CoCoALib - Feature #206

Matrix equation solving: LinKer

10 Jul 2012 09:40 - Anna Maria Bigatti

Status:	In Progress	**Start date:**	10 Jul 2012
Priority:	Urgent	**Due date:**	
Assignee:		**% Done:**	30%
Category:	New Function	**Estimated time:**	0.00 hour
Target version:	CoCoALib-0.99880	**Spent time:**	11.50 hours

Description

find all solutions M*x = 0

Related issues:

Related to CoCoALib - Feature #122: Porting "LinKer"	**Closed**	**04 Apr 2012**
Related to CoCoALib - Feature #303: Rows and Columns of a matrix	**Closed**	**11 Feb 2013**
Related to CoCoALib - Feature #125: Matrix equation solving; linear system so...	**In Progress**	**05 Apr 2012**
Related to CoCoALib - Feature #1444: HNF: Hermite Normal Form	**New**	**10 Mar 2020**

History

#1 - 10 Jul 2012 09:43 - Anna Maria Bigatti

- Subject changed from Linear system kernel to Matrix equation solving: Linear system kernel

#2 - 10 Jul 2012 09:58 - Anna Maria Bigatti

- Subject changed from Matrix equation solving: Linear system kernel to Matrix equation solving: LinKer

#3 - 10 Jul 2012 10:17 - Anna Maria Bigatti

Written **LinKerByGauss** for matrices with entries in a field.

The code uses **RankAndGauss(M, ncols)** which computes a simple Gaussian reduction (modifying M) only on the first **ncols** columns. The code was originally written by J Abbott inside **LinSolveByGauss**.

We can use this function in other functions: I have already modified **LinSolveByGauss**.
Probably can be easily called by **InverseByGauss**, **RankByGauss**, ...

#4 - 10 Jul 2012 15:10 - Anna Maria Bigatti

Added **LinKer** to CoCoA-5 and test.

Added test for CoCoAlib (added into ex-matrix3, written for *brother function* **LinSolve**)

Added documentation for CoCoALib and CoCoA-5

#5 - 04 Feb 2013 12:51 - John Abbott

Renzo does not like the current interface where the elements of the kernel appear as **columns** rather than rows. He suggests transposing the result.

In C4 the result was a list of lists; if you applied Mat to the result produced by C4 then you'd get the transpose of the result produced by C5.

In other words LinKer solves M*transposed(x) = 0 for the matrix x.

Think about it. Make a considered decision. Implement & document!

#6 - 05 Feb 2013 09:06 - Anna Maria Bigatti

I think that solutions (and kernel) **X** should be written so that
AX = b and that is what we have chosen for CoCoA-5.

One thing we could add is a couple of functions **Rows** and **Columns** which take a matrix and return a list of list. We already have **List** as in CoCoA-4, but maybe **Rows** and **Columns** would be better names?

In summary: the result as it was in CoCoA-4 is currently given by **List(Transposed(X))**.
Should I add it to the manual?

#7 - 05 Feb 2013 11:55 - Lorenzo Robbiano

Non sono d'accordo. Se si guarda la risposta, non si può proprio capire che le soluzioni sono le colonne.
Infatti quello che si vede sono le righe e sembra una risposta sbagliata. Non ha senso anticipare quello che farà l'utente.
Se ha voglia di verificare e se conosce l'algebra lineare, sa che deve fare moltiplicare la matrice di partenza con la trasposta dela matrice delle soluzioni.
Se non lo sa... deve studiare.

Seconda osservazione. LinSolve fornisce UNA SOLA soluzione. Allora il nome non va bene, è ingannevole.
Bisogna chiamare la funzione OneSolution o qualcosa del genere.

#8 - 06 Feb 2013 08:10 - Anna Maria Bigatti

Both ways make sense, then I think we should follow the usual rule: "do what other systems do" (and then find a good name for the other choice)

So I'll make a search and list here what others do.

LinSolve (work in progress..)

- **Matlab** "X = linsolve(A,B) solves the linear system A*X = B (...) The number of rows of A must equal the number of rows of B. If A is m-by-n and B is m-by-k, then X is n-by-k." (infinite solutions???)
- **Maple** (vertical) vector (infinite solutions???) http://www.maplesoft.com/support/help/Maple/view.aspx?path=Linsolve
- **Mathematica** (horizontal) vector. "LinearSolve gives one of the possible solutions to this underdetermined set of equations"
- **Macaulay2** A, a matrix m by n, b, of size m by r --> x, of size n by r, such that Ax=b
  (one solution) http://www.math.uiuc.edu/Macaulay2/doc/Macaulay2-1.4/share/doc/Macaulay2/Macaulay2Doc/html/_solve.html

LinKer (work in progress..)

- **Mathematica** list of (horizontal) vectors (function is called "NullSpace")
- **Macaulay2** kernel(matrix) --> module (matrix is seen as a homomorphism)

#9 - 01 Aug 2014 08:59 - Anna Maria Bigatti

- Target version set to CoCoALib-1.0

#10 - 06 Jun 2016 15:37 - Anna Maria Bigatti

For aesthetics: now changes sign to the LinKer matrix (so that the entries corresponding to the "pivots" are "1" instead of "-1"). Slightly less efficient (changing signs), but more "natural".
Changing tests accordingly.

#11 - 06 Jun 2016 15:44 - Anna Maria Bigatti

- % Done changed from 0 to 20

#12 - 28 Mar 2017 13:57 - Anna Maria Bigatti

Add also the syntax LinKer(L) where L is a list of linear polynomials.

#13 - 28 Mar 2017 22:32 - Anna Maria Bigatti

- % Done changed from 20 to 30

Anna Maria Bigatti wrote:

> Add also the syntax LinKer(L) where L is a list of linear polynomials.

Modified cousin function LinKerBasis(L) in CoCoA-5, a bit more fiddly in CoCoALib (input checking and managing).

#14 - 27 Feb 2020 18:04 - John Abbott

What is the status of this issue? Surely more than 30% complete?

#15 - 28 Feb 2020 10:09 - John Abbott

- Target version changed from CoCoALib-1.0 to CoCoALib-0.99800

I have implemented **LinKerZZ** simply by copying the old code from TmpFactorDir/linalg/Zkernel.c
The new code probably makes some wasteful copies (so is likely to be a little slower than the old code).

#16 - 10 Mar 2020 15:39 - John Abbott

Florian suggests that the name **LinKerZZ** is superfluous, and that **LinKer** should automatically call LinKerZZ if the matrix is over ZZ.

This seems to be a reasonable suggestion to me. So far we have blurred the distinction between matrices of integers and matrices of rationals: *i.e.* by default a matrix is constructed over the rationals even if all the entries are integers; one must say explicitly if the matrix is to be over ZZ.

The current impl does allow a matrix over QQ to passed to LinKerZZ; internally LinKerZZ simply clears denoms in each row, then calls the fn which does the actual work (on a vector<vector<BigInt>>).

Possible "ambiguity": a user could be surprised by an automatic choice of LinKer function: let M1 and M2 be two matrices with the same entries (which "happen to be" integers). Let's assume that M1 is over ZZ and M2 is over QQ. So the two matrices look almost identical, yet computing the kernel with LinKer would produce two essentially different results. Mmmmm :-/

#17 - 10 Mar 2020 15:44 - John Abbott

- Related to Feature #1444: HNF: Hermite Normal Form added

#18 - 11 Mar 2020 11:22 - Anna Maria Bigatti

John Abbott wrote:

> Florian suggests that the name **LinKerZZ** is superfluous, and that **LinKer** should automatically call LinKerZZ if the matrix is over ZZ.
>
> This seems to be a reasonable suggestion to me. So far we have blurred the distinction between matrices of integers and matrices of rationals: *i.e.* by default a matrix is constructed over the rationals even if all the entries are integers; one must say explicitly if the matrix is to be over ZZ.

I see this point of view, but I like to have LinKerZZ because the meaning (and the timings, I suppose) is more explicit.

Maybe could have both:
LinKerZZ(M) defined as a shortcut of LinKer(mat(ZZ,M)) (currently working only for fields)

Or instead we say that LinKer only works on a field, and we make very explicit functions for other cases (syz, I suppose..)
So, instead of NYI, LinKer(mat(ZZ,M)) would give ERR::NotField, suggesting LinKerZZ(M).

> The current impl does allow a matrix over QQ to passed to LinKerZZ; internally LinKerZZ simply clears denoms in each row, then calls the fn which does the actual work (on a vector<vector<BigInt>>).
>
> **Possible "ambiguity":** a user could be surprised by an automatic choice of LinKer function: let M1 and M2 be two matrices with the same entries (which "happen to be" integers). Let's assume that M1 is over ZZ and M2 is over QQ. So the two matrices look almost identical, yet computing the kernel with LinKer would produce two essentially different results. Mmmmm :-/

#19 - 14 Jan 2021 14:31 - John Abbott

The fn **LinKerZZ** is available in my copy of CoCoA-5, but there is no doc.
The doc for CoCoALib seems explicit.

Still need to answer the qn in comment 16: should LinKerZZ really have a separate name?

NOTE: LinKerZZ is made available in BuiltinOneLiners.C

#20 - 15 Jan 2021 21:07 - John Abbott

This was originally in issue #122; moving it to here.

What is the difference between LinKer and LinKerBasis?
Is LinKerBasis essentially the same as GetCols(LinKer(..))?

Regarding LinKerZZ, I am now inclined ti think that the name LinKerZZBasis might be clearer...
though one could argue that LinKer anyway produces a basis for the kernel (so there is no point in saying so explicitly in the name).

#21 - 15 Jan 2021 21:15 - John Abbott

The source of **LinKerBasis** looks a bit dodgy.
Given a matrix it does what is expected; but it also works for a *list of linear forms*. Anna thinks she wrote the code, but does not recall why now.
Anyway, that feature is undocumented, and the impl is "unsafe" because it does not check its args properly.

So what is the correct way to offer a fn to find the "kernel" of a list of polys (or linear forms)?

HINT: LinKerBasis is defined in **mat.cpkg** around line 600

#22 - 04 Feb 2022 22:08 - John Abbott

- Target version changed from CoCoALib-0.99800 to CoCoALib-0.99850

#23 - 16 Mar 2024 21:51 - John Abbott

Why such a long discussion and no resolution?
Can we finish this quickly? Or should we postpone?

#24 - 22 Mar 2024 09:42 - John Abbott

- Priority changed from Normal to Urgent

- Target version changed from CoCoALib-0.99850 to CoCoALib-0.99880