FAUGÈRE'S F_5 ALGORITHM

JOHN PERRY

ABSTRACT. This report gives pseudocode for an implementation the F_5 algorithm, noting errors found both in Faugère's seminal 2002 paper and Stegers' 2006 undergraduate thesis. It also outlines a relatively easy optimization that, in our experience, produces a substiantial improvement to the algorithm's efficiency.

1. Introduction

The F_5 algorithm has twice appeared in print [2, 6], both times with errors that are not easy to detect and debug. In this note we present new pseudocode written during the creation of two new, open-source implementations [5, 1]. We do not review the theory underlying the algorithm; aside from the references already noted, the interested reader should review [3, 4].

In Section 2 we preview the notation and a few terms. In Section 4 we comment on the Maple implementation. In Section 3 we present a relatively easy optimization that provides a substantial improvement in performance, despite one researcher's having advised against it. In Section 5 we comment on the pseudocode, which begins on page 5. We note, as appropriate, the places where our pseudocode diverges from that of Faugère or Stegers (or both!).

If the reader should find an error, please notify the author immediately so that he can correct it.

2. Notation

We write $\mathcal{R} = \mathbb{F}[x_1, x_2, \ldots, x_n]$ for a polynomial ring, and $<_T$ for an admissible ordering on the power products of $\mathcal{R}$. For the leading term of a polynomial $f \in \mathcal{R}$ we write $\mathrm{lt}_{<_T}(f)$; for the leading coefficient we write $\mathrm{lc}_{<_T}(f)$. We do not include the coefficient in the leading term.

Essential to F_5 is the notion of a **labeled polynomial**. We denote this as a tuple of a signature (μ, ν) and a polynomial p. (Faugère uses $\mu\mathbf{F}_\nu$ to indicate the signature; Stegers, $(\mu, \mathbf{e}_\nu)$.) The signature should always satisfy for some $h_\nu, h_{\nu+1}, \ldots, h_m \in \mathcal{R}$

$$p = h_\nu f_\nu + h_{\nu+1} f_{\nu+1} + \cdots + h_m f_m \qquad \text{and} \qquad h_\nu \neq 0 \qquad \text{and} \qquad \mu = \mathrm{lt}_{<_T}(h_\nu). \tag{1}$$

The fact that $h_\nu \neq 0$ is crucial. A labeled polynomial that satisfies (1) is called **admissible**, but F_5 is careful not to generate inadmissible labeled polynomials. (See for example the precautions taken in Algorithm Top_Reduction to assign the correct signature to a top-reduced polynomial.) Except for the algorithm's inputs, $\mathrm{lt}_{<_T}(h_\nu) \cdot \mathrm{lt}_{<_T}(f_\nu) >_T \mathrm{lt}_{<_T}(p)$.

We use L to denote a list of labeled polynomials. The functions $\mathrm{Sig}(k)$ and $\mathrm{Poly}(k)$ return the signature and the polynomial, respectively, of the labeled polynomial L_k. The signature ordering $\prec$ is defined exactly as Faugère defines it: $(\mu_k, \nu_k) \prec (\mu_\ell, \nu_\ell)$ if and only if

Key words and phrases. Gröbner bases, F5.

- $\nu_k > \nu_\ell$, or
- $\nu_k = \nu_\ell$ and $\mu_k <_T \mu_\ell$.

As do Faugère and Stegers, given a power product u and a signature (μ, ν) we define $u \cdot (\mu, \nu) = (u \cdot \mu, \nu)$.

We also use B to denote the reduced Gröbner basis of a previous system; see Section 3 for some explanation.

3. Two easy optimizations

Stegers writes on page 41 that,

> We experienced a speed-up by computing the (unique) *reduced* Gröbner basis G_i^{red}... after completing step i, and subsequently checking if polynomials are top-reducible with respect to G_i^{red} instead of $\widehat{G_i}$. It is not suggested to *replace* the polynomials in $\widehat{G_i}$ completely, as this would almost certainly result in inadmissible polynomials.

The last sentence is correct, but we have found a workaround. After computing the reduced basis G_i^{red}, one need only set up the appropriate records in the labeled polynomails L and the rewrite rules W, with (a) new, admissible signatures of the reduced basis, and (b) the reductions to zero of the reduced basis' S-polynomials. This is quite easy to implement; the reader will find our solution in Algorithm Basis, where our notation for the reduced Gröbner basis is B. We have observed a 30% improvement on problems like Cyclic-n.

Another optimization, with less drastic results, is to sort the input to Algorithm S by increasing *signature* rather than by increasing *lcm*. We have found that this decreases the number of polynomials computed in an unoptimized F5.

4. Notes on the Maple implementation

The Maple implementation is quite slow. Stegers reports timings of seven seconds for Cyclic 6 on an Athlon XP 2500 with 512 MB RAM running MAGMA 2.11-14. On our machine, a 2.8 GHz Intel Core 2 Duo with 2 GB RAM, the same system requires *nearly fifteen minutes* on Maple 11. Other systems produce similar differences. Is Maple really one hundred twenty times slower than MAGMA? Our implementation is very close to that of Stegers', so we doubt the slowdown lies in the actual code. In most cases, the results are the same: we compute the same number of S-polynomials, and obtain the same number of zero reductions. However, certain systems appear to have a few more S-polynomials and zero reductions than Stegers, so there may be a minor issue or two.[1]

The subalgorithms require several global variables; the Maple code implements this using variables local to the module containing all the procedures. The user can read the value of these variables using the `Report_...()` functions.

The implementation also relies on a few short procedures designed to assist with certain tasks. These include:

`Top_Reducible` to determine if a monomial is top-reducible by the previously-computed Gröbner basis;

[1]We doubt it. More recently, we implemented F5 as a Singular library. The result is substantially faster than Maple, with Cyclic 6 completing in 160 seconds on a 1.33 GHz G4 with 1 GB RAM in the unoptimized implementation. The optimized implementation completes in 114 seconds. This is still slower than MAGMA (assuming that the timing mechanisms measure the same things).

`First_Sig_Min`	to test whether signature of the one labeled polynomial is smaller than the signature of a second;
`Sig_Comp`	to use `First_Sig_Min` in a sorter;
`Sig_Prod`	to multiply a monomial to a signature;
`Add_Labeled_Poly`	to add a labeled polynomial to the list of labeled polynomials.

We have omitted the details of these subalgorithms, which in any case are trivial. We also use Maple's `Groebner` package for a number of utilities.

5. Notes on the pseudocode

We have indicated where our pseudocode diverges from that of Faugère or Stegers. As far as we can tell, the differences in Faugère and Stegers are for the most part fatal errors, with two exceptions. In Algorithm Crit_Pair, neither tests for equal signatures in the construction of the S-polynomial, although this is a sign of a pair that is not normalized. Further, Stegers sometimes tests for normalized polynomials using a smaller set. This may be more correct given the actual criterion, but in following Faugère's choice to use the previous Gröbner basis *only* we have found no cases where the result is any different. We plan to investigate this further.

Global variables. The algorithm relies on four global variables:

L	a list of labeled polynomials
B	a reduced Gröbner basis, computed previously
W	an array of lists of rewrite rules
$<_T$	an admissible ordering

Each subalgorithm specifies which global variables are necessary. In a language that does not use global variables (Stegers states that MAGMA is such a language), one could pass these variables as inputs.

In the Maple implementation, L is actually a Maple table, to avoid difficulties inherent to Maple lists (inefficiency, size restraints, etc.). This requires us to keep track of how many elements are in the list, and we have set aside a variable for that. The same is true for the lists of rules in W.

Incremental nature of the algorithm. To maximize the effect of Faugère's Criterion, the F_5 algorithm takes an incremental approach, computing Gröbner bases of $\{f_1\}$, $\{f_1, f_2\}$, $\ldots$, $\{f_1, f_2, \ldots, f_m\}$. Algorithm Basis controls this loop; on each iteration, Algorithm PartialBasis computes a Gröbner Basis G_{curr} of $\{f_1, f_2, \ldots, f_i\}$, using as its starting point a basis G_{prev} for $\{f_1, f_2, \ldots, f_{i-1}\}$.

Faugère uses the title "incremental F_5" for our Algorithm Basis; Stegers titles it "Algorithm 3: F5 – Main loop." Faugère uses the title "F_5" for our Algorithm PartialBasis; Stegers calls it "Algorithm 4: F5 – Core routine".

Construction and reduction of S-polynomials. Algorithms Crit_Pair and SPols generate the critical pairs and construct the S-polynomials. Each employs a criterion that discards certain useless critical pairs; Crit_Pair uses Faugère's criterion, and SPols uses a criterion developed from [4].

Algorithms Reduction, Top_Reduction, and Find_Reductor:

(1) reduce an S-polynomial with respect to the previous Gröbner basis, and
(2) top-reduce the reduced form with respect to the current set of generators.

Faugère's name for the last of these three is "IsReducible"; we have followed Stegers' convention.

Record-keeping. One of the major strengths of F_5 is how it keeps records of syzygies that have already been computed; the "rewrite rules" keep track of this. Faugère gives Algorithm Is_Rewritable the name "Rewritten?", and Algorithm Find_Rewriting the name "Rewritten". Stegers named the first "Rewritable" and the second "Rewrite". We have followed Stegers for the first, but prefer our name for the second.

6. Acknowledgments

The author would like to thank Christian Eder, Jean-Charles Faugère, Till Stegers, Alexander Semyonov, and Alexei Zobnin for discussions on this or related topics. He would also like to thank the Center for Computer Algebra at Universität Kaiserslautern for allowing him to use their facilities during March of 2008.

Algorithm 1. *Basis.*
globals L, W, $<_T$
inputs
$F = (f_1, f_2, \ldots, f_m) \in \mathcal{R}^m$
$<$, *an admissible ordering*
outputs *a Gröbner basis of* F *with respect to* $<$
do
$<_T := <$
Sort F *by increasing total degree, breaking ties by increasing leading term.*
— *Initialize the record keeping.*
$W := \text{Array}\,(1 \ldots 2)$
$W_1 := \text{List}\,()$
$W_2 := \text{List}\,()$
$L := \text{List}\,(1 \ldots 2)$
$L_1 := \left((1,1), f_1 \cdot (\text{lc}_{<_T}(f_1))^{-1}\right)$
$G := \text{Array}\,(1 \ldots m)$
— *Compute the bases of* (f_1), (f_1, f_2), $\ldots$, $(f_1, f_2, \ldots, f_m)$.
$G_{\text{prev}} = \{1\}$
$B = \{f_1\}$
$ctr := 2$
while $ctr \leq m$
$G_{\text{curr}} := \text{PartialBasis}\,(\#B + 1, B, G_{\text{prev}})$
if $\exists i \in G_{\text{curr}}$ *such that* $\text{Poly}\,(i) = 1$
return $\{1\}$
Let B *be the reduced Gröbner basis of* $(\text{Poly}\,(j))_{j \in G_{\text{curr}}}$
— *Set up records for* B
if $ctr \neq 1$
$L := \text{List}\,(1 \ldots \#B + 1)$
$W := \text{Array}\,(1 \ldots \#B + 1)$
$G_{\text{curr}} := \{\}$;
for $i := 1$ ***to*** $\#B$
$L_i := ((1, i), B_i)$
$G_{\text{curr}} := G_{\text{curr}} \cup \{i\}$
$W_{ctr} := \text{List}\,()$
$t := \text{lt}_{<_T}(B_i)$
— *All the S-polynomials of* B *reduce to zero; document this*
for $j := i + 1$ ***to*** $\#B$
$u := \text{lcm}\,(t, \text{lt}_{<_T}(B_j))\,/\text{lt}_{<_T}(B_j)$
Append $((j, u), 0)$ *to* L
$\text{Add\_Rule}\,()$
$ctr := ctr + 1$
$W_{ctr} := \text{List}\,()$
$L_{ctr} := \left((1, ctr), f_{ctr} \cdot (\text{lc}_{<_T}(f_{ctr}))^{-1}\right)$
return B

Algorithm 2. *Partial_Basis.*
globals L, $<_T$
inputs
$i \in \mathbb{N}$
B, *a reduced Gröbner basis of* $(f_1, f_2, \ldots, f_{i-1})$ *with respect to* $<_T$
$G_{\text{prev}} \in \mathbb{N}^{\#G_{\text{prev}}}$, *indices in* L *of* B
outputs G_{curr}, *indices in* L *of a Gröbner basis of* $(f_1, f_2, \ldots, f_i)$ *with respect to* $<_T$
do
$G_{\text{curr}} := G_{\text{prev}} \cup \{i\}$
$P := \bigcup_{j \in G_{\text{prev}}} \text{Crit\_Pair}\,(i, j, G_{\text{prev}})$
while $P \neq \emptyset$
$d := \min\{\deg t : \{t, k, u, \ell, v\} \in P\}$
$P_d := \{\{t, k, u, \ell, v\} \in P : d = \deg t\}$
$P := P \backslash P_d$
$S := \text{SPols}\,(P_d)$
$R := \text{Reduction}\,(S, B, G_{\text{prev}}, G_{\text{curr}})$
for $k \in R$
$P := P \cup \left(\bigcup_{j \in G_{\text{curr}}} \text{Crit\_Pair}\,(j, k, G_{\text{prev}})\right)$
$G_{\text{curr}} := G_{\text{curr}} \cup \{k\}$
return G_{curr}

Algorithm 3. *Crit_Pair.*
globals $<_T$
inputs
$k, \ell \in \mathbb{N}$ *such that* $i \leq k < \ell \leq \#L$
$G_{\text{prev}} \in \mathbb{N}^{\#G_{\text{prev}}}$, *indices in* L *of a Gröbner basis of* $(f_1, f_2, \ldots, f_{i-1})$ *w/respect to* $<_T$
outputs $\{(t, u, k, v, \ell)\}$, *corresponding to a critical pair* $\{k, l\}$ *necessary for the computation of a Gröbner basis of* $(f_1, f_2, \ldots, f_i)$; $\emptyset$ *otherwise*
do
$t_k := \mathrm{lt}_{<_T}(\mathrm{Poly}(k))$
$t_\ell := \mathrm{lt}_{<_T}(\mathrm{Poly}(\ell))$
$t := \mathrm{lcm}(t_k, t_\ell)$
$u_1 := t/t_k$
$u_2 := t/t_\ell$
— *Neither Faugère nor Stegers notes the test below, useful in non-regular systems.*
if $u_1 \cdot \mathrm{Sig}(k) = u_2 \cdot \mathrm{Sig}(\ell)$
return $\emptyset$
$(\mu_1, \nu_1) := \mathrm{Sig}(k)$
$(\mu_2, \nu_2) := \mathrm{Sig}(\ell)$
if $\nu_1 = i$ ***and*** $u_1 \cdot \mu_1$ *is top-reducible by* G_{prev} — *Stegers writes* G_{ν_1}
return $\emptyset$
if $\nu_2 = i$ ***and*** $u_2 \cdot \mu_2$ *is top-reducible by* G_{prev} — *Stegers writes* G_{ν_2}
— *Another minor optimization is to check* Is_Rewritable *here*
return $\emptyset$
if $u_1 \cdot \mathrm{Sig}(k) \prec u_2 \cdot \mathrm{Sig}(\ell)$ — *Faugère's writeup compares* $\mathrm{Sig}(k) \prec \mathrm{Sig}(\ell)$.
Swap u_1 *with* u_2 *and* k *with* ℓ
return $\{(t, u_1, k, u_2, \ell)\}$

Algorithm 4. *Spols.*
globals L, $<_T$
inputs
    P, *a list of critical pairs*
outputs S, *a list of indices in* L *of* S*-polynomials computed for a Gröbner basis of* $(f_1, f_2, \ldots, f_i)$
do
    $S := ()$
    — *Faugère and Stegers do not say to sort* P*, but performance suffers if not.*
    — *For the example in Faugère's paper, 8 polynomials would be computed, not 7.*
    Sort P *from smallest to largest lcm*
    for $(t, k, u, \ell, v) \in P$
        if not Is_Rewritable (u, k) ***and not*** Is_Rewritable (v, ℓ)
            Compute s*, the* S*-polynomial of* Poly (k) *and* Poly (ℓ)
            if $s \neq 0$
                — *Stegers writes* Sig (ℓ).
                Append $(u \cdot \mathrm{Sig}\,(k), s)$ *to* L
                Add_Rule $()$
                Append $\#L$ *to* S
    Sort S *by increasing signature*
    return S

Algorithm 5. *Reduction.*
globals L
inputs
S, *a list of indices of polynomials added to the generators* G_i
B, *a reduced Gröbner basis of* $(f_1, f_2, \ldots, f_{i-1})$ *with respect to* $<_T$
$G_{\text{prev}} \in \mathbb{N}^{\#G_{\text{prev}}}$, *indices in* L *corresponding to* B
$G_{\text{curr}} \in \mathbb{N}^{\#G_{\text{curr}}}$, *indices in* L *of a list of generators of the ideal of* $(f_1, f_2, \ldots, f_i)$
outputs *completed, a subset of* G *corresponding to reduced polynomials*
do
$to\_do := S$
$completed := \emptyset$
— *Our implementation optimizes out the next two lines*
— *Stegers writes* G_{curr} *(his notation:* G'*).*
— $reducers := \{\text{Poly}(ctr) : ctr \in G_{\text{prev}}\}$
while $to\_do \neq ()$
Let k *be the element of* to_do *such that* $\text{Sig}(k)$ *is minimal.*
$to\_do := to\_do \setminus \{k\}$
— *Unoptimized (Faugère, Stegers):*
— $h := \text{Normal\_Form}(\text{Poly}(k), reducers, <_T)$
$h := \text{Normal\_Form}(\text{Poly}(k), B, <_T)$
$L_k := (\text{Sig}(k), h)$
$newly\_completed, redo := \text{Top\_Reduction}(k, G_{\text{prev}}, G_{\text{curr}} \cup completed)$
$completed := completed \cup newly\_completed$
— *Faugere and Stegers both write* $to\_do := to\_do \cup redo$,
— *but* to_do *is not a set, and for reasons of efficiency needs to be sorted.*
for $j \in redo$
Insert j *in* to_do *, sorting by increasing signature*
return *completed*

Algorithm 6. *Top_ Reduction.*
globals L, $<_T$
inputs
k, *the index of a labeled polynomial*
$G_{\text{prev}} \in \mathbb{N}^{\#G_{\text{prev}}}$, *indices in* L *of a Gröbner basis of* $(f_1, f_2, \ldots, f_{i-1})$ *w/respect to* $<_T$
$G_{\text{curr}} \in \mathbb{N}^{\#G_{\text{curr}}}$, *indices in* L *of a list of generators of the ideal of* $(f_1, f_2, \ldots, f_i)$
outputs
completed, which has value $\{k\}$ *if* L_k *was* ***not*** *top-reduced and* $\emptyset$ *otherwise*
to_ do, which has value $\emptyset$ *if* L_k *was not top-reduced,*
$\{k\}$ *if top-reduction generates a labeled polynomial with the same signature as* L_k, *and*
$\{k, \#L\}$ *if top-reduction generates a labeled polynomial with a new signature. (note the generation of a new labeled polynomial in the latter case!)*
do
if $\text{Poly}(k) = 0$ — *This will not happen in a regular sequence.*
warn *"Reduction to zero!"*
return $\emptyset, \emptyset$
$p := \text{Poly}(k)$
$J := \text{Find\_Reductor}(k, G_{\text{prev}}, G_{\text{curr}})$
if $J = \emptyset$
if $p \neq 0$
$L_k := \left(\text{Sig}(k), p \cdot (\text{lc}_{<_T}(p))^{-1}\right)$
return $\{k\}, \emptyset$
— $J \neq \emptyset$, *so top-reduce.*
Let j *be the single element in* J
$q := \text{Poly}(j)$
$u := \frac{\text{lt}_{<_T}(p)}{\text{lt}_{<_T}(q)}$
$c := \text{lc}_{<_T}(p) \cdot (\text{lc}_{<_T}(q))^{-1}$
$p := p - c \cdot u \cdot q$
if $p \neq 0$
$p := p \cdot (\text{lc}_{<_T}(p))^{-1}$
if $u \cdot \text{Sig}(j) \prec \text{Sig}(k)$
$L_k := (\text{Sig}(k), p)$
return $\emptyset, \{k\}$
else
Append $(u \cdot \text{Sig}(j), p)$ *to* L
Add_ Rule()
— *Faugère writes* $\emptyset, \{k, j\}$ *below.*
return $\emptyset, \{k, \#L\}$

Algorithm 7. *Find_ Reductor.*
globals $<_T$
inputs
k, *the index of a labeled polynomial*
$G_{\text{prev}} \in \mathbb{N}^{\#G_{\text{prev}}}$, *indices in* L *of a Gröbner basis with respect to* $<_T$ *of* $(f_1, f_2, \ldots, f_{i-1})$
$G_{\text{curr}} \in \mathbb{N}^{\#G_{\text{curr}}}$, *indices in* L *of a list of generators of the ideal of* $(f_1, f_2, \ldots, f_i)$
outputs
J, *where* $J = \{j\}$ *if* $j \in G_{\text{curr}}$ *and* $\text{Poly}(k)$ *is* safely *top-reducible by* $\text{Poly}(j)$;
otherwise $J = \emptyset$
do
$t := \text{lt}_{<_T}(\text{Poly}(k))$
for $j \in G_{\text{curr}}$
$t' = \text{lt}_{<_T}(\text{Poly}(j))$
if $t' \mid t$
$u := t/t'$
$(\mu_j, \nu_j) := \text{Sig}(j)$
if $u \cdot \text{Sig}(j) \neq \text{Sig}(k)$ ***and not*** Is_Rewritable(u, j)
and $u \cdot \mu_j$ *is not top-reducible by* G_{prev}
return $\{j\}$
return $\emptyset$

Algorithm 8. *Add_ Rule.*
globals L, W
do
$k := \#L$
$(\mu, \nu) := \text{Sig}(k)$
Append (μ, k) *to* W_ν
return

Algorithm 9. *Is_ Rewritable.*
inputs
u, *a power product*
k, *the index of a labeled polynomial in* L
outputs *true if* $u \cdot \text{Sig}(k)$ *is rewritable by another labeled polynomial*
(see Find_ Rewriting)
do
$j :=$ Find_Rewriting(u, k)
return $j \neq k$

Algorithm 10. *Find_Rewriting.*
globals W
inputs
u, *a power product*
k, *the index of a labeled polynomial in* L
outputs
j, *the index of a labeled polynomial in* L *such that if* $(\mu_j, \nu_j) = \mathrm{Sig}(j)$
and $(\mu_k, \nu_k) = \mathrm{Sig}(k)$, *then* $\nu_j = \nu_k$ *and* $\mu_j \mid u \cdot \mu_k$
and L_j *was added to* W_{ν_k} *more recently than* L_k.
do
$(\mu_k, \nu) := \mathrm{Sig}(k)$
$ctr := \#W_\nu$
while $ctr > 0$
$(\mu_j, j) := W_{\nu,ctr}$
if $\mu_j \mid u \cdot \mu_k$
return j
$ctr := ctr - 1$
return k

References

[1] Christian Eder and John Perry. Faugère's F5 algorithm. Singular library, 2008. Available online at `http://www.math.usm.edu/perry/Research/f5_library.lib`.

[2] Jean-Charles Faugère. A new efficient algorithm for computing Gröbner bases without reduction to zero F5. In *International Symposium on Symbolic and Algebraic Computation Symposium - ISSAC 2002, Villeneuve d'Ascq, France*, pages 75–82, Jul 2002.

[3] Daniel Lazard. Gröbner bases, Gaussian elimination, and and resolution of systems of algebraic equations. In J. A. van Hulzen, editor, *EUROCAL '83, European Computer Algebra Conference*, volume 162, pages 146–156. Springer LNCS, 1983.

[4] Hans Möller, Ferdinando Mora, and Carlo Traverso. Gröbner bases computation using syzygies. In P. S. Wang, editor, *Proceedings of the 1992 International Symposium on Symbolic and Algebraic Computation*, pages 320–328. Association for Computing Machinery, ACM Press, July 1992.

[5] John Perry. Faugère's F5 algorithm. Maple worksheet, 2007. Available online at `http://www.math.usm.edu/perry/Research/F5_module.mw`.

[6] Till Stegers. *Faugère's F_5 Algorithm Revisited*. Diplom. thesis, Technische Universität Darmstadt, Germany, 2006.

Department of Mathematics, Box 5045, University of Southern Mississippi, Hattiesburg, MS 39406.

E-mail address: `john.perry@usm.edu`

URL: `http://www.math.usm.edu/perry/`