円分体の相対類数の高速計算アルゴリズムについて

東京理科大学大学院 理工学研究科 吾郷研究室
谷口　哲也

本日の講演内容.

- 円分体の相対類数 h_p^- の高速アルゴリズムの紹介
- 本アルゴリズムの特徴：絶対ノルムの高速化
  - 2次元FFT, Galois群の作用の「圧縮」
  - 最良$O(p(\lg p)^3 (\lg\lg p))$, 最悪$O(p^2 (\lg p)^2 (\lg\lg p))$
- 世界記録の更新
  - 旧記録 $p<10000$ → 新記録 $p<$**45000**
  - $p=$**8503057**
- 本アルゴリズムの応用
  - Det(巡回行列), Res(f(x), x^n-1) の高速計算

使用公式：解析的類数公式.

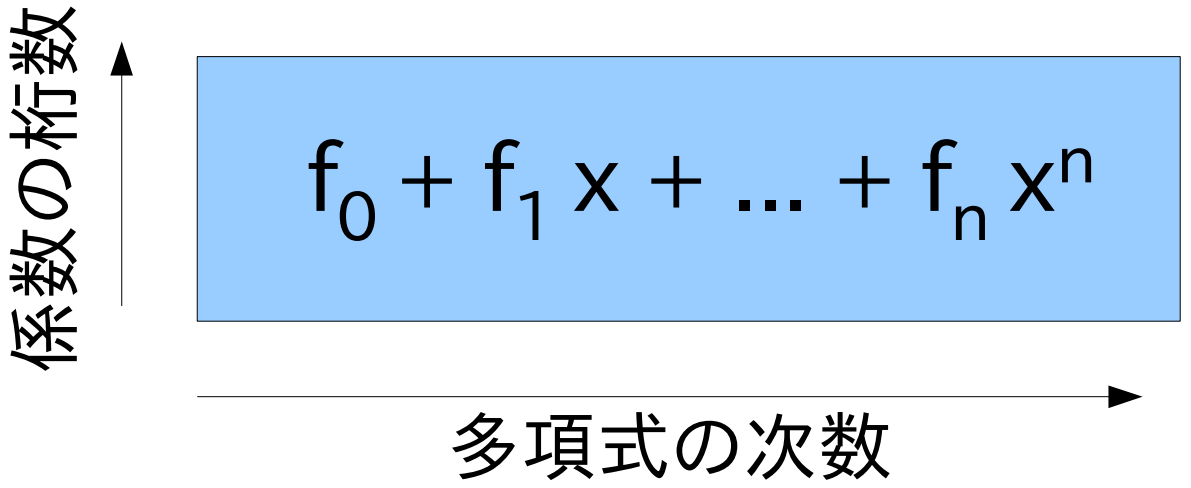
- $(2p)^{(p-3)/2}\, h_p^-$
- $= \prod f(\zeta^{2k+1})$　(f : ∃多項式, $\zeta = e^{2\pi i/(p-1)}$)
- $= f(x^1)\, f(x^3) \ldots f(x^{p-2}) \pmod{x^{p-1}-1}$.
-
- $\rightarrow h_p^-$ は多項式の積で計算できる
- では, Mathematicaで実験してみましょう.

実験のまとめ.

- h_p^- の計算は「巨大多項式」の積
- まともにやったら大変なことになる ($O(p^5)$以上).
-
- 高速化の方針
  - 積の計算を高速化　→ 2次元FFT
  - 積の回数を削減　→ 絶対ノルム
  - 　　相対ノルム
  - 　　Galois群を圧縮

記号説明

「巨大多項式」を模式的に次の図で表す:

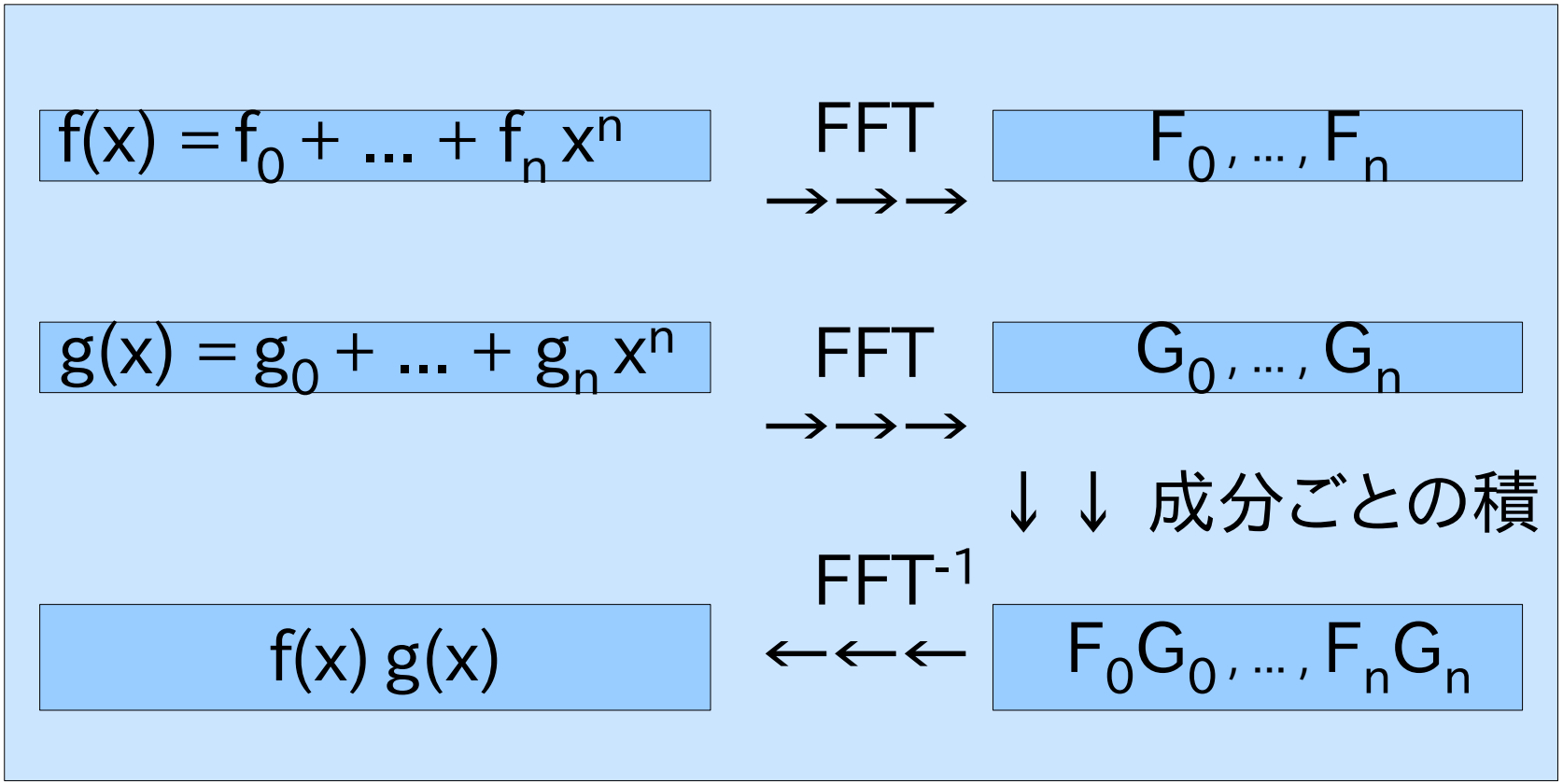


積の高速化(1/4)　筆算方式

- 筆算方式
  - $f(x) = f_0 + \ldots + f_n x^n$
  - $g(x) = g_0 + \ldots + g_n x^n$
  - $f(x)\,g(x) = (\Sigma f_i g_j) x^k$
  - 計算量 : $O(n^2)$
- これは遅い!!

積の高速化(2/4)　FFT乗算.

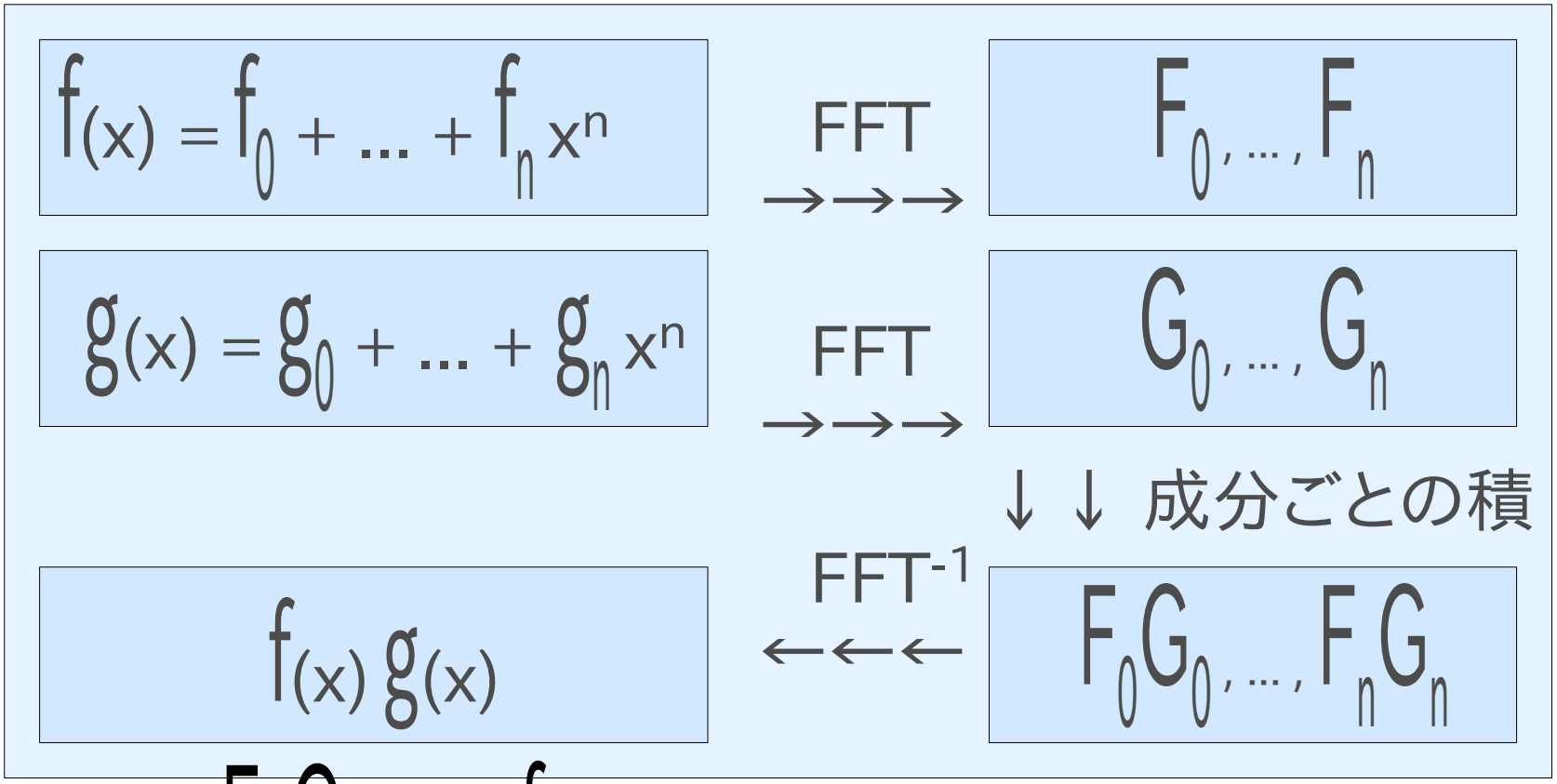
多項式 f(x), g(x) の高速乗算



計算量 : O(n (lg n) (lglg n))　高速!
ただし係数 f_i g_i F_i G_i はdouble型でないとダメ

積の高速化(3/4)　FFT乗算

FFT乗算の枠組を「巨大多項式」に当てはめる

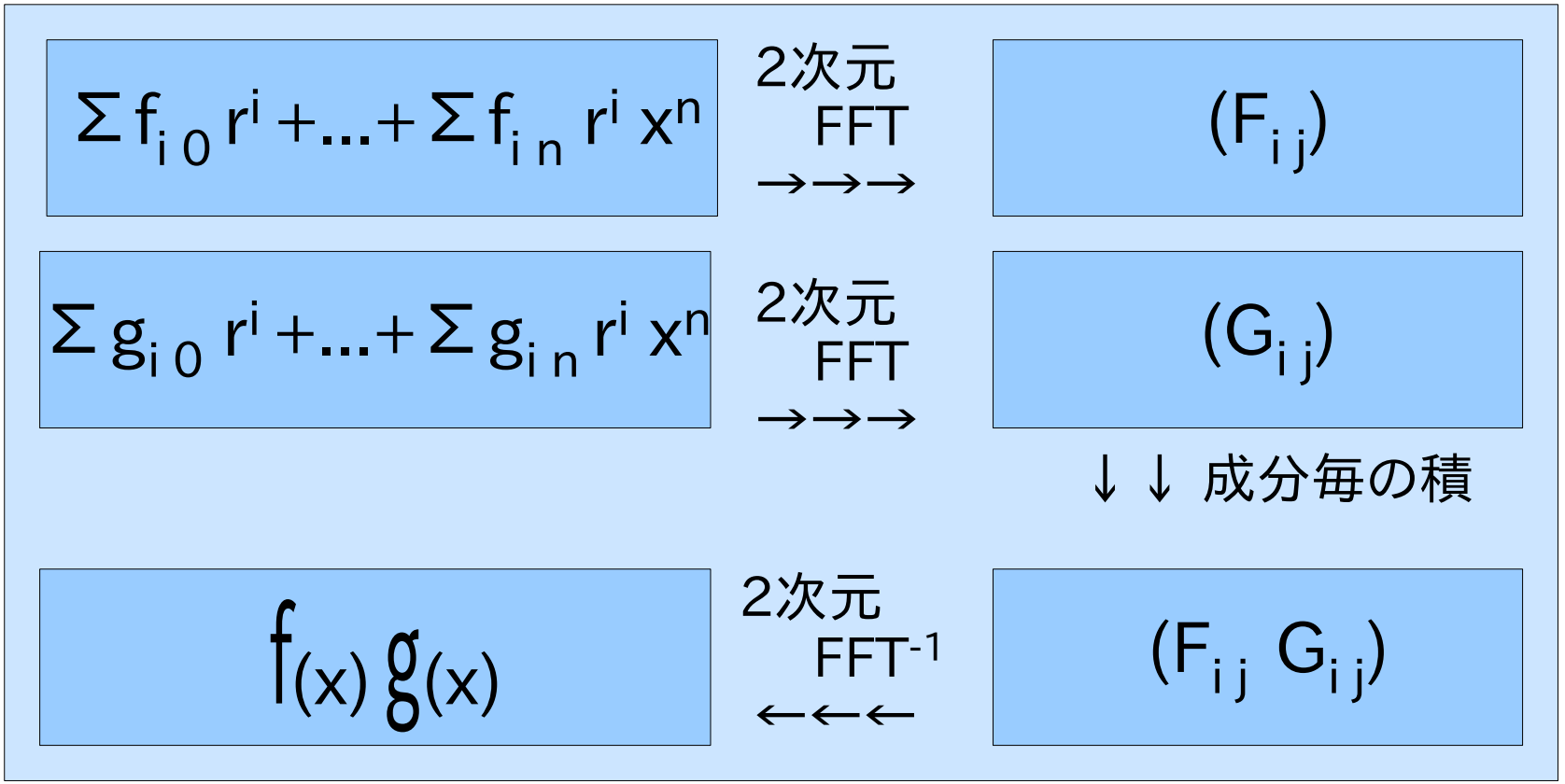


問題点：$F_i, G_i, \ldots, f_i, g_i$ が多倍長数!!

通常のFFTでは計算できない→ さてどうするか.

積の高速化(4/4)　2次元FFT乗算

係数をr進数で表示し, 2次元FFTをすればOK



計算量はO(mn{ (lg m) (lglg m) + (lg n) (lglg n) })

(mは係数のr進での桁数)

積の回数削減の準備

- 代数体のノルム
  - L/K, $f \in L$ に対し, $N_{L/K}(f) \in K$　　(K共役の積)
  - 今はLとして円分体を想定
  - →ノルムはGalois群の作用で書ける
- 積の回数削減の方針:
  - h_p^- の計算を円分体のノルムに帰着し, 構造をフルに利用

積の回数削減(1/4)　絶対ノルムへ

$$(2p)^{(p-3)/2}\, h_p^-$$

$$= \prod f(\zeta_{p-1}^{2k+1})$$

$$= \prod_d \prod_{(p-1,2k+1)=d} f(\zeta_{p-1}^{2k+1})$$

$$= \prod_d \prod_{(p-1,2k+1)=d} f(\zeta_{(p-1)/d}^{(2k+1)/d})$$

$$= \prod_d \prod_{\alpha} \alpha(f(\zeta_{(p-1)/d}))$$

$$= \prod_d N_d(f(\zeta_{(p-1)/d}))$$

f：∃多項式, $\zeta = e^{2\pi i/(p-1)}$

$d \mid p-1$, d:奇数

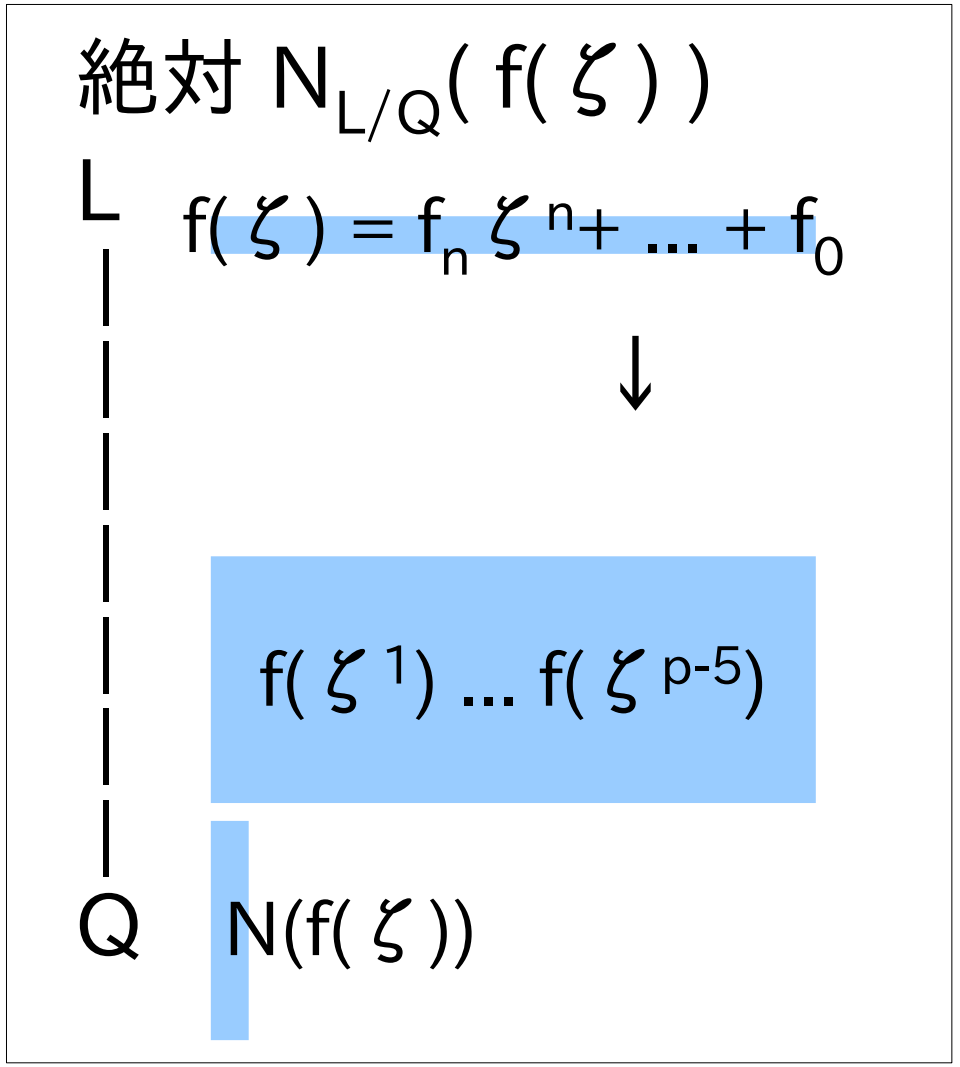
$\zeta_{(p-1)}^{\,d} = \zeta_{(p-1)/d}$

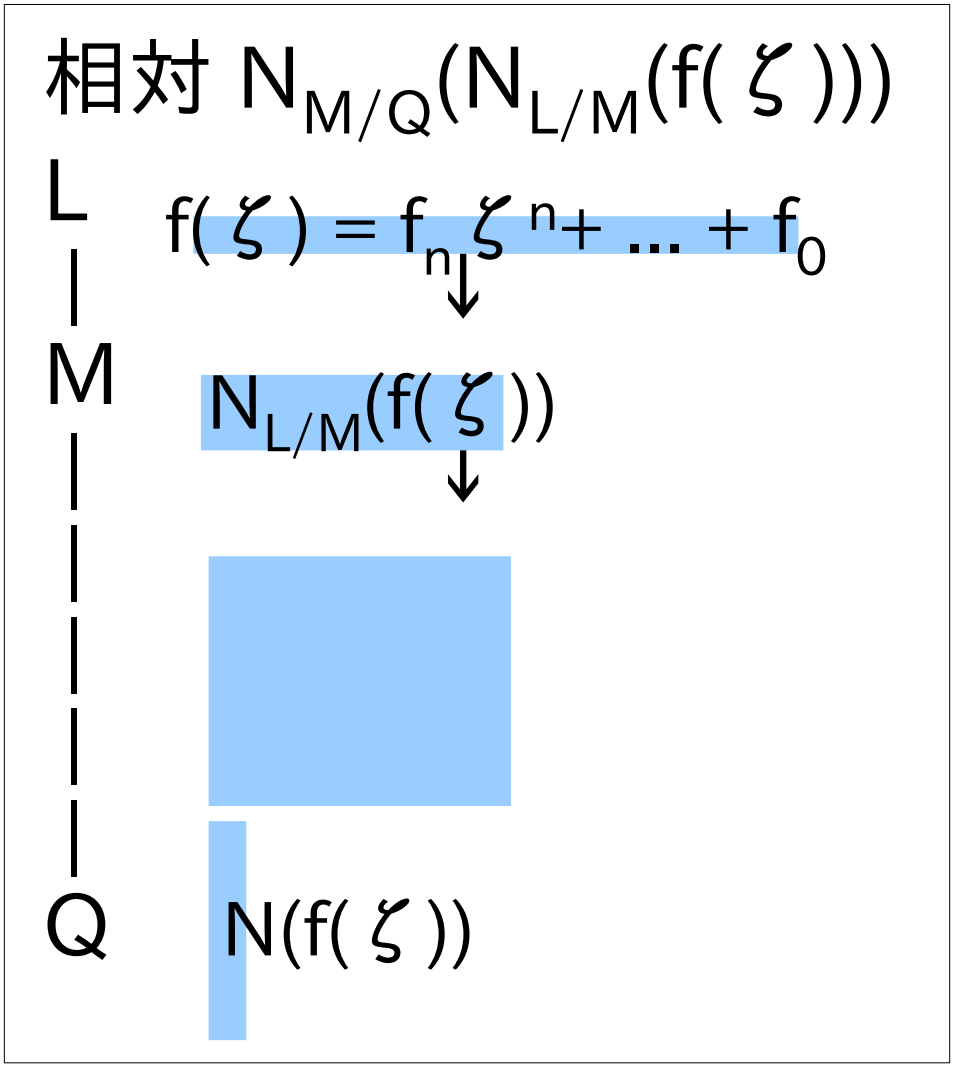
$\alpha \in \mathrm{Gal}(Q(\zeta_{(p-1)/d})/Q)$

$N_d = \mathrm{Norm}\ Q(\zeta_{(p-1)/d})/Q$

※副産物：h_p^- の部分的な因数分解

積の回数削減(2/4)　相対ノルムへ.





中間体を経由した方が計算のサイズが小さくなる!

積の回数削減(3/4)　相対ノルムへ.

まとめ：絶対ノルムを相対ノルムの合成で表す
中間体の列

$$L=Q(\zeta) = L_m \supset \ldots \supset L_0 = Q$$

$$N_{L/Q} = N_{L_1/L_0} \circ N_{L_2/L_1} \circ \ldots \circ N_{L_m/L_{m-1}}$$

と分解して計算する.

中間体　多→サイズ縮小，高速化！
中間体　少→サイズ縮まらない　　→さて...

実は...

中間体が少なくても
積の回数削減可能

→Galois群を「圧縮」

積の回数削減(4/4)　Galois群圧縮.

Galois群を「圧縮」する

$f \in L$, $Gal(L/M)=\{1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6, \alpha^7\}$

$N_{L/M}(f)$　　↓乗算:7回

$= f\ \alpha(f)\ \alpha^2(f)\ \alpha^3(f)\ \alpha^4(f)\ \alpha^5(f)\ \alpha^6(f)\ \alpha^7(f)$

$= f\ \alpha(f)\ \alpha^2(f)\ \alpha^3(f)\ \alpha^4\{ f\ \alpha(f)\ \alpha^2(f)\ \alpha^3(f) \}$

再帰的に適用

$f_1 := f\ \alpha(f), \quad f_2 := f_1\ \alpha^2(f_1), \quad f_3 := f_2\ \alpha^4(f_2)$

$\rightarrow \quad N_{L/M}(f) = f_3$　←乗算:3回

乗算回数 : O(p) → O(lg p)

本アルゴリズムのまとめ..

多項式の積　→2次元FFT
積の回数　→絶対・相対ノルム, Galois群圧縮

- 計算量全体の評価
  - p-1 が細かく分解されるときほど速い
  - 最良時　$O(p (\lg p)^3 (\lg\lg p))$
  - 最悪時　$O(p^2 (\lg p)^2 (\lg\lg p))$
  -
- メモリ使用量削減の工夫
  - 最初の多項式生成時に係数が0の所を沢山作る
  - （桶屋氏の指摘による　感謝）

計算環境

- 使用計算機
  - 「Core2Duo　2.66Ghz,　L2 4MB,　4GB」　×2台
  - 「Pentium4(HT) 2.8Ghz, L2 1MB,　2GB」　×2台
  - 「PentiumD　　2.8Ghz, L2 2MB,　1GB」　×1台
- 使用言語:
  - Gcc ver 4.1.2
  - GMP 4.2.1
  - OouraFFT
- プログラムの規模
  - 4000行弱

結果詳細(記録).

- 一般の素数 p の全数検査
  - $p<45000$ に対して h_p^- を計算
- 特殊な素数 p の全数検査
  - 以下の形の p に対して h_p^- を計算
  - $p=2^a3^b+1$, $p<10000000$
  - $p=2^a3^b5^c7^d+1$, $p<2000000$
- 最大記録 $p = 8503057 = 2^4 3^{12} + 1$
  - h_p^- の値は10進で 1133万7165桁
  - 部分分解：755万桁×251万桁×83万桁×...

計算速度

- 今回の記録
  - $p < 10000$　　2時間半　　(Core2　2.66Ghz)
  - $p = 5038849$　23時間　　(Core2　2.66Ghz)
  - $p = 5308417$　23時間　　(Core2　2.66Ghz)
  - $p = 8503057$　50時間　　(Core2　2.66Ghz)
  - 最良$O(p(\lg p)^3 (\lg\lg p))$, 最悪 $O(p^2 (\lg p)^2 (\lg\lg p))$
- 参考：Shokrollahi氏の記録
  - $p < 10000$　　約1.5日　　(UltraSPARC 167Mhz)
  - ERHの下で $O(p^2 (\lg p)^2 (\lg\lg p))$

データの分析

- h_p^-の偶奇性
  - $p < 45000$で, $2 \mid h_p^-$をみたすpの個数：628個
    -
  - 偶奇性に関する予想
    - $p=2q+1$, (p, q:prime)のときh_p^-は奇数
    - →今回の計算の範囲ではすべて成立

データの分析

- h_p^- の小さい素数による整除性
  - p<45000で, q | h_p^-をみたすpの個数
    - q=3, 1247個
    - q=5, 1217個
    - q=7, 812個
    - q=11, 485個
    - q=13, 723個
    - q=17, 560個
    - q=19, 382個
    - q=23, 228個
    - q=29, 319個

検算.

- 計算中の検算
  - 相対ノルム計算後, 下の体に収まるかをチェック
  -
  - 絶対ノルムの計算後, pで割れる回数をチェック
  - i.e. $(2p)^{(p-3)/2} h_p^- = \Pi_d N_d(f(\zeta))$ のp巾に注目
-
- →全てパス

本アルゴリズムの応用

- 円分体の絶対ノルムを高速化した
- 下記は円分体の絶対ノルムで計算可能
  - いくつかの数論の不変量
  - 巡回行列の行列式
  - Skew巡回行列の行列式
  - Circulant Resultant　　$\mathrm{Res}(f(x), x^n-1)$
- よって, 上記は高速計算可能

今後の課題

- メモリ使用量の削減
  - p=2q+1 (SophieGermain素数)は中間体がなく, 係数爆発
- 高速化
  - まずはメモリ使用量の削減
  - Galois群圧縮の改善(繰り返し2乗法と同様?)
  - キャッシュチューニング, マルチスレッド化
- さらなる記録更新
  - 分割乗算, 分割剰余乗算
- データの分析
  - 結果の素数判定, 素因数分解 (絶望的?)

時間があれば...

ここで計算の実演をします

ありがとうございました