CoCoA-5 - Slug #1270

RationalSolve: use MinPolyQuot instead of elim

05 Apr 2019 20:19 - John Abbott

Status:	Closed	**Start date:**	05 Apr 2019
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	enhancing/improving	**Estimated time:**	0.99 hour
Target version:	CoCoA-5.4.0	**Spent time:**	0.95 hour

Description

Currently RationalSolve uses elim, but every use is equivalent to a call to MinPolyQuot (in the subring generated by the indets actually appearing).

Update the code; maybe also translate it into C++?

Related issues:

Related to CoCoA-5 - Bug #724: RationalSolve: wrongly complains about non zer...	**Closed**	**02 Jun 2015**
Related to CoCoALib - Slug #777: SLUG: elimination	**In Progress**	**15 Sep 2015**

History

#1 - 05 Apr 2019 20:19 - John Abbott

- Related to Bug #724: RationalSolve: wrongly complains about non zero-dim even in finite char added

#2 - 05 Apr 2019 20:56 - John Abbott

Before we decide to replace elim by MinPolyQuot we should check that MinPolyQuot is usually faster (I would certainly expect it to be faster).

There is a slightly tricky aspect: the implementation works by "eliminating" indets one at a time; so a recursive call is with a set of polynomials which define a 0-dim ideal in a subring (because we substitute for the indet rather than leaving a linear generator).

#3 - 01 Oct 2019 14:28 - John Abbott

- Target version changed from CoCoA-5.3.0 to CoCoA-5.4.0

#4 - 03 Oct 2019 17:24 - Anna Maria Bigatti

- Related to Slug #777: SLUG: elimination added

#5 - 21 Oct 2019 23:00 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

Here is an example which shows that RationalSolve can be unreasonably slow:

```
use P ::= ZZ/(32003)[x,y,z];
use P ::= QQ[x,y,z];

X := indets(P);
S := support((1+sum(X))^3);  --> deg = 3
define rndpoly(S)
  return sum([random(-9,9)*t | t in S]);
enddefine; -- rndpoly

L := [rndpoly(S) | i in 1..3];
I := ideal(L);
//SetVerbosityLevel(100);
```

```
println "=======================================================";
println "ReducedGBasis";
println "=======================================================";
t0 := CpuTime();
RGB := ReducedGBasis(I);
println "RGB TIME ", TimeFrom(t0); println; println;

J := ideal(L);
println "=======================================================";
println "MinPolyQuot";
println "=======================================================";
t0 := CpuTime();
mu := MinPolyQuot(x,J,x);
println "MPQ TIME ", TimeFrom(t0); println; println;

println "=======================================================";
println "ApproxSolve";
println "=======================================================";
t0:=CpuTime();
solns:=ApproxSolve(L);
println "ApproxSolve TIME ", TimeFrom(t0); println; println;

println "=======================================================";
println "RationalSolve";
println "=======================================================";
t0 := CpuTime();
RationalSolve(L);
println "RatSolve TIME ", TimeFrom(t0); println; println;
```

On my computer, typical timings are about RGB 0.04, MPQ 0.08, ApproxSolve 1.0, RatSolve 2.1.
Increasing deg to 4 makes RationalSolve unreasonably slow: RGB 0.06, MPQ 1.2, ApproxSolve 22, RatSolve 2250

#6 - 12 Feb 2021 11:54 - John Abbott

- Status changed from In Progress to Feedback

- Assignee set to John Abbott

- % Done changed from 10 to 90

- Estimated time set to 0.99 h

I have just tried the example from comment 5, and RationalSolve is now tolerably fast (less than 2s).

#7 - 16 Sep 2021 12:40 - Anna Maria Bigatti

- Status changed from Feedback to Closed

- % Done changed from 90 to 100

tested on Mac, ~0.1s