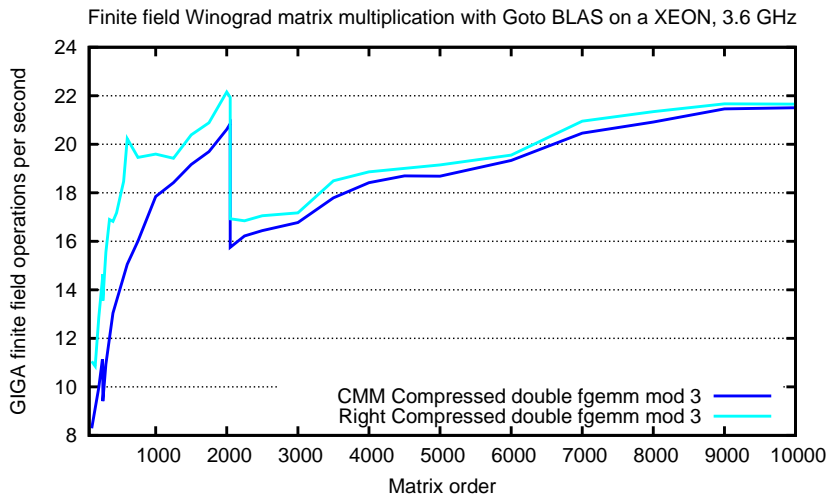


FIG.: CMM et compression droite.



Plan

- 1 Problématique
- 2 Deux idées magiques
 - Substitution de Kronecker
 - REDQ
- 3 Division euclidienne en flottants
- 4 Applications
 - Multiplication polynomiale
 - Multiplication matricielle compressée
- 5 Conclusion



Conclusion

Fait

- Algorithme de réduction simultanée de plusieurs résidus stockés dans un seul mot machine (avec compromis temps/mémoire) ;
- Analyse de la division euclidienne en flottants ;
- Application au produit de polynômes ;
- Application au produit de matrices.



Conclusion

Fait

- Algorithme de réduction simultanée de plusieurs résidus stockés dans un seul mot machine (avec compromis temps/mémoire) ;
- Analyse de la division euclidienne en flottants ;
- Application au produit de polynômes ;
- Application au produit de matrices.

À faire

- Explorer les choix possibles de q plus en détail ;
- Arithmétique multi-précision compressée ?
- Regarder *Correctly Rounded Multiplication by Arbitrary Precision Constants* [Brisebarre et Muller 2008]

