# CoCoALib - Slug #1646

## radical: could be more clever

17 Jan 2022 12:07 - John Abbott

| | | | |
|---|---|---|---|
| **Status:** | Closed | **Start date:** | 17 Jan 2022 |
| **Priority:** | Normal | **Due date:** | |
| **Assignee:** | John Abbott | **% Done:** | 100% |
| **Category:** | Maths Bugs | **Estimated time:** | 1.33 hour |
| **Target version:** | CoCoALib-0.99800 | **Spent time:** | 1.30 hour |

**Description**

Who would have predicted the following behaviour?

```
/**/ radical(ideal(x^2,x-1,y^2));
ideal(x^2,  x -1,  y^2)
```

It is not wrong, but could be more helpful

**Related issues:**

| | | |
|---|---|---|
| Related to CoCoALib - Design #1647: Suppress zero from ideal generators?  Det... | **Closed** | **20 Jan 2022** |
| Related to CoCoALib - Bug #1779: Radical error with lex (again) | **Closed** | **05 Feb 2024** |
| Related to CoCoALib - Feature #1780: radical for ideals in SparsePolyRing:  c... | **Closed** | **06 Feb 2024** |

## History

**#1 - 17 Jan 2022 13:48 - John Abbott**

*- Category set to Maths Bugs*

*- Target version set to CoCoALib-0.99800*

Bug originally reported by Florian Walsh.  Also:

```
use QQ[x,y];
radical(ideal(x^2,x-x,y^2)); --> error about 0 poly?!?
```

**#2 - 20 Jan 2022 19:14 - John Abbott**

*- Status changed from New to In Progress*

*- % Done changed from 0 to 10*

The source code for radical is still in package radical.ckpg5.

If the ideal is 0-dim then the C++ function **radical_tmp** is called.
This is defined in SparsePolyOps-ideal.C near line 1010.
If the ideal is 0-dim then the work is done by radical_0dim which is
defined in the same file around line 980.  This in turn delegates to
radical_0dimDRL if the ideal already has a GBasis, or else a copy is
made in ring with DegRevLex before delegating.

The actual work is then done by a member function (?!?) called
myRadical_dimDRL with a comment that the mem fn "behaves differently".

Oha!

Now the source is in SparsePolyOps-ideal-ZeroDim.C  around line 340.

**#3 - 20 Jan 2022 19:31 - John Abbott**

*- Status changed from In Progress to Resolved*

*- Assignee set to John Abbott*

*- % Done changed from 10 to 70*

The problem was that myGBasisByHomog tried to homogenize all generators without checking whether they are zero.

I have added a simple check, and now it seems to work as hoped/desired.

I do wonder whether whether ideals should automatically suppress zeroes from the list of generators.
This should probably be a new issue -- see issue #1647.

**#4 - 20 Jan 2022 19:35 - John Abbott**

*- Related to Design #1647: Suppress zero from ideal generators?  Detect 1 and simplify generators? added*

**#5 - 20 Jan 2022 20:28 - John Abbott**

*- % Done changed from 70 to 80*

I have checked in my changes (and asked Anna to check them).
I have added tests (CoCoA-5 exbugs... currently radical is not really available from CoCoALib).

**#6 - 21 Jan 2022 10:38 - John Abbott**

*- Status changed from Resolved to Closed*

*- % Done changed from 80 to 100*

*- Estimated time set to 1.33 h*

**#7 - 24 Jan 2022 09:07 - Anna Maria Bigatti**

I think that, along the same line -- actually in the previous line ;-)
we should change

```
/**/ radical(ideal(zero(CurrentRing), zero(CurrentRing)));
ideal(0,  0)
```

**#8 - 05 Feb 2024 19:00 - Anna Maria Bigatti**

*- Related to Bug #1779: Radical error with lex (again) added*

**#9 - 06 Feb 2024 09:03 - Anna Maria Bigatti**

*- Related to Feature #1780: radical for ideals in SparsePolyRing:  code in C++  added*