grobner-libgiac

Times are on a 2 core 2012 notebook:

Intel(R) Core(TM) i5-2435M CPU @ 2.40GHz

```
n=6;R=PolynomialRing(QQ,n,'x');
I = sage.rings.ideal.Cyclic(R,n);
I.gens()
```

```
[x0 + x1 + x2 + x3 + x4 + x5, x0*x1 + x1*x2 + x2*x3 + x3*x4 +
+ x4*x5, x0*x1*x2 + x1*x2*x3 + x2*x3*x4 + x0*x1*x5 + x0*x4*x5
x3*x4*x5, x0*x1*x2*x3 + x1*x2*x3*x4 + x0*x1*x2*x5 + x0*x1*x4*
x0*x3*x4*x5 + x2*x3*x4*x5, x0*x1*x2*x3*x4 + x0*x1*x2*x3*x5 +
x0*x1*x2*x4*x5 + x0*x1*x3*x4*x5 + x0*x2*x3*x4*x5 + x1*x2*x3*x
x0*x1*x2*x3*x4*x5 - 1]
```

```
time B = I.groebner_basis("libsingular:std")
```

```
Time: CPU 0.14 s, Wall: 0.15 s
```

```
import giacpy
from giacpy import *
```

```
// Giac share
root-directory:/home/fred/dev/sage/git-trac-command/local/sha
/
// Unable to find keyword file
/home/fred/dev/sage/git-trac-command/local/share/giac/doc/fr/
s
// Giac share
root-directory:/home/fred/dev/sage/git-trac-command/local/sha
/
Help file
/home/fred/dev/sage/git-trac-command/local/share/giac/doc/fr/
s not found
Added 0 synonyms
```

```
I1=libgiac(I.gens())
```

In giac 1.1 there is a new code for grobner basis with revlex.

It is much faster than the old ones, but only with revlex and doesn't use cocoa.

```
time B1=I1.gbasis([R.gens()],'revlex')
```

```
    Time: CPU 0.09 s, Wall: 0.09 s
```

```
time B1=I1.gbasis([R.gens()],'revlex,cocoa=false')
```

```
    Time: CPU 0.09 s, Wall: 0.09 s
```

```
[(B[-i-1]/B1[i]).simplify()  for i in range(len(B))] #singular
donne des fractions. les degres ne sont pas dans le meme ordre
```

```
    [1,
     1,
     1/14,
     1,
     1/2,
     1/7,
     1/2,
     1/903,
     1/1806,
     1/258,
     1/903,
     1/903,
     1/1806,
     1/903,
     1/1806,
     1/1806,
     1/7,
     1/602,
     1/258,
     1/1806,
     1/18,
     1/274400,
     1/54880,
     1/109760,
     1/164640,
     1/86596,
     1/43298,
     1/86596,
     1/35739035160,
     1/2978252930,
     1/1191301172,
     1/2382602344,
     1/3573903516,
     1/35739035160,
     1/14,
     1/24990,
     1/42,
     1/24990,
     1/99960,
     1/6664,
     1/99960,
```

```
 1/19992,
 1/99960,
 1/2040,
 1/16660]
```

```
n=8;R=PolynomialRing(QQ,n,'x');
I = sage.rings.ideal.Cyclic(R,n);I
```

```
Ideal (x0 + x1 + x2 + x3 + x4 + x5 + x6 + x7, x0*x1 + x1*x2 +
+ x3*x4 + x4*x5 + x5*x6 + x0*x7 + x6*x7, x0*x1*x2 + x1*x2*x3
x2*x3*x4 + x3*x4*x5 + x4*x5*x6 + x0*x1*x7 + x0*x6*x7 + x5*x6*
x0*x1*x2*x3 + x1*x2*x3*x4 + x2*x3*x4*x5 + x3*x4*x5*x6 + x0*x1
+ x0*x1*x6*x7 + x0*x5*x6*x7 + x4*x5*x6*x7, x0*x1*x2*x3*x4 +
x1*x2*x3*x4*x5 + x2*x3*x4*x5*x6 + x0*x1*x2*x3*x7 + x0*x1*x2*x
x0*x1*x5*x6*x7 + x0*x4*x5*x6*x7 + x3*x4*x5*x6*x7, x0*x1*x2*x3
+ x1*x2*x3*x4*x5*x6 + x0*x1*x2*x3*x4*x7 + x0*x1*x2*x3*x6*x7 +
x0*x1*x2*x5*x6*x7 + x0*x1*x4*x5*x6*x7 + x0*x3*x4*x5*x6*x7 +
x2*x3*x4*x5*x6*x7, x0*x1*x2*x3*x4*x5*x6 + x0*x1*x2*x3*x4*x5*x
x0*x1*x2*x3*x4*x6*x7 + x0*x1*x2*x3*x5*x6*x7 + x0*x1*x2*x4*x5*
x0*x1*x3*x4*x5*x6*x7 + x0*x2*x3*x4*x5*x6*x7 + x1*x2*x3*x4*x5*
x0*x1*x2*x3*x4*x5*x6*x7 - 1) of Multivariate Polynomial Ring
x1, x2, x3, x4, x5, x6, x7 over Rational Field
```

```
#time B = I.groebner_basis("libsingular:std")
```

```
I1=libgiac(I.gens())
```

```
# cyclic 8 over QQ (with probabilistic algorithms)
time B1=libgiac(I.gens()).gbasis([R.gens()],'revlex')
```

```
Running a probabilistic check for the reconstructed Groebner
If successfull, error probability is less than 1e-15 and is
estimated to be less than 10^-105. Use proba_epsilon:=0 to ce
(this takes more time).
Time: CPU 274.18 s, Wall: 84.38 s
```

```
p=65521
n=8;R=PolynomialRing(GF(p),n,'x');
I = sage.rings.ideal.Cyclic(R,n);
```

```
# cyclic 8 over GF(65521)
time B = I.groebner_basis("libsingular:std")
```

```
Time: CPU 47.90 s, Wall: 47.91 s
```

```
Ip=libgiac(I.gens()) % p;
```

```
# cyclic 8 over GF(65521)
time BG = Ip.gbasis([R.gens()])
```

```
    Time: CPU 10.84 s, Wall: 10.84 s
```

```
Ip=libgiac(I.gens()) % 33554393;
```

```
# cyclic 8 over GF(33554393)
time BG = Ip.gbasis([R.gens()])
```

```
    Time: CPU 13.00 s, Wall: 13.00 s
```

Two different way to convert the giac answer to sage.

```
time rep1=vector(B1.revlist(),R).list();
time rep2=[R(j) for j in B1.revlist()];
rep1==rep2
```

```
    Time: CPU 2.02 s, Wall: 2.83 s
    Time: CPU 1.90 s, Wall: 1.90 s
    True
```

```
B1.savegen('c8.gen')
```

```
    c8.gen
```

```
giacsettings.proba_epsilon=0
```

```
# cyclic 8 over QQ with giac without proba algo
time B2=I1.gbasis([R.gens()],'revlex')
```

```
    Time: CPU 363.17 s, Wall: 165.87 s
```

```
giacsettings.proba_epsilon=1e-15
```

```
n=9;R=PolynomialRing(QQ,n,'x');
I = sage.rings.ideal.Katsura(R,n);
I.gens()
```

```
    [x0 + 2*x1 + 2*x2 + 2*x3 + 2*x4 + 2*x5 + 2*x6 + 2*x7 + 2*x8 -
    x0^2 + 2*x1^2 + 2*x2^2 + 2*x3^2 + 2*x4^2 + 2*x5^2 + 2*x6^2 +
    + 2*x8^2 - x0, 2*x0*x1 + 2*x1*x2 + 2*x2*x3 + 2*x3*x4 + 2*x4*x
    2*x5*x6 + 2*x6*x7 + 2*x7*x8 - x1, x1^2 + 2*x0*x2 + 2*x1*x3 +
    + 2*x3*x5 + 2*x4*x6 + 2*x5*x7 + 2*x6*x8 - x2, 2*x1*x2 + 2*x0*
    2*x1*x4 + 2*x2*x5 + 2*x3*x6 + 2*x4*x7 + 2*x5*x8 - x3, x2^2 +
    + 2*x0*x4 + 2*x1*x5 + 2*x2*x6 + 2*x3*x7 + 2*x4*x8 - x4, 2*x2*
    2*x1*x4 + 2*x0*x5 + 2*x1*x6 + 2*x2*x7 + 2*x3*x8 - x5, x3^2 +
    + 2*x1*x5 + 2*x0*x6 + 2*x1*x7 + 2*x2*x8 - x6, 2*x3*x4 + 2*x2*
    2*x1*x6 + 2*x0*x7 + 2*x1*x8 - x7]
```

```
time B = I.groebner_basis("libsingular:std")
```

```
    Time: CPU 20.70 s, Wall: 20.70 s
```

```
giacsettings.proba_epsilon=1e-15;
Igiac=libgiac(I.gens());
time Bgiac=Igiac.gbasis([R.gens()],'revlex')
```

```
Running a probabilistic check for the reconstructed Groebner
If successfull, error probability is less than 1e-15 and is
estimated to be less than 10^-133. Use proba_epsilon:=0 to ce
(this takes more time).
Time: CPU 4.74 s, Wall: 3.43 s
```

```
test=[(Bgiac[i]/B[-i-1]).simplify()  for i in range(len(B))]
#singular donne des coeffs ds Q et giac dans Z. les degres ne
sont pas dans le meme ordre.
test[123]
```

```
234541422096679153195398915016777983756306316800000000000000000
```

```

```

```
p=65521
n=12;R=PolynomialRing(GF(p),n,'x');
I = sage.rings.ideal.Katsura(R,n);
I.gens()
```

```
[x0 + 2*x1 + 2*x2 + 2*x3 + 2*x4 + 2*x5 + 2*x6 + 2*x7 + 2*x8 +
2*x10 + 2*x11 - 1, x0^2 + 2*x1^2 + 2*x2^2 + 2*x3^2 + 2*x4^2 +
+ 2*x6^2 + 2*x7^2 + 2*x8^2 + 2*x9^2 + 2*x10^2 + 2*x11^2 - x0,
2*x0*x1 + 2*x1*x2 + 2*x2*x3 + 2*x3*x4 + 2*x4*x5 + 2*x5*x6 + 2
+ 2*x7*x8 + 2*x8*x9 + 2*x9*x10 + 2*x10*x11 - x1, x1^2 + 2*x0*
2*x1*x3 + 2*x2*x4 + 2*x3*x5 + 2*x4*x6 + 2*x5*x7 + 2*x6*x8 + 2
+ 2*x8*x10 + 2*x9*x11 - x2, 2*x1*x2 + 2*x0*x3 + 2*x1*x4 + 2*x
2*x3*x6 + 2*x4*x7 + 2*x5*x8 + 2*x6*x9 + 2*x7*x10 + 2*x8*x11 -
x2^2 + 2*x1*x3 + 2*x0*x4 + 2*x1*x5 + 2*x2*x6 + 2*x3*x7 + 2*x4
2*x5*x9 + 2*x6*x10 + 2*x7*x11 - x4, 2*x2*x3 + 2*x1*x4 + 2*x0*
2*x1*x6 + 2*x2*x7 + 2*x3*x8 + 2*x4*x9 + 2*x5*x10 + 2*x6*x11 -
x3^2 + 2*x2*x4 + 2*x1*x5 + 2*x0*x6 + 2*x1*x7 + 2*x2*x8 + 2*x3
2*x4*x10 + 2*x5*x11 - x6, 2*x3*x4 + 2*x2*x5 + 2*x1*x6 + 2*x0*
2*x1*x8 + 2*x2*x9 + 2*x3*x10 + 2*x4*x11 - x7, x4^2 + 2*x3*x5
2*x2*x6 + 2*x1*x7 + 2*x0*x8 + 2*x1*x9 + 2*x2*x10 + 2*x3*x11 -
2*x4*x5 + 2*x3*x6 + 2*x2*x7 + 2*x1*x8 + 2*x0*x9 + 2*x1*x10 +
2*x2*x11 - x9, x5^2 + 2*x4*x6 + 2*x3*x7 + 2*x2*x8 + 2*x1*x9 +
2*x0*x10 + 2*x1*x11 - x10]
```

```
# Katsura 12 over GF(65521)
time B = I.groebner_basis("libsingular:std")
```

```
Time: CPU 814.13 s, Wall: 814.09 s
```

```
Ip=libgiac(I.gens()) % p
```

```
# Katsura 12 over GF(65521)
time BG = Ip.gbasis([R.gens()])
```

```
Time: CPU 90.45 s, Wall: 90.45 s
```

```
# conversion to sage
```

```
time V=vector(BG.revlist(),R)
```

```
    Time: CPU 42.91 s, Wall: 42.90 s
```

```
type(V[0])
```

```
    <type
    'sage.rings.polynomial.multi_polynomial_libsingular.MPolynomi
    ingular'>
```

```

```