CoCoALib - Slug #1375

Radical 0-dim: varied timings

09 Dec 2019 21:06 - John Abbott

Status:	Closed	**Start date:**	09 Dec 2019
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Improving	**Estimated time:**	4.90 hours
Target version:	CoCoALib-0.99800	**Spent time:**	4.85 hours

Description

I have observed some strange variations in timings when computing the RGB of a radical:

```
L := [-x^4*y +x^2*y^3,  x^5 -y^4*z -x*y*z^2,  z^5 -y^2 -y];

TIMES := [];
for j := 1 to 10 do
  I := ideal(L);
  RGBI := ReducedGBasis(I);
  radI := radical(I);
  t0 := CpuTime();
  RGBrad := ReducedGBasis(radI);
  t1 := CpuTime();
  append(ref TIMES, floor(t1-t0));
endfor;
println TIMES;
```

The list of times observed is:

```
[0,  75,  75,  76,  81,  75,  75,  75,  77,  80]
```

The initial 0 is correct: sometimes the computation is instant, sometimes it's quite slow. Why?

Related issues:

Related to CoCoA-5 - Slug #948: radical is slow (compared to singular) on the...	**Closed**	**18 Oct 2016**
Related to CoCoALib - Bug #1376: GBasisTimeout: not working as expected	**Closed**	**12 Dec 2019**
Related to CoCoALib - Design #1378: Create two separate radical fns (for 0-di...	**New**	**20 Dec 2019**
Related to CoCoA-5 - Support #1387: John's visit Feb 2020	**Closed**	**07 Jan 2020**
Related to CoCoALib - Feature #1417: RadicalZeroDim with extra parameter for ...	**In Progress**	**14 Feb 2020**
Related to CoCoALib - Support #1584: Benchmarks?	**New**	**19 Mar 2021**

History

#1 - 09 Dec 2019 21:08 - John Abbott

I wanted to show the students an example where it is much better to use IsInRadical than compute the radical (with GB), and then check if the poly is in the radical. But it is a bit tricky if sometimes the radical computation is instant rather than lengthy. :-/

#2 - 10 Dec 2019 13:28 - John Abbott

This getting more puzzling!

Here is another example: essentially the same code as in the original description but

```
L := [-y^5 -x*y^3*z -x*y^2*z,  x^2*y^3 +x^3*z^2 +x*y^2*z^2,  -x*y^3*z -x^4 +z^4];
```

and make the loop do 80 iterations. This is quite quick -- less than 0.25 per iteration.
BUT if I them tell the computer to do 90 iterations, the result is:

```
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  111,  0,  0,  0,  0,
 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0
,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
 0,  0,  0,  0,  0,  0,  0,  0]
```

In both cases I started from a brand new CoCoA-5 session, so the random sequences should be the same (does radical use random numbers?)

#3 - 10 Dec 2019 14:41 - John Abbott

I have done more testing. The timings really do vary from run to run.

I wrote a shell script which took exactly the same input, and ran it several times in separate completely new CoCoA-5 sessions...

```
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  20,  0,  0,  20,  0,  0,  0,  0,  0,
 0,  19,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  20,  0,  0,  0,  0,  19,  0,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  20,  19,  0,  0,  0,  0,  0,  0,  0,  0,  0,
 0,  0,  0,  0,  0,  19,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  19,  0,  0,  0,  0,  0,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
,  0,  0,  0,  0,  0,  0,  0,  0,  0,  19,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  19,  19,
 0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  19,  0,  0,  0,  0,  0,  0,  19,  19,  19,  19,  19,
 19,  19,  19,  19,  0,  0,  19,  19,  0,  0,  19,  19,  20,  19,  19,  19,  19,  19,  21,  21,  19,  19,  0,
 19]
[0,  0,  0,  0,  0,  0,  19,  0,  0,  0,  0,  19,  0,  19,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
  0,  0,  0,  19,  0,  0,  0,  0,  0,  19,  21,  21,  0,  0,  0,  0,  0,  0,  19,  19,  0,  0,  21]
```

As you can see the timing do vary considerably. What is going on?

For this experiment I modified the CoCoA input to 50 iterations, and the generators were

```
L := [-y^5 -x^3*y*z +x^2,  z^5 -x*y^3 +y*z^3,  x^2*y^3 -x^3*y -x^2*z^2];
```

UPDATE I repeated again the experiment, but with only 10 iters per run: here is the output:

```
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  19,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  0,  18,  0]
[0,  0,  0,  0,  19,  19,  0,  0,  0,  0]
[0,  19,  19,  19,  18,  18,  18,  19,  18,  18]
```

```
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[0,  0,  0,  0,  0,  0,  0,  0,  0,  0]
[0,  0,  18,  0,  0,  0,  0,  0,  0,  0]
```

#4 - 10 Dec 2019 15:12 - Anna Maria Bigatti

Sometimes radical is computed with the GBasis, sometimes not: this depends on a timeout, which probably is a very critical value. Try changing your append line with

```
  append(ref TIMES, [floor(t1-t0), floor(CpuTime()-t1)]);
```

I'm guessing that the sum of the two components might be similar. (a bit too slow to test on my computer)
For me it is always [0, 230].

#5 - 11 Dec 2019 15:08 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 20

Following Anna's suggestion I moved t0 := CpuTime(); to just before the computation of the radical; so the time difference counts both the computation of the radical, and the computation of the RGB. But I still observed considerable variation in the times.

Anna also pointed out that radical for zero-dim ideals is an *introspective* implementation: *i.e.* the computation path it follows is affected by how long it took to do some earlier steps. Specifically, internally the function computes a GB with a timeout, and the value for the timeout depends on computation time so far. So slight variations in computation time could lead to radical following difference computation paths and consequently producing different (but mathematically equivalent) answers.

I changed SparsePolyOps-ideal-radical.C around line 353 to increase the timeout, and now all my test examples are "almost instant".

So what is a sensible value to use for the timeout?

#6 - 11 Dec 2019 15:25 - Anna Maria Bigatti

- Related to Slug #948: radical is slow (compared to singular) on these examples added

#7 - 12 Dec 2019 11:33 - John Abbott

The current solution (increasing the "increment" to 0.5) works well on the small examples I have tried: sometimes the radical takes a noticeable amount of time (*e.g.* 1-2s), but then the RGB can be computed very quickly.
What I fear is that at some point the problem will resurface with larger ideals...

Anyway, the current code (already in CVS) certainly works better than the old version on the examples I have.

#8 - 12 Dec 2019 13:33 - John Abbott

Here are some more examples where it takes a long time to compute RGB after computing radical:

```
I1 := ideal([x^6 +y^6,  y^6 -y^5*z +x^2*y^3,  z^6 +x^3*y +x]);  --> stopped after 3800s, delta was 2.5s
I2 := ideal([x^6 -x^5*y -x*y^2*z^3,  y^6 +y*z^5,  z^6 -x*y^4 -x^4*z]); --> stopped after about 900s, delta was
 2.5s
I3 := ideal([x^6 +x^2*z^4,  y^6 +x*y^3*z^2 -x^2*z^4,  z^6 -y^4*z +x^3*z^2]); --> took 970s, delta was 2.5s
I4 := ideal([x^6 +2*x*y^4*z -x*y^2*z^3,  y^6 +x^2*y*z^2,  z^6 -2*x^3*y +2*y^3*z]); --> stopped after 6000s, de
lta was 2.5s
```

#9 - 12 Dec 2019 15:30 - John Abbott

I have modified my program to call GBasisTimeout, and I specified 30s as the time-out.
I was a bit surprised to find an example where the time-out actually occurred after 975s; does that mean that 1 single reduction took over 900s? 8-O

Here is the computation:

```
L:=[x^6 +x^2*z^4,  y^6 +x*y^3*z^2 -x^2*z^4,  z^6 -y^4*z +x^3*z^2];
radI := radical(ideal(L));
GB := GBasisTimeout(radI,30); -->  radical+GB took together 975s
```

Maybe radical was very slow? But the internal GB computations have each a timeout delta of 2.5s

#10 - 12 Dec 2019 17:02 - John Abbott

- Related to Bug #1376: GBasisTimeout: not working as expected added

#11 - 13 Dec 2019 11:57 - Anna Maria Bigatti

Instead of guessing, do this ;-)

```
L := [x^6 +x^2*z^4,  y^6 +x*y^3*z^2 -x^2*z^4,  z^6 -y^4*z +x^3*z^2];
t0 := CpuTime(); radI := radical(ideal(L)); println "*** radical *** ", TimeFrom(t0);
SetVerbosityLevel(100);
t0 := CpuTime(); GB := GBasisTimeout(radI,30); println "*** GB *** ", TimeFrom(t0);
```

I got

```
--> ERROR: Computation exceeded given time limit
```

```
--> [CoCoALib] ReduceActiveLM
--> t0 := CpuTime(); GB := GBasisTimeout(radI,30); println "GB: ", TimeFrom(t0) ...
-->                       ^^^^^^^^^^^^^^^^^^^^^^
GB: 106.306
```

I think you had dome a "trick" to minimize the internal number of calls of CpuTime, so it might be that the estimate of the number of steps to do was too high.

#12 - 20 Dec 2019 17:05 - John Abbott

- % Done changed from 20 to 30

I propose doing an experiment along the following lines.

We create a collection of test ideals (perhaps as many as 25-100?)
For each ideal we compute radical, and then RGB of radical, and note the times of the two steps.

We generate several tables of times choosing different delta values for the GB time-outs **inside** the radical code.

The hope is that this will help us find a reasonable *heuristic* value for delta.

#13 - 20 Dec 2019 17:10 - John Abbott

- Related to Design #1378: Create two separate radical fns (for 0-dim ideals) added

#14 - 09 Jan 2020 11:32 - John Abbott

- Related to Support #1387: John's visit Feb 2020 added

#15 - 14 Feb 2020 15:38 - Anna Maria Bigatti

Now the problem is: which time limit to give to GBasis computations?

Currently there is "variable bound" + "fixed bound".
I suggest
"variable bound" = max_time_MinPoly*i (i = #of MinPoly yet to do)
"fixed bound" = 1 sec

The examples at point #1375#note-8 below:
With "fixed bound" = 1sec they take about 2*1 + epsilon sec (no GB computed in time)
With "fixed bound" = 5sec they take about 2*5 + epsilon sec (no GB computed in time)
With "fixed bound" = 10sec, they all succeed in computing the first GB, and then the rest of the radical computation becomes a lot easier: they take 7~9 sec

```
/**/ SetVerbosityLevel(40);  -- to see the timeouts
/**/ t0 := CpuTime();   radI := radical(I1);   TimeFrom(t0);
/**/ t0 := CpuTime();   radI := radical(I2);   TimeFrom(t0);
/**/ t0 := CpuTime();   radI := radical(I3);   TimeFrom(t0);
/**/ t0 := CpuTime();   radI := radical(I4);   TimeFrom(t0);
```

#16 - 14 Feb 2020 16:06 - Anna Maria Bigatti

- Related to Feature #1417: RadicalZeroDim with extra parameter for GBasis timeout added

#17 - 15 Feb 2020 15:13 - John Abbott

- % Done changed from 30 to 50

There is a similar line in myTestIsPrimary_0dim (around line 252).
Make the same change there???

#18 - 21 Feb 2020 20:23 - John Abbott

Here are some test cases I found using a random search (ensuring the gens were x^6+lower, y^6+lower, z^6+lower):

```
-- For these ideals cocoa failed to compute radical with GB with 4s allowance per internal GB
-- [filtered from previous list -- see radical-speed-test1.cocoa5]
L :=[
  [x^6 -2*x*y^4*z +2*y^2*z^4,  y^6 -x*y^3*z^2 -y^4*z^2 -2*x*y*z^4,  z^6 +x^4*z -x^3*z^2],
  [x^6 -2*x*y^4 -x^2*y^2*z,  y^6 -2*x^4*y,  z^6 +2*y^4 -x^3*z -2*y^2*z^2],
  [x^6 -x*z^5 +z^6 -2*x^3*y,  y^6 +x^5 -y^4*z -2*x*y*z^2,  z^6 +x^2*y^3 +2*x^3*y],
  [x^6 -y^6 -2*x^2*y^2*z^2 -2*x^4*z,  y^6 +2*x^5*z -x*z^4,  z^6 +x^2*y^2*z +2*x^2*y*z^2 +y^2*z^3],
  [x^6 -2*x^2*y^4 -2*y^4*z^2 -x^2*y^2*z,  y^6 +2*y^2*z^3 +2*x^2*y*z,  z^6 +2*x^3*y +x^2*y^2 +2*y^3*z],
  [x^6 -2*y^4*z +y^2*z^3,  y^6 +x^2*y*z^3 -2*x*y*z^3 +2*x*y^3,  z^6 -2*x^4*z +2*x^4 -y^2*z],
  [x^6 -y^4*z +2*x^2*z^3,  y^6 -2*z^6 +2*x^3*z^2,  z^6 -x^5 -x^4*y +2*x^2*y*z^2],
  [x^6 -x^5*y +2*y^5 -y^4*z,  y^6 -2*x^2*y^3*z +2*y^3*z^3 -2*x*y^4,  z^6 -y^2*z^3 +2*x^4]
     ];
```

Here are some more "harder" ones:

```
-- Examples where we failed to compute GB for radical with delta=8

L=[x^6 +y^3*z^3 +2*x*y*z^3,  y^6 +2*x^4*y*z -2*x^3*y*z^2 +y^2*z^4,  z^6 +x*y^3 +2*y^4]
L=[x^6 +2*y^4*z^2 -x^3*y^2 -2*x^2*y*z^2,  y^6 -x^2*y^3*z +2*x^2*y^3,  z^6 -2*x^3*y -x]
L=[x^6 +y^6 -2*x*y*z^4,  y^6 +y^2*z^4 +x^2*y*z^2 -2*x^2*z^3,  z^6 -2*x^3*y^2 -x*y^2*z^2 +z^5]
L=[x^6 +2*x^5*y +2*y^5*z,  y^6 +x^2*y^3 +2*x*y^4 -2*x*z^4,  z^6 -2*x^4 -2*y^3*z +2*x*z^3]
L=[x^6 -2*y^6 +2*x^2*z^3 -x*y*z^3,  y^6 -2*y^5*z -2*x^3*y*z^2,  z^6 -x^5 -2*x^3*z^2]
L=[x^6 +x^3*y^2*z -2*y^5*z -x^2*y^2*z^2,  y^6 -2*x^4*y*z -2*x^5,  z^6 -2*x^3*y -2*x^3]
L=[x^6 -2*x^3*y^3 +x^5*z,  y^6 +x^2*y^2*z +2*x*y*z^3 -2*x^2*z,  z^6 +2*x^5 -x^4*y -y*z]
L=[x^6 -2*x^2*y^4 -x^2*y^3*z,  y^6 +2*x^2*y^3*z +2*x^3*y^2 -x*y^3*z,  z^6 -2*x^4*z +y^3*z^2]
L=[x^6 -x^2*y^4 -2*y^4*z^2,  y^6 +y^5*z -2*x^3*z^3 -2*x*y*z^4,  z^6 -2*x^4*y -x*z^4]
L=[x^6 -y^6 -y^3*z^3,  y^6 -2*x*y^4*z -x^2*z^4,  z^6 +2*x^4*z -x^3*z^2]
L=[x^6 +x^3*y^3 -2*x^2*y^2*z +x^3*z^2,  y^6 -2*x*y^3*z^2 +2*z^6,  z^6 +x^5 +2*x*z^4]

-- failed with delta=16
L=[x^6 -x^2*y^4 -2*y^4*z^2,  y^6 +y^5*z -2*x^3*z^3 -2*x*y*z^4,  z^6 -2*x^4*y -x*z^4]
```

NOTE the very last one takes about 28s on my computer if I set a high timeout.

#19 - 12 Mar 2021 10:27 - John Abbott

- Status changed from In Progress to Resolved

- Assignee set to John Abbott

- % Done changed from 50 to 70

I have just tried most of the examples listed in this issue.
The early part about varied timings now seems to be completely fine: timings are (faster and) quite regular.

In the later part about potentially tougher computations, most examples seem to be tolerably fast now.
Nevertheless, I do have the nasty feeling that there is a time limit of 30s hard-coded somewhere :-/

We should gather some of the examples from this issue, and put them into a "speed benchmark test suite",
so that if we make future changes then we can easily verify that there has been no great speed penalty
(or even verify that suddenly everything is much faster... hope, hope).

#20 - 19 Mar 2021 15:42 - John Abbott

- Related to Support #1584: Benchmarks? added

#21 - 04 Feb 2022 21:20 - John Abbott

- Status changed from Resolved to Closed

- % Done changed from 70 to 100

- Estimated time set to 4.90 h

Maybe the current impl is not perfect, but it is good enough for these tests.
Closing.