Basic user's manual of the Faugère's GB package

Corinne Chauvin (chauvin@informatik.uni-tuebingen.de)
Jean-Charles Faugère (jcf@posso.ibp.fr)

1 Presentation

This 'user's manuel' is a short extract from the French documentation ([1, 2]) of the Faugère's GB package. Do not expect a lot from this : only the basic commands that one needs in order to compute Gröbner basis are presented here. If you need more detailed informations. The present translation will be completed and improved.

The purpose of the Faugère's GB package is to compute Gröbner basis of polynomials ideals, and to solve systems of algebraic equations. It is supposed to do quick computations, and has a quite poor documentation in English. The package was written with c and c++.

2 How to call the program

The package is in the directory :

```
/usr/local/lib/GB
```

To call the program :

```
Gb -nox
```

The option -nox is for : no XWindows. If all is right, then you'll see :

```
Grobner Basis Computations (C++ version) , Version SCCS: 3.%I%  %W%  %G%(792)

Warning: this is not the definitive version !
Author:Jean-Charles Faugere ***  Universite Paris 6/LITP
Bug Report:    jcf@posso.ibp.fr
Working directory /u/chauvin
GB>
```

The commands must be typed after the prompt and ended with a simple <return> (no need of semi-column or something else).

3 The basic types

The syntax of GB is a subset of Axiom's one.

- **INT** are the integers. ex:

```
GB> p:INT
GB> p:=3

       3

                         Type: Integer
```

- Elements of prime field ($\mathbf{Z}/\mathrm{p}\mathbf{Z}$). (of course the program doesn't check wether p is a prime integer or not.) ex :

```
GB>  p:PF(3)
GB> p:= 3*2 +1

       1

                         Type: PF(3)
```

  Rem: p must be $< 2^{16}$.

- **Boolean** are available to. ex:

```
GB> b:Boolean
GB> b:= false

       false

                         Type: Boolean
```

  You don't need to declare the type of variables when the type isn't ambiguous.

- For the variables, the type **Variable** exists. It is possible to use simple variable names (a,b,c, etc...) or indexed variables : x.1, ..., x.n.

- Lists : **List(d:Domain)**. A list begins with a [and ends with a]. The elements are separated with ',' . ex:

```
GB> L:=[v.1,v.2,v.3]

       [v ,v ,v ]
         1  2  3
                    Type: List(Variable)
```

There are several functions associated to the lists :

- first element of a list : **first**. ex :

  ```
  GB> first(L)

         v
          1
                      Type: Variable
  ```

- n^{th} element of a list : <**list**>.<**number**>. ex

  ```
  GB> L.2

         v
          2
                      Type: Variable
  ```

  The first element of a list is numbered one.

4 Polynomials

The different polynomials are :

- **DMP(L:List Variable, d:Ring)**. Multivariate Polynomials ordered in respect with the lexicographic ordering. L is the list of the variables, and d the ring of the coefficients. ex:

  ```
  GB> P:DMP([a,b],INT)
  GB> P:= 3* a**2 * b**5 + 2* a**3 + 1
           3     2 5
         2a  + 3a b  + 1

                      Type: GDMP([a,b],Integer,LEXICO(2))
  ```

- **HDMP(L:List, d:ring)**. Multivariate Polynomials ordered in respect with the total ordering[1]. ex:

[1] Degree Reverse Lexicographical

```
GB> Q:HDMP([a,b],INT)
GB> Q:= 3* a**2 * b**5 + 2* a**3 + 1
          2 5     3
        3a b  + 2a  + 1

                      Type: GDMP([a,b],Integer,DEGREVLEX(2))
```

- **SMP(d:ring, L:List Variable)** : Multivariate Polynomials with coefficients in the ring *d*. ex:

```
GB> Q:HDMP([a,b],INT)
GB> Q:= 3* a**2 * b**5 + 2* a**3 + 1;
GB> Q::SMP(INT,[a,b])
          3    5 2
        2a + 3b a + 1

                      Type: SMP(Integer,[a,b])
```

(Use ; to prevent display)

You have to enter your polynomials with type `HDMP`,`DMP` first; in a second time *coerce* then in `SMP` by using operator ::.

- **UP(d:Ring,x:Variable)**: Univariate Polynomials with coefficients in the ring *d* ex:

```
GB> k:=UP(INT,x)
        UP(Integer,x)
                      Type: Domain
GB> p1:k:=x
        x
                      Type: UP(Integer,x)
GB> p2:k:=1
        1
                      Type: UP(Integer,x)
GB> p1:=p1+p1+p2
        2x+ 1
                      Type: UP(Integer,x)
```

- **SUP(d:Ring)**: Univariate Polynomials with coefficients in the ring *d*.

Rem : By default, the sum of the exposants of a monomial have to be less than 2^8.

5 Special functions

The special functions begin with a ')'.

- Quit Gb :

  ```
  )quit, )end, )bye  or )q
  ```

  .

- Display the computating time after each computation : `)time on`. The time is displayed as follows :

  ```
  Serveur: 0.02 sec Eval: 0.03 sec Total: 0.05 sec
  ```

  where :

  – Serveur is the sum of CPU time used by any devoted servers called by the central process.

  – Eval is the evaluating time by the central process.

  – Output is the displaying time.

  – Total is the sum of all the times.

- Clear the user's variables : `)clear all`

- Read and execute a file : `)read <file>` or `)r <file>` In the data files, the commentary lines begin with the two characters `--` .

- Display the current directory `)pwd`

- Change the directory : `)cd <directory>`

- `)fortran on` display results in a form suitable for input.

- `)akcl on` display results in lisp form.

6 How to compute the Gröbner basis of an ideal ?

6.1 Three implementations

Three different implementations of the Buchberger algorithm are available in the package : **groebner**, **sugar** and **tgroebner**. It is *hardly* recommended to use:

- **tgroebner** when coefficients of polynomials are integers. It uses to processors. **Warning:** the display "Serveur"time is the sum of CPUs. It is not a probabilistic method.
- **sugar**, which is the most efficient algorithm in most of the cases for $\mathbf{Z}/p\mathbf{Z}$ computations (modular arithmetic).

Syntax :

```
sugar(L)
```

where L is a list of polynomials. The result is an object whose type is ideal.

6.2 Examples

use of `sugar` (example I):

```
GB> I:=sugar(L)
          2       2       2        2     3       2   2 2    2
       [2a b c + a b + b c  + 1,a b c + a  + 3a b ,2a c  + b c

   2
+ c ]

                    Type: List(GDMP([a,b,c],Integer,DEGREVLEX(3)))
Utilisation du serveur rapide
[5]++|x0^4|[6]+[7]++++[8]*****+[9]++*++[10]*+|x1^5|*+|x2^5||29|****[11]****
End of gbasis computation !

End of minimal gbasis computation !
                   4      3          2      3     3        2
       <base=[21a b  + 31a c + 2a b c  + a c  -95a  -285a b

                2          5     2 2         3      4     2
+ 12a b c + 2a c  -7a b,21b  -12b c  + 120b c  + 12c  -63a b

   2     2        2     3    2     2           2                 3
-2a c -6b c -84b c  -12c  -6a  -25b  -12b c -6c  + 95c -63,2a b c

   3        3        2               2 3      4    2 2       3
+ a b + 6a b  -3a b c  + 2a c  - a,6b c  + 21b  -6b c  + 2b c

   2      3     2       2        2    3             2
+ a b + 6b  + 7a c + 20b c  - b c  + c  + 12b c  - c  -7b + 2c
```

```
       4    3        3     3           2      3    3       2
-1,2a c  -2a b -12a b  + 2a c + 18a b c  -2a c  -5a  -15a b

              2                4    2 2        3     4    2
-2a b c  - a c  -4a c + 2a,4b c  -2b c  + 10b c  + 2c  -7a b

   2       2    3           2            5      4    2 2       3
- b c -7b c  -2c  -2b c + 3c  + 10c -7,6c  + 21b  -3b c  + 8b c

    4      2      3     2       2         2     3     2     2
-12c  + 22a b + 6b  + 7a c + 11b c + 14b c  + 4c  -12a  -36b

            2                 4      4     2 2        3     2
+ 18b c -10c  -7b -28c + 20,6a  + 15b  -18b c  + 10b c  + 5a b

     3     2       2       2     3            2
+ 30b  + 5a c + 10b c -5b c  + 5c  + 30b c -5c  -23b + 10c -5

   2 2    4     2 2      3     2     3     2     2       2
,3a b  -3b  + 3b c  -2b c   - a b -6b   - a c -2b c + b c

   3          2                2       2       2        2     3
- c  -6b c + c  + 4b -2c + 1,2a b c + a b + b c  + 1,a b c + a

+ 3a b ,2b c -2b c  + a b + 3b c  -2c + 1,2a c  + b c + c ],

  nb solutions= 29,serveur=|Serveur: serveur__Dexp__INT|>

                    Type: Ideal
```

and with `tgroebner` (example II):

```
GB> vars:=[a,b,c,d,e]
       [a,b,c,d,e]
GB> D:=HDMP(vars,INT)
GB> l:=cyclic(5,D)

       [a b c d e -1,a b c d + a b c e + a b d e + a c d e + b c d e

,a b c + b c d + a b e + a d e + c d e,a b + b c + c d + a e

+ d e,a + b + c + d + e]
```

```
                    Type: List(GDMP([a,b,c,d,e],Integer,DEGREVLEX(5)))
GB> ideal:=tgroebner(l);
Assistant on: fusy.ibp.fr path: /usr/local/lib/GB/serveur_t__Dexp__F
Connection established with assistant
     /usr/local/lib/GB/serveur_t__Dexp__INT on posso.ibp.fr
Connection established with serveur_t__Dexp__INT on posso.ibp.fr
Utilisation du serveur rapide

*** Extend Memory (DMP__Dexp__INT) Tue Mar  8 18:52:44 1 alloc:800000 octets
a) Compute gbasis using modular trace
[2]+|x1^2|[3]+[4]++[5]+++![6]++++++!![7]+++++!!!!+!!!!![8]++++++|x2^3|
End of modular computation (0.25 sec)
!!!!!!!!!!![9]++++|x3^4|!!!!!!!!!!!!![10]!!!!!!!!!![11]+++!!![12]++
!!!!!!!!!![13]+++|x4^8||70|!!!!!!![14]!!!!!!!!!
End of gbasis computation !

End of minimal gbasis computation !
    a) done in 0.18 sec + 0.25 sec
b) Check that input polynomials are in gbasis ...done in 0.00 sec
c) Check that gbasis is a groebner basis
[4]**[5]**********[6]***[7]*******[8]***********[9]************
End of gbasis computation !
    c) done in 0.10 sec
0.25 sec
```

6.3 Manipulating ideals

You can obtain the basis under the form of a list of polynomials with the command : `base(I)` where I is an ideal. You can obtain the *the number of solutions* of the basis with `nbSolutions(I)`. `generators(I)` gives you the initial system of generators of I. If you have a list of generators l and that you *know* it is also a Gröbner basis of I you can use `pretendgbasis(l)` to create I without computations.

6.4 Change of ordering

It is possible to convert a Gröbner basis computed with respect to the total ordering into a basis with respect to the lexicographic ordering in zero dimensional case (FGLM algorithm). example I': let *ideal* be the ideal from example I

```
GB> I:=pretendgbasis(base(ideal));
Connection established with serveur__Dexp__INT on posso.ibp.fr
```

```
Initialisation du serveur rapide

*** Extend Memory (DMP__Dexp__INT) Tue Mar  8 19:09:45 1 alloc:800000 octets
GB> glex:=totolex(I)
(70)
e/e**2/e**3/e**4/e**5/e**6/e**7/e**8/e**9/e**10/e**11/e**12/e**13/
e**14/e**15/|x4|d/d**2/d**3/d**4/d**5/d**6/d**7/|x3|c/c*d/c**2/
c**3/|x2|b/b*d/b*c/b**2/|x1|a/|x0|
                             2               6         5 2
       [a + b + c + d + e,275b  + 825b e + 550d e + 1650d e

      4 3      3 4       2         11           6                  12
+ 275d e  -550d e  + 275d  -566d e   -69003d e  + 69019d e -1467e

               ................................

            12       7       2  15        10       5
+ 233d e -8e   -979e  + 987e ,e   + 122e   -122e  -1]

                    Type: List(GDMP([a,b,c,d,e],Integer,LEXICO(5)))
```

6.5 Triangular sets

It is now possible to compute a list of triangular sets (that is to say $\sqrt{I}$) – this works only in zero dimensional case –. Use `lextriang(l)` where l is lexicographic Gröbner basis (type `List SMP(INT, List Variable)`.

For instance with example I':

```
GB> S:=SMP(INT,vars)

       SMP(Integer,[a,b,c,d,e])
GB> G:=glex :: List(S);
GB> lextriang(G)
Connection established with lextriang on posso.ibp.fr
Utilisation du serveur rapide
2/3/4/5/
          5      6      5    2 4    3 3    4 2               5
       [[e + -1,d + 4ed + 5e d + 5e d + 5e d + 4d+ e,5c+ 8ed

     2 4     3 3     4 2                  5       2 4      3 3
+ 30e d + 30e d + 25e d + 30d+ 22e,5b+ -2ed + -10e d + -15e d

      4 2                   5       2 4      3 3      4 2
```

```
+ -10e d + -10d+ -8e,5a+ -6ed + -20e d + -15e d + -15e d + -15d

         5                     2        2             5
+ -9e],[e + -1,d+ - e,c+ - e,b + 3eb+ e  ,a+ b+ 3e],[e + -1,d

       2        2                   10       5       2      6
+ - e,c + 3ec+ e ,b+ c+ 3e,a+ - e],[e  + 123e + 1,55d + (-2e

             7        2       6             6
+ -233e)d+ -8e + -987e ,55c+ e + 144e,55b+ e + 144e,55a+ 55d

     6
+ -2e + -233e]]

                    Type: List(List(SMP(Integer,[a,b,c,d,e])))
```

Generally it is easy to read solutions on that form.

6.6 Normal form of a polynomial

You can compute the normal form of a polynomial in respect to an ideal with the function **normalForm(P,G)**, where **P** is a polynomial and **G** is an ideal (the result of a call to the sugar function for instance). The result will be a "record" **(Pr, coef)**, such that **Pr** x **coef** is the normal form of **P**.

Ex : Let G be the ideal from the last example.

```
GB> P1
         3 2         4      3
       2a b c + 5a b c  + a c  + 2

                    Type: GDMP([a,b,c],Integer,DEGREVLEX(3))
GB> P1r:= normalForm(P1, G)
Utilisation du serveur rapide
             3           3      3             2        3      3
       [1050a b + 1890a b  + 96a c + 1002a b c  + 60a c  -324a

       2                 2
-972a b  -120a b c -384a c  -420a c + 420a + 168,84]

                    Type: List(GDMP([a,b,c],Integer,DEGREVLEX(3)))
```

Rem: the right name of `normalForm` should be `pseudonormalForm` for integer coefficients.

6.7 What does it mean `*+!` ?

During computation of a Gröbner basis by Buchberger algorithm GB uses a convenient way to show briefly the state of computations:

- `+` means that the last S-polynomial computed was not zero.
- `*` means that the last S-polynomial computed was zero.
- `!` means that the last S-polynomial computed was not computed because it is probably zero.
- `x2 ^3` means that the leading term of the last S-polynomial computed was $\alpha \times x_2^3$, α a coefficient.
- `[7]` means that Sugar Strategy reach sugar 7
- `|70|` means that there is a finite number of solutions ≤ 70.

References

[1] Jean-Charles Faugère *Documentation hyper-doc de Gb* Try `xmosaic -home http://posso.ibp.fr/Gb.html` everywhere in the world.

[2] Jean-Charles Faugère *Résolution des systèmes d'équation algébriques* PHD Thesis Université Paris 6 (1994)