DEPARTMENT OF MATHEMATICS

TECHNICAL REPORT

GRÖBNER BASIS ALGORITHMS FOR GRASSMAN ALGEBRAS IN A MAPLE PACKAGE

MR. TROY BRACHEY

T nn T n r

OCTOBER 2008

No. 2008 - 1

TENNESSEE TECHNOLOGICAL UNIVERSITY
Cookeville, TN 38505

Gröbner Basis Algorithms for Grassmann Algebras in a Maple Package

Troy Brachey

September 30, 2008

Abstract

A brief discussion of an approach to calculating Gröbner bases in the Grassmann (exterior) algebra with emphasis on the ideal membership problem is presented. A new package written for computing Gröbner bases in Grassmann algebras for Maple 11 is introduced.

1 Introduction

This paper will give details of the algorithms used by the author for the Maple package `TNB`. The `TNB` package is capable of calculating Gröbner bases in the Grassmann algebra. Very little space is devoted to background information and more to the workings of the algorithms themselves.

Two separate types of Gröbner bases will be developed. However, only one will be capable of determining left ideal membership. The steps taken to arrive at each type of Gröbner basis will be similar to the steps taken to compute a *(ii)* $\alpha_i \alpha_j = \alpha_j \alpha_i$, $1 \leq i, j \leq m$

(iii) $e_i \alpha_j = \alpha_j e_i$, $1 \leq i \leq n, 1 \leq j \leq m$

Note that as a consequence of (i) above $e_i^2 = 0$. Thus there exist nonzero zero divisors. Also observe that $\bigwedge_{n,0}$ is a copy of the Grassmann algebra of order n over the field k. Note also that $\bigwedge_{0,m} = k[x_1, x_2, \ldots, x_m]$, is the ring of commutative polynomials in m variables over the field k. Thus it is not a big leap to see that the commutative techniques should produce similar results on the super Grassmann algebra as they do on $k[x_1, \ldots, x_n]$ since our commutative ring is embedded in the algebra. For the remainder of this paper, let $m = 0$, and so we will be working within the Grassmann algebra.

The quality of being Noetherian is a very important concept in the computation of commutative Gröbner bases as it guarantees termination of the division

- Deg[InvLex] order gives [$e_{123}, e_{23}, e_{13}, e_{12}, e_3, e_2, e_1, 1$]
- InvDeg[Lex] order gives [$1, e_1, e_2, e_3, e_{12}, e_{13}, e_{23}, e_{123}$]

Note that RevLex, InvRevLex, and InvDeg[Lex] orders are not admissible. The admissibility of the monomial order is important when redhTd[(e)]TJd; e

Algorithm 1 [4] Left Normal Form LeftNF($f|G$)

```
h = f
while h = 0 and G_h = {g   G | LM(g)|LM(h)} =   do
    choose g   G_h
    h = Spoly(h, g)
end while
return h
```

Theorem 1 (Characterization Theorem for GLBs). *[7] The following conditions are equivalent.*

(i) F *is a GLB.*

(ii) *If* $f_1, f_2 \quad F$*, and* $t \quad T_n$ *satisfies* $t \cdot \mathrm{lcm}(\mathrm{LMon}(f_1), \mathrm{LMon}(f_2)) = 0$*, then* $\mathrm{LeftNF}(t \cdot \mathrm{LeftSpoly}(f_1, f_2)), F) = 0$*.*

Based on the characterizations in Theorem 1, Algorithm 2 will compute a GLB for a given list of Grassmann polynomials. Note that lines 6 and 7 of Algorithm 2 ensure that the "if" condition in (ii) of Theorem 1 is met while lines 8 through 17 perform the "then" statement of the Theorem.

Algorithm 2 [7] Gröbner left basis in $\bigwedge_n$

```
1:  G := F, B = {{f_1, f_2} | f_1, f_2   G; f_1 = f_2}
2:  while B =   do
3:      {f_1, f_2} := an element of B
4:      B := B − {f_1, f_2}
5:      g := LeftSpoly(f_1, f_2)
6:      V := A − V(lcm(LMon(f_1), LMon(f_2)))
7:      T := T(V)   {1}
8:      while T =   do
9:          t := an element of T
10:         T := T − {t}
11:         h := t · g
12:         h  := N(G, h)
13:         if h  = 0 then
14:             B := B   {{g, h } | g   G}
15:             G := G   {h }
16:         end if
17:     end while
18: end while
19: return G
```

De1026Tf21.535t177[]149922Td[(13)-1(t343.711il)-1(e)]T17Td[]1243(oa[]()-222(f)]Tld1492Td[(13)-1(3

Theorem 2 (Characterization Theorem for GLIBs). *[7] The following statements are equivalent.*

(i) F *is a GLIB.*

(ii) *If* f_1, f_2 F*, then for any* t*,* $\mathrm{LeftNF}(t \cdot f_1, F) = 0$ *and* $\mathrm{LeftNF}(t \cdot \mathrm{LeftSpoly}(f_1, f_2$

3 The TNB Package

The `TNB` package is a package of procedures written for Maple 11 that can compute Gröbner left bases and Gröbner left ideal bases in a Grassmann algebra. When the package is loaded, 20 functions are exported.

```
> with(TNB);
```

```
TNB - package for computing Groebner bases in Grassmann algebras
TNB version delta 0.3 (March 12, 2008)
20 functions exported
```

[*Deg*, *InvDeg*, *InvLex*, *InvRevLex*, *LCoe* , *LMon*, *LTerm*, *Lex*, *NF*, *RevLex*, *TLSpolyG*, *TglbG*, *TglibG*, *TlcmG*, *TmingbG*, *idealpoly*, *isadmissible*, *makelist*, *pairs*, *testing*]

Several procedures are from the package `GfG` [2]. These procedures are the monomial orderings and procedures that use the monomial orderings to find the leading term, monomial, and coe□cients. The procedures are *Deg*, *InvDeg*, *InvLex*, *InvRevLex*, *LCoe* , *LMon* a space of dimension n. The procedure does not return a zero polynomial. If one of its randomly generated polynomials happens to be zero, it recursively calls itself until it returns exactly N nonzero such polynomials. This procedure is useful for testing purposes.

9. Procedure *idealpoly* creates a random polynomial (Clifford or Grassmann, depending on the second argument) in an one-sided ideal J - left or right, depending on the third argument- and generated by polynomials supplied as the first list F. The second argument is expected to be wedge - for the wedge product in a Grassmann algebra, or cmul - for the Clifford product in a Clifford algebra. In either case, the algebra is considered over a space of dimension equal to the largest index found in the list F. An output of this procedure contains three items, in the following order:

   (a) A list of random coefficients.

   (b) A list of generators (it is a permuted list F due to removal of duplicates in the first line of the procedure when Maple may and usually does permute elements of F).

   (c) Random polynomial. Assigning a name to this procedure will result in the random polynomial being assigned to that name.

   This procedure is useful for testing purposes.

10. Procedure *testing* takes a list of polynomials F and for each $t \quad T(V)$ and computes the product $t \quad f_i$ for $i = 1, \ldots, n$ where n is the number of polynomials in the given list F then finds the LeftNF$(t \quad f_i|F)$. This is based on Theorem 2. It is useful in determining whether a given Gröbner basis is a GLIB.

Package TNB also relies upon the CLIFFORD package [1]. The CLIFFORD package provides the basis monomials and exterior, or wedge, product which is used for multiplication of Grassmann monomials. Type checking of arguments passed to TNB procedures is also provided by the CLIFFORD package. When loading, the TNB package automatically loads the CLIFFORD package and the BIGEBRA package which is part of the CLIFFORD library.

4 Applications and Examples

This section will show examples of the TNB package in use. In many examples the results are compared to those from

Example 2. [7] Given $f_1 = e_{56} - e_{13}$ and $f_2 = e_{45} - e_{23}$ procedures TglibG and

$$p4 := e123$$

$$P := [Id + 2\,e1 + 3\,e3 + e13,\ e1 + 3\,e23,\ e2 + e1,\ e123]$$

First a GLIB will be computed by TNB. This Gröbner basis is capable of determining ideal membership.

```
> Tgb:=TglibG(P,Deg[Lex]);
```

```
Computed 21 S-polynomials among 7 polynomials in Groebner basis and
needed to compute 21

 This procedure took 3.495 seconds to run.
```

$$Tgb := [e123,\ Id + 2\,e1 + 3\,e3 + e13,\ \tfrac{1}{3}e1 + e23,\ e2 + e1,\ e2,\ e3,\ Id]$$

$$P := [e1 + e3 + e13, e1 + e23, e2 + e1, e123]$$

```
> GBt:=TglibG(P,Deg[Lex]);
```

```
Computed 15 S-polynomials among 6 polynomials in Groebner basis and
needed to compute 15

 This procedure took 3.494 seconds to run.
```

$$GBt := [e123, e1 + e3 + e13, e1 + e23, e2 + e1, -e3 + e2, e3]$$

```
> GBtm:=TmingbG(GBt,Deg[Lex]);
```

$$GBtm := [e2 + e1, -e3 + e2, e3]$$

The minimal Gröbner basis *GBtm* is capable of determining ideal membership. Now `Plural` will compute a reduced Gröbner basis.

```
> Pgb:=PLURALforGlink(P,0,dp,[e1,e2,e3],input_for_Singular,
  input_for_Maple,'infty','d');
```

$$Pgb := [e3, e2, e1]$$

```
> GBtm<>Pgb;
```

$$[e2 + e1, -e3 + e2, e3] = [e3, e2, e1]$$

Note that the lists `GBtm` and `Pgb` are not identical. The di□erence is due to the fact that Gröbner basis returned by `TNB` is a minimal Gröbner basis and the one returned by `Plural` is a reduced Gröbner basis, which is a concept that is also applicable in the non-commutative case [5]. To show that they are in fact equal, it is su□cient to show that each of the generators of one ideal is contained in the other. This can be accomplished by using the `NF` procedure to reduce each generator with respect to the generators of the other ideal. Since `Pgb` is a reduced Gröbner basis it can determine ideal membership.

```
> for i from 1 to 3 do NF(GBtm[i],Pgb,Deg[Lex]) end do;
```

$$0, 0, 0$$

This shows that every generator in the list `GBtm` is in the ideal generated by the polynomials (in this case monomials) of `Pgb`.

```
> for i from 1 to 3 do NF(Pgb[i],GBtm,Deg[Lex]) end do;
```

$$0, 0, 0$$

It is important to recall that a minimal Gröbner basis given by the procedure `TmingbG`

```
> N:=4:numofpols:=4:
  i:='i':
  P:=[]:
  for i from 1 to numofpols do
  p||i:=0:
  end do:
  for i from 1 to numofpols do
  while p||i=0 or p||i=Id do
  p||i:=rd_clipolynom(N):
  end do:
  P:=[op(P),p||i];
  end do;
  vars:=[e||(seq(i,i=1..N))];
```

$$P := [7\,Id - 8\,e14]$$

$$P := [7\,Id - 8\,e14,\ 6\,Id + 6\,e13]$$

$$P := [7\,Id - 8\,e14,\ 6\,Id + 6\,e13,\ -4\,e1 + 5\,e3]$$

$$P := [7Id - 8e14,\ 6Id + 6e13,\ -4e1 + 5e3,\ Id - 2e34 - 2e23 + e14 + e13 + e3]$$

$$vars := [e1,\ e2,\ e3,\ e4]$$

```
> GB1:=TglibG(P,Deg[Lex]);

Computed 15 S-polynomials among 6 polynomials in Groebner basis and
needed to compute 15

 This procedure took 17.784 seconds to run.
```

$$GB1 := [Id - 2\,e34 - 2\,e23 + e14 + e13 + e3,\ Id + e13,\ -\frac{7}{8}\,Id + e14,\ e1 - \frac{5}{4}\,e3,\ e3,\ Id]$$

```
> GB1m:=TmingbG(TNBglib,Deg[Lex]);
```

$$GB1m := [Id]$$

```
> Pgb:=PLURALforGlink(P,0,dp,vars,input_for_Singular,
  input_for_Maple,'infty','d');
```

$$Pgb := [Id]$$

```
> evalb(GB1m=Pgb);
```

$$true$$

The last result shows that the two lists are equivalent. Thus the ideal generated by the four polynomials is the whole algebra. Even though the polynomials were restricted to degree at most 4, any degree polynomial will be in the left ideal generated by them. This is from the fact that the reduced Gröbner basis is only the *Id* element and so any polynomial in any degree algebra will be a multiple of the generator.

Example 6. This example will use randomly generated polynomials of a higher degree. Instead of using four polynomials of degree up to 4, there will be two polynomials of unto degree 6 generated. That is the polynomials will be from the Grassmann algebra $\bigwedge_6$. The same procedure will be used as before.

```
> N:=6:numofpols:=2:
  i:='i':
  P:=[]:
  for i from 1 to numofpols do
  p||i:=0:
  end do:
  for i from 1 to numofpols do
  while p||i=0 or p||i=Id do
  p||i:=rd_clipolynom(N):
  end do:
  P:=[op(P),p||i];
  end do;
  vars:=[e||(seq(i,i=1..N))];
```

$$P := [-e13 + 3\,e46 - e6 - e2356]$$

$$P := [-e13 + 3\,e46 - e6 - e2356,\ e2]$$

$$vars := [e1,\ e2,\ e3,\ e4,\ e5,\ e6]$$

```
> GB1:=TglibG(P,Deg[Lex]);
```

```
Computed 45 S-polynomials among 10 polynomials in Groebner basis and
needed to compute 45

 This procedure took 16.769 seconds to run.
```

$$GB1 := [e1456 + \tfrac{1}{3}e156,\ e13 - 3\,e46 + e6 + e2356,\ e135 + 3\,e456 + e56,\ e146 - \tfrac{1}{3}e16, e156,\ e346 - \tfrac{1}{3}e36,\ e13 - 3\,e46 + e6,\ e16,\ e36,\ e2]$$

```
> GB1m:=TmingbG(GB1,Deg[Lex]);
```

$$GB1m := [e13 - 3\,e46 + e6,\ e16,\ e36,\ e2]$$

```
> Pgb:=PLURALforGlink(P,0,dp,vars,input_for_Singular,
input_for_Maple,'infty','d');
```

$$Pgb := [e2,\ e36,\ e16,\ e13 - 3\,e46 + e6]$$

```
> evalb(convert(GB1m,set)=convert(Pgb,set));
```

$$true$$

In the last result the two lists `GB1m` and `Pgb` were converted to sets and compared since Maple usually puts lists in random order making it di□cult to compare longer lists. The result is *true* and so the two sets are equal and thus the lists are equal. This means that the ideal generated by the polynomials that were randomly generated can be generated by the four polynomials given in `GB1m` and `Pgb`. Thus `TNB` gave the same result as `Plural` did.

Example 7. This example will demonstrate that the GLB does not solve the membership problem but the GLIB will. A few extra lines of code also allow one to follow the computation of the GLB by listing out the S-polynomials as they are computed and the new polynomials as they are added to the GLB G.

```
> f1:=e2456-e3;f2:=e14-e36;f3:=e1256-e13;
  F:=[f1,f2,f3];
```