CoCoALib - Feature #1466

Ops += *= etc for Matrices?

21 Jun 2020 11:11 - John Abbott

Status:	Closed	**Start date:**	21 Jun 2020
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Improving	**Estimated time:**	0.00 hour
Target version:	CoCoALib-0.99850	**Spent time:**	2.00 hours

Description

Should we make "assign-op" operators for matrices?

History

#1 - 22 Jun 2020 10:26 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

It is not *necessary* to make these operators, but anyone used to C++ might expect them to work.

I suggest:

- make a simple impl -- if this is "short and sweet" then we accept it (costs little & causes no harm)
- the semantics are: M1 += M2 is **identical** to M1 = M1 + M2 etc. (indeed the impl could be this)

#2 - 26 Jun 2020 17:26 - John Abbott

- Status changed from In Progress to Resolved

- Assignee set to John Abbott

- % Done changed from 10 to 60

I have put in an obvious impl in **MatrixOps.H** (all defns are inline).
Everything compiles fine, but I have done no tests.
Also no doc (I'll write that after the tests).

#3 - 09 Nov 2020 21:13 - John Abbott

It is not clear to me how to implement += etc. better than the naive way, and maintain exception safety.

It might make some sense to make some **non-exception-safe** functions (if they can be usefully faster). There would have to be two separate interfaces, so it is clear whether the caller wants the exception-safe version or not.

#4 - 21 May 2021 16:43 - Anna Maria Bigatti

I think the syntax is worth having, and I think it should be exception safe, as for RingElems.

#5 - 21 May 2021 16:47 - John Abbott

Maybe we could get some benefit from using swap or std::move?
Investigate?

#6 - 31 May 2022 15:26 - John Abbott

- % Done changed from 60 to 70

As an example I showed this issue to my students, and we discussed some points in it.

I do not expect a non-exception-safe impl to be significantly faster: computing the new matrix entries will cost the same.
The main advantage is that less RAM will be needed (this may help with locality of reference?).
Conclusion: I suggest rejecting the idea of offering a non-exception-safe impl (unless someone can exhibit a genuine speed gain).

The naive impl (in comment 1) seems to be OK currently because matrices use ref counts, so the assignment
does not copy the matrix entries -- so there would be no speed gain by using swap instead of assignment.
However, if we should choose to stop using ref counts then (smart) swap would definitely be faster.
Conclusion? Maybe KISS? Leave the code as is until someone reports a problem?

#7 - 08 Aug 2022 20:01 - John Abbott

- Status changed from Resolved to Closed

- % Done changed from 70 to 100

As in previous comment it seems best to accept the KISS solution, and dedicate our efforts elsewhere (more important).