Signature based algorithm における F_4 スタイルの簡約アルゴリズムの実装について

野呂 正行

立教大学

December 20, 2022

概要

- signature based algorithm (SBA) の実装を高速化したい
  F_4 類似の簡約を導入するのが自然
- F_4 の簡約アルゴリズムをそのまま適用するのは困難
  reducer の選び方に対する制限, 行列の簡約に対する制限, . . .
- 逐次 SBA のまま, 各簡約をベクトルに変換して実行
  条件付き symbolic preprocessing で reducer リスト, 単項式リストは作るが S 多項式の簡約は 1 つずつ行う
  $\Rightarrow$ それなりに高速化することができた

Contents

- signature based algorithm (SBA)
- F_4 における簡約
- SBA における F_4 スタイル簡約の問題点
- signature を考慮した symbolic preprocessing
- SBA with F_4-like reduction
- 計算機実験

Signature

- K : 体, $R = K[x_1, \ldots, x_n]$
- $I = \langle f_1, \ldots, f_m \rangle$:R のイデアル
- $<$: R の項順序
- $\prec$: $R^m = Re_1 \oplus \cdots \oplus Re_m$ の加群項順序
- M : R の係数 1 の単項式全体

定義 1 (minimal signature)

- $f \in I$, $f \neq 0$ に対し,

$$S(f) = \min_{\prec}\{\mathrm{LM}(h) \mid h = \sum_{i=1}^{m} h_i e_i \in R^m, \sum_{i=1}^{m} h_i f_i = f\}$$

を f の (minimal) signature と呼ぶ.

- $m \in M$ に対し $mS(f)$ を mf の guessed signature と呼ぶ.

㋡-既約性と ㋡-グレブナー基底

定義 2 (㋡(シグマ)-簡約, ㋡-既約)

1. $f \in I$ の先頭項を $g \in I$ で簡約する際,
$$S(f) \succ S(mg) \quad (m = \frac{\mathrm{LM}(f)}{\mathrm{LM}(g)})$$
が成り立つとき, regular (㋡-) 簡約と呼ぶ.
2. f を regular 簡約する $g \in I$ が存在しないとき f は ㋡-既約という.

定義 3 (㋡-GB)

$G \subset I$ が次を満たすとき G を I の ㋡-GB と呼ぶ.
任意の $p \in I$ に対し $g \in G, m \in M$ が存在して
$$\mathrm{LM}(p) = m\mathrm{LM}(g), S(p) \geq mS(g)$$
有限 ㋡-GB は GB である.

regular な S ペア

定義 4 (regular な S ペア)

1. $p = (g, g')$, $\mathrm{Spoly}(p) = cmg - c'm'g'$ において $mS(g) \neq m'S(g)$ なら p を regular な S ペアと呼ぶ.
2. $mS(g) \succ m'S(g')$ なら mg を, $mS(g) \prec m'S(g')$ なら $m'g'$ を主成分とよぶ.
3. ペア p の主成分 mg に対する $mS(g)$ をペアの signature と呼び, $S(p)$ と書く.
4. $\mathrm{LM}(mg)$ を $\mathrm{LM}(p)$ と書く.

注意 5 (regular 性の判定)

S ペア (g, g') の regular 性は, $S(g)$, $S(g')$ がわからないと判定できない.

signature based algorithm の構成

- regular な S ペアのみを考える.
  regular S ペアの signature の小さいものから順に処理する.
- S 多項式の regular 簡約
  通常の単項簡約に制限がついたもの
  regular S ペア p に対し, $\mathrm{Spoly}(p)$ の簡約に使えるのは $S(p) \succ mS(g)$ を満たす $m \in M$, $g \in G$
- regular 簡約して 0 にならなければ, 剰余の signature が $S(p)$ に決まる.
- signature が等しい S ペアは一つだけ処理すればよい.
  同一 signature のペアのうち, LM が最小のもののみ残せばよい

signature based algorithm

アルゴリズム 6 ($SignatureBasedAlgorithm(F)$)

Input: $F = (f_1, \ldots, f_m)$
Output: $\langle F \rangle$ の $\mathfrak{S}$-GB

1: $S(f_i) \leftarrow e_i$ $(i = 1, \ldots, m)$; $Syz \leftarrow \emptyset$; $G \leftarrow \emptyset$
2: **for** $f \in F$ **do** $Update(f, G, D)$
3: **while** $D \neq \emptyset$ **do**
4: $s \leftarrow \min\{S(p) \mid p \in D\}$; $p \leftarrow S(p) = s$ なる $p \in D$; $D \leftarrow D \setminus \{p\}$
5: **if** s を割り切る Syz の元がない **then**
6: **if** $\mathrm{LM}(p) = \min\{\mathrm{LM}(mg) \mid g \in G, m \in M, mS(g) = s\}$ **then**
7: $r \leftarrow RegularReduce(\mathrm{Spoly}(p), G)$
8: **if** $r \neq 0$ **then**
9: $S(r) \leftarrow s$; $Update(r, G, D)$
10: **else**
11: $Syz \leftarrow Syz \cup \{s\}$
12: **return** G

S ペアの消去

1. syzygy criterion による消去 (5 行目)
   $S(p) \in \mathrm{LM}_{\prec}(\mathrm{syz}(f_1, \ldots, f_m))$ なるペア p は捨ててよい
2. 同一 signature のペアを一つだけ処理 (2, 9 行目)
3. $mS(g) = s$ なる mg 中 LM 最小の元が regular 簡約できない場合に s を捨てる (6 行目)
   $S(p) = s$ なるペアは捨ててよいという意味

注意 7 (消去基準の判定)

いずれも, $S(g)$, $S(p)$ が正確にわかっている必要がある.

$Update(r, G, D)$

同一 signature のペアのうち, LM が最小のもののみ残す.

アルゴリズム 8 ($Update(r, G, D)$)

1: $New \leftarrow G$ と r から作った regular S ペア
2: **for** $p \in New$ **do**
3: **if** $S(q) = S(p)$ なる $q \in D$ が存在する **then**
4: **if** $\mathrm{LM}(p) < \mathrm{LM}(q)$ **then** $D \leftarrow (D \setminus \{q\}) \cup \{p\}$
5: **else** $D \leftarrow D \cup \{p\}$
6: $G \leftarrow G \cup \{r\}$

F_4 スタイル簡約のための準備

- 項順序 $<$, 加群項順序 $\prec$ は次を満たすことを仮定:
  $t, s \in M$ とする.
  1. $\mathrm{tdeg}(t) < \mathrm{tdeg}(s)$ ならば $t < s$
  2. $t < s$ ならば $te_i \prec se_i$ ($<$ と $\prec$ は compatible)
  3. $\mathrm{tdeg}(t\mathrm{LM}(f_i)) < \mathrm{tdeg}(s\mathrm{LM}(f_j))$ ならば $te_i \prec se_j$

  例: $<$ を grevlex, $\prec$ を $<$ 上の Schreyer 順序とする
- 係数体は有限体とする.
  有理数体上では係数膨張など別種の困難が生ずるので.
- 簡単のため入力イデアルは斉次とする.,
  非斉次イデアルは, 生成系を斉次化してアルゴリズムを実行する.

F_4 における簡約

1. ある条件を満たす S ペアを集める.
   通常は, 全次数最小の S ペアを全部取り出す.
2. 取り出した S ペアを全て簡約するのに十分な reducer リストを作る.
   symbolic preprocessing と呼ばれるアルゴリズムで行う.
   簡約に現れる可能性がある単項式のリストも得られる.
3. 単項式のリストに従って, S 多項式, reducer をベクトルに変換する.
4. 得られたベクトルを並べて行列を作り, 簡約する.

F_4 における symbolic preprocessing

$T(p)$: p に現れる単項式全体

アルゴリズム 9 ($SymbolicPreproc(S, G)$)

Input : S ペアの集合 S, 多項式集合 G

Output : $P = \{\mathrm{Spoly}(p) \mid p \in S\}$ の元を G に関する正規形に簡約できる
多項式集合 Red および,
$P \cup Red$ に現れる全ての単項式を並べた列 T

1: $w \leftarrow ()$
2: **for** $p \in S$ **do** $w \leftarrow \mathtt{merge}(w, T(\mathrm{Spoly}(p)))$
3: $Red \leftarrow \emptyset$; $T \leftarrow ()$
4: **while** $w \neq ()$ **do**
5: **if** $\mathrm{LM}(g_i) \mid \mathrm{TOP}(w)$ を満たす g_i が存在する **then**
6: $r \leftarrow \frac{\mathrm{TOP}(w)}{\mathrm{LM}(g_i)} \cdot g_i$; $T \leftarrow \mathtt{merge}(T, T(r))$; $Red \leftarrow Red \cup \{r\}$
7: w から先頭を外して T の末尾に追加する
8: **return** (Red, T)

SBA で F_4 スタイルの行列簡約を行う場合の問題点

- 全次数 $d-1$ 次の基底が全て得られているとする.
- S_d を全次数 d 次の S ペア全体とする.
  仮定により, $d+1$ 次以上の S ペアの signature は全て S_d の元のそれより大きい

1. 逐次的な regular 簡約で行われる剰余計算の保証
   単項式 m が reducer h で簡約できるかどうかは, m が由来する S 多項式の signature に依存する
   $\Rightarrow F_4$ の symbolic preprocessing のままでは不十分
   signature を考慮した symbolic preprocessing を行う

SBA で F_4 スタイルの行列簡約を行う場合の問題点

2. S_d だけで全次数 d 次の基底が尽くされるとは限らない
   regular 簡約の制限により, $\mathrm{LM}(g)$ が他の $\mathrm{LM}(h)$ $(h \in G)$ で割り切れる g が生成される場合がある
   $\Rightarrow$ 新たに d 次の S ペアが生成される場合がある.
3. 新たに得られた基底の signature が確定できない (?)
   生成された基底が $\mathfrak{S}$-既約でない可能性がある (?)
   $\Rightarrow$ この基底から作られる S ペアの signature が不正確になる (?)

$\Rightarrow$ いまのところ, この方法でアルゴリズムの正当性を示す見通しが立たない
$\Rightarrow$ とりあえず、行列でまとめて簡約することは諦める.

折衷案：逐次 SBA において, ベクトルに変換して簡約

- アルゴリズム自体は逐次 SBA
  各ステップで signature が確定し, 各種 S ペア消去基準も正確に働く
- 簡約操作をベクトルに変換して行う
  最小全次数の S ペア全体から, regular 簡約を保証する reducer リストを作り, (疎な) ベクトルに変換しておく
- reducer リストの作り方
  signature を考慮した symbolic preprocessing を行う
- 0 でない剰余はベクトルに変換して reducer リストに追加する
  次以降の簡約に使える

signature を考慮した symbolic preprocessing

symbolic preprocessing 中に単項式 t が現れたとする.

- F_4 の場合
  単に, 単項式 t を割る $\mathrm{LM}(g)$ $(g \in G)$ を見つければよい
- sba の場合
  - $t = m\mathrm{LM}(g)$ だけでなく, t が signature s の S ペアに由来する場合, $s \succ mS(g)$ となる g を見つける必要がある
  - t は複数の S ペアに由来する可能性がある
    $\Rightarrow$ どの S ペアの簡約に t が実際に現れても簡約できるような g を選んでおく必要がある
    やり方によっては, reducer を探す手間が大きくなる可能性がある.

signature を考慮した reducer の見つけ方

全次数 d の S ペア全体を S_d とする.
単項式 t に対する reducer を探す場合

方法 1 $\mathrm{LM}(g) \mid t$ かつ $\frac{t}{\mathrm{LM}(g)} S(g)$ が最小の g を探す
t がどの S 多項式の regular 簡約に現れて, 実際に regular 簡約可能でも, この g で regular 簡約できる.
常に, その時点での G を全てスキャンする必要がある.

方法 2 $s_{min} = \min\{S(p) \mid p \in S_d\}$ とし, $\frac{t}{\mathrm{LM}(g)} S(g) \prec s_{min}$ を満たす g があればそれを使う.
ない場合には $\frac{t}{\mathrm{LM}(g)} S(g)$ 最小のものを使う.
この条件でも regular 簡約が完遂できる.

$\Rightarrow$ 実験で比較

全次数 d の処理における問題点

- F_4 の場合
  全次数 d の S ペア全体から出発して行列化して簡約して得られた新しい基底からは, d 次の S ペアは生じない
  (他の $\mathrm{LM}(g)$ で割れる LM を持つ新基底は存在しないから)
- sba の場合
  regular 簡約の制限により, 他の $\mathrm{LM}(g)$ で割り切れる LM を持つ新基底が生じる可能性がある
  $\Rightarrow$ 新たに d 次の S ペアが生じる可能性がある
  $\Rightarrow$ 新しい S 多項式に, symbolic preprocessing では現れなかった単項式が現れる可能性がある

S_d の処理中に生じた全次数 d 次の S ペア

S_d から symbolic preprocessing で得られた単項式リストを T, reducer リストを Red とする.
$p \notin S_d$ なる S ペアを処理する場合,

- $T(\mathrm{Spoly}(p)) \subset T$ の場合
  $\mathrm{Spoly}(p)$ を regular 簡約するのに十分な reducer がすでにリストにある
- $T(\mathrm{Spoly}(p)) \not\subset T$ の場合
  $\mathrm{Spoly}(p)$ に対する symbolic preprocessing の結果得られた単項式リスト, reducer リストをそれぞれ T, Red にマージする.

先行研究

- F4/5[AP10] (F4-RB [EF17])
  F4 において signature を考慮した symbolic preprocessing を行う.
  F4/5 : termination の保証がない. (正当性も?)
- MatrixF5 [EF17] [V17]
  全次数 d の Macaulay 行列を作って簡約する.
  計算量が求めやすいが実用的ではない.

signature based algorithm with F_4-like reduction

アルゴリズム 10 ($SignatureBasadAlgorithmF_4(F)$)

Input: 斉次多項式集合 $F = (f_1, \ldots, f_m)$
Output: $\langle F \rangle$ の $\mathfrak{S}$-GB

1: $S(f_i) \leftarrow e_i$ $(i = 1, \ldots, m)$; $S \leftarrow \emptyset$
2: **for** $f \in F$ **do** $Update(f, G, D)$
3: **while** $D \neq \emptyset$ **do**
4: $d \leftarrow \min\{\mathrm{tdeg}(p) \mid p \in D\}$; $S_d \leftarrow \{p \mid \mathrm{tdeg}(p) = d\}$; $D \leftarrow D \setminus S_d$
5: $(R, T) \leftarrow SymbolicPreprocSig(S_d, G)$; $VR \leftarrow PtoV(R, T)$
6: **while** $S_d \neq \emptyset$ **do**
7: $s \leftarrow \min\{S(p) \mid p \in S_d\}$; $p \leftarrow S(p) = s$ なる $p \in S_d$; $S_d \leftarrow S_d \setminus \{p\}$
8: **if** s を割り切る S の元がない **then**
9: **if** $\mathrm{LM}(p) = \min\{\mathrm{LM}(mg) \mid g \in G, m \in M, mS(g) = s\}$ **then**
10: **if** $T(\mathrm{Spoly}(p)) \notin T$ **then**
11: $(R_p, T_p) \leftarrow SymbolicPreprocSig(\{p\}, G)$
12: $R \leftarrow \mathtt{merge}(R, R_p)$; $T \leftarrow \mathtt{merge}(T, T_p)$; $VR \leftarrow PtoV(R, T)$
13: $r \leftarrow RegularReduceV(\mathrm{Spoly}(p), VR, T)$; $S(r) \leftarrow s_{min}$
14: **if** $r \neq 0$ **then**
15: $Update(r, G, D)$; $R \leftarrow \mathtt{merge}(R, \{r\})$; $VR \leftarrow \mathtt{merge}(VR, PtoV(\{r\}, T))$
16: $S_d \leftarrow S_d \cup \{p \in D \mid \mathrm{tdeg}(p) = d\}$; $D \leftarrow D \setminus S_d$
17: **else** $S \leftarrow S \cup \{s\}$
18: **return** G

$SymbolicPreprocSig(S, G)$

アルゴリズム 11 ($SymbolicPreprocSig(S, G)$)

$SymbolicPreprocSig(S, G)$

Input : S ペアの集合 S, 多項式集合 G

Output : $P = \{(\mathrm{Spoly}(p), S(p)) \mid p \in S\}$ の元を G に関する regular 簡約で正規形に簡約できる多項式と signature のペア集合 Red および, $P \cup Red$ に現れる全ての単項式を並べた列 T

1: $w \leftarrow ()$
2: **for** $p \in S$ **do** $w \leftarrow \mathtt{merge}(w, T(\mathrm{Spoly}(p)))$
3: $s_{min} \leftarrow \min\{S(p) \mid p \in S\}$
4: $Red \leftarrow \emptyset$; $T \leftarrow ()$
5: **while** $w \neq ()$ **do**
6: $\quad g \leftarrow FindReducerSig(\mathrm{TOP}(w), G, s_{min})$
7: $\quad$ **if** $g \neq 0$ **then**
8: $\quad\quad m \leftarrow \frac{\mathrm{TOP}(w)}{\mathrm{LM}(g)}$; $T \leftarrow \mathtt{merge}(T, T(mg))$; $Red \leftarrow Red \cup \{(mg, mS(g)\}$
9: $\quad w$ から先頭を外して T の末尾に追加する
10: **return** (Red, T)

$FindReducerSig(t, G, s_{min})$

アルゴリズム 12 ($FindReducerSig(t, G, s_{min})$)

Input : 単項式 t, 多項式集合 G, 加群単項式 s_{min}
Output :$\mathrm{LM}(g) \mid t$ かつ $mS(g)$ がなるべく小さい $g \in G$ (ない場合は 0)

1: $s \leftarrow 0$; $g_{min} \leftarrow 0$
2: **for** $g \in G$ **do**
3: **if** $\mathrm{LM}(g) \mid t$ **then**
4: $m \leftarrow \frac{t}{\mathrm{LM}(g)}$
5: **if** $mS(g) < s_{min}$ **then return** g
6: **if** $s = 0$ or $mS(g) < s$ **then**
7: $s = mS(g)$; $g_{min} = g$
8: **return** g_{min}

注意 13

- 5 行目を削除すると, 常に G 全体をスキャンする.
- F4/5 では, t を含む多項式の 最小 signature を求めてはいるが使われていない.

$PtoV(R,T)$, $RegularReduceV(p,R,T)$

アルゴリズム 14 ($PtoV(R,T)$)

Input : 多項式と signature のペアの集合 R, 単項式列 $T = (t_0, \ldots, t_{l-1})$
Output :疎なベクトルに変換した reducer 列 $VR = (VR_0, \ldots, VR_{l-1})$

1: **for** $i = 0$ to $l-1$ **do**
2: 　**if** $\mathrm{LM}(r) = t_i$ なる $(r,s) \in R$ が存在する **then**
3: 　　$r/\mathrm{LC}(r) = \sum_{j=1}^{k} c_j t_{i_j}$ とするとき $v \leftarrow ((i_1,c_1),\ldots,(i_k,c_k))$; $VR_i \leftarrow (v,s)$
4: 　**else** $VR_i \leftarrow (0,0)$
5: **return** VR

アルゴリズム 15 ($RegularReduceV(p,R,T)$)

Input : 多項式 p, reducer 列 $R = (r_0, \ldots, r_{l-1})$, 単項式列 $T = (t_0, \ldots, t_{l-1})$
Output :p を R で regular 簡約した結果

1: $p = c_0 t_0 + \cdots + c_{l-1} t_{l-1}$ のとき $v \leftarrow (c_0, \ldots, c_{l-1})$
2: **for** $i = 0$ to $l-1$ **do**
3: 　$(r_i, s_i) \leftarrow R_i$
4: 　**if** $v_i \neq 0$ and $r_i \neq 0$ and $s_i \prec S(p)$ **then**
5: 　　$v \leftarrow v - v_i \cdot r_i$ (r_i の非零成分に対してのみ更新)
6: **return** $v_0 t_0 + \cdots + v_{l-1} t_{l-1}$

Asir 上での実験

- $\mathbb{F}_{32003}$ 上での計算
- Risa/Asir 20221212 on MacBookPro M1 Max
- 入力がパラメタで生成できる例で比較
- `nd_sba` (SBA), `nd_sba_f4` (SBA+F4 reduction) の比較
  相互簡約抜きの時間 (秒) を示す
  `nd_sba` はデフォルトで full reduction, 表で `(t)` は top reduction (こちらの方が速い場合)
- 参考までに `nd_f4`, maple `Basis` (F4) のデータも示す
  `nd_f4` (有限体上) はとても遅い (simplify を実装していない, 行列計算が最適化されていない etc.)
  `nd_sba_f4` との比較のため やむなく載せる

タイミングデータ1

	nd_sba_f4	nd_sba	nd_f4	Basis
eco(12)	5.5	30	13	1.7
eco(13)	35	280	110	10
eco(14)	270	2700	1000	61
katsura(11)	5.3	39	28	2.7
katsura(12)	33	340	230	16
katsura(13)	210	3100	1800	110
noon(9)	7.5	2.6(t)	31	5.9
noon(10)	73	30(t)	420	56
noon(11)	830	510(t)	6700	720

タイミングデータ 2

`hrandom(n,d,m)` : dense な `n` 変数斉次 `d` 次式 `m` 個で生成されるイデアル

	`nd_sba_f4`	`nd_sba`	`nd_f4`	`Basis`
hrandom(8,4,6)	74	570	230	44
hrandom(8,4,7)	170	2300	840	110
hrandom(8,4,8)	220	3100	940	130
hrandom(9,3,8)	28	280	130	16
hrandom(9,3,9)	26	270	110	13
hrandom(10,3,9)	470	7700	3200	320
hrandom(10,3,10)	460	6500	2400	260

nd_sba_f4 における symbolic preprocessing の比較

	下限あり		常に最小	
	total	symb(find)	total	symb(find)
eco(13)	35	6.6(2.2)	31	8.3(5.6)
eco(14)	270	36(11)	230	47(32)
katsura(12)	33	3.6(0.73)	29	3.7(1.5)
katsura(13)	210	16(3.0)	180	18(7.6)
noon(9)	7.5	3.2(1.8)	15	11(8.7)
noon(10)	73	22(15)	170	120(110)
noon(11)	830	190(140)	2400	1700(1600)

- symb : symbolic preprocessing の計算時間

  (find) は symb のうち, reducer を見つけるのにかかった時間

- 下限あり : S ペアの signature の最小値未満の signature の reducer なら OK とする (p17 方法 2)

- 常に最小 : 常に signature 最小の reducer を探す (p17 方法 1)

計算時間に関する考察

- `nd_sba` と `nd_sba_f4` の比較
  `eco(n)`, `katsura(n)`, `hrandom(n,d,m)` は 10 倍程度高速化
  `noon(n)` はかえって遅くなった (top reduction にすると更に遅くなる. 理由は不明.)
- symbolic preprocessing における reducer の見つけ方
  下限なしにすると, 探す時間は増えるが全体の計算時間が短くなる場合がある
  $\Rightarrow$ 「よい」reducer (項数が少ないとか) が見つかっている?
  しかし, `noon(n)` では大きく効率を落としている (理由は不明)
  $\Rightarrow$ 下限ありの方が安全?

その他

- S_d の処理中に新たに現れる d 次の S ペア
  数多くの例で処理中の全次数の S ペアが現れていることがわかった.
  $\Rightarrow S_d$ だけ作った行列の簡約だけでこれらと同等のものが全て得られるかは不明
- 単項式リストの更新の必要性
  S_d の処理中に, symbolic preprocessing で得られた単項式リストにない単項式を持つ d 次の S 多項式が現れる例が存在する.
  今のところ, `filter9` という例のみだが, 実際に単項式リストを更新する必要があることがわかった

まとめ

- F_4 スタイルの簡約の効果
  逐次 SBA の簡約をベクトルで行うだけ
  ある種の問題に対しては大きな効果がある.
- 通常の F_4 では起きないことが起きる
  - d 次の簡約から d 次の元が生じる
  - 単項式リストの更新が必要になる

  $\Rightarrow S_d$ 全体を行列にして簡約することが (今の所) 困難
- 今後の課題
  - 有理数体係数の場合
  - 非斉次の場合

文献

F02 A new efficient algorithm for computing Gröbner bases without reduction to zero (F_5), in Proc. ISSAC2002, 75-83, 2002.

AP10 M. Albrecht, J. Perry, F4/5 (2010). https://arxiv.org/abs/1006.4933v2.

AP11 A. Arri, J. Perry, The F5 criterion revised, J. Symb. Comp., 46, 1017-1029 (2011). (A revised version : https://arxiv.org/abs/1012.3664v6)

EF17 C. Eder, J.-C. Faugère, A survey on signature-based algorithms for comptuing Gröbner bases, J. Symb. Comp., 80, 719-784, 2017.

V17 T. Vaccon, Matrix-F5 algorithms over finite-precision complete discrete valuation fields, J. Symb. Comp., 80, 329-350, 2017.