CoCoALib - Design #64

submat takes only vector<long>

15 Dec 2011 12:03 - Anna Maria Bigatti

Status:	Closed	**Start date:**	15 Dec 2011
Priority:	Normal	**Due date:**	
Assignee:	John Abbott	**% Done:**	100%
Category:	Improving	**Estimated time:**	2.06 hours
Target version:	CoCoALib-0.99850	**Spent time:**	2.05 hours

Description

```
submat(ConstMatrixView M,
       const std::vector<long>& rows,
       const vector<long>& cols)
```

Should we also allow vector<int> and other types?

Related issues:

Related to CoCoALib - Design #825: IsPositiveGrading -- really need 2 signatu...	**Closed**	**26 Nov 2015**
Related to CoCoALib - Feature #202: MatrixView/function for viewing a single ...	**Closed**	**04 Jul 2012**
Related to CoCoALib - Feature #312: LongRange(a,b) returning vector of long a...	**Closed**	**14 Feb 2013**
Related to CoCoALib - Feature #1788: New MatrixView/function "FirstRows/First...	**Closed**	**08 Mar 2024**

History

#1 - 21 Dec 2011 19:10 - John Abbott

submat(ConstMatView M, V1 rows, V2 cols)

Do we require that V1 and V2 are the same type, or can they be different types?
Could one have V1 of type @vector<int> and V2 of type vector<long> ?

If there are N different valid types for V1, and N for V2 then we risk having to offer N^2 signatures for submat.

#2 - 03 Feb 2012 15:18 - John Abbott

For functions which take single machine integer values as arguments, the coding guidelines say to use MachineInt which sorts out any possible risks arising from overflow and/or use of unsigned types.

However, it does not seem like a good idea to recommend using vector<MachineInt> when a collection of integer values is to be passed as a parameter.

Generally speaking for internal interfaces we have used long for passing machine integer values; however, in this case it would seem that int should be large enough for any practical purpose. The only likely problem with vector<long> is that many users may expect the type to be vector<int>. We could allow both vector<long> and vector<int>, but then someone might wonder why we excluded vector<T> for other integral types T... sigh.

#3 - 01 Aug 2014 08:59 - Anna Maria Bigatti

- Target version set to CoCoALib-1.0

#4 - 13 May 2015 09:58 - Redmine Admin

- Category set to Improving

#5 - 16 Dec 2015 11:42 - Anna Maria Bigatti

I don't think we should allow other integer types, after all we want to discourage other integer types.
One think I thought might be handy is having something like "all" as an argument:
submat(M, all, LongRange(0,2)) as a shortcut for LongRange(1, NumRows(M)).
This would add two signatures: for rows and for cols (meaningless with both!)
I think that would improve readability and coding speed (I always have to think of the arguments ;-).

#6 - 16 Dec 2015 18:11 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

Here are some other possible suggestions:

- submat(M, AllRows, ...) to select just certain columns
- submat(M, ..., AllCols) to select just certain rows
- submat(M, AllRows, AllCols) should this be allowed?
- FirstRows(M, 3) just a special case, but perhaps quite common?
- FirstCols(M, 3) (ditto)

One awkward aspect of AllRows and AllCols is that they must be of a type different from std::vector<long>, and this could be annoying if someone wants to write a function which accepts as args selectors for row/cols: to make this work also with AllRows there would have to be (at least) 2 different fn signatures.

#7 - 16 Dec 2015 18:23 - John Abbott

Scott Meyers advises making interfaces which are "easy to use correctly and hard to use incorrectly"; I think this is good advice.

If we use vector<long> for both row selectors and column selectors, it would not be too hard to make an erroneous call (with the row/col selectors exchanged).

We could consider creating two new types: *e.g.* RowSelect and ColSelect. These types can be constructed from vector<long> (and perhaps in other ways); the types could also represent internally AllRows and AllCols as special cases.

Objects of type RowSelect/ColSelect could also check that no rows are duplicated, and that all indices are non-negative... no sure about checking upper limits for the indices (somehow the upper limit must be indicated).

Right now I'm uncertain whether the extra effort of writing such new classes would be sufficiently repaid in terms of making it easier to write correct code (and to read that same code); the idea does seem to offer enticing gains in safety.

Comments? Criticisms?

#8 - 17 Dec 2015 14:33 - Anna Maria Bigatti

John Abbott wrote:

> Here are some other possible suggestions:

I think that's too many, and I think it's not worth the extra effort in separating rows and cols (actually, having AllRows would make me think I can write it in the position I want)

I think it could be handy to have just

- submat(M, all, ...) to select just certain columns
- submat(M, ..., all) to select just certain rows

.. but in fact in all CoCoALib (except AdjByDetOfMinors) we use submat only for doing

- FirstRows(M, n)

#9 - 17 Dec 2015 16:07 - John Abbott

If we were to adopt the idea of RowSelect and ColSelect then we could offer several signatures for submat:

- submat(M, rows, cols) same as we currently have
- submat(M, rows) to select certain rows
- submat(M, cols) to select certain cols
- submat(M, cols, rows) would be technically possible (but desirable???)

For instance a fn called FirstRows(n) could return a RowSelect object which selects just the first n rows; so we could have a call like submat(M, FirstRows(3)). JAA thinks this is not as readable as FirstRows(M,3), but there is the advantage that the only fn which makes submatrices is submat.

What do you think?

#10 - 17 Dec 2015 16:25 - Anna Maria Bigatti

John Abbott wrote:

> If we were to adopt the idea of RowSelect and ColSelect then we could offer several signatures for submat:...

I'm not convinced.
Apart from the extra work for doing it, I think it will be less readable.
Think of submat(M, RowSelect(VectorLong(2,3)), ColSelect(VectorLong(0,4)))

(is that the syntax you are thinking of?
Moreover a user might think he can swap RowSelect and ColSelect.

submat(M, FirstRows(3)).

I keep thinking submat(M, VectorLong(0,3), all) is more readable ;-)

#11 - 24 Sep 2019 10:20 - Anna Maria Bigatti

- Related to Design #825: IsPositiveGrading -- really need 2 signatures? added

#12 - 11 Jan 2021 11:07 - John Abbott

- Target version changed from CoCoALib-1.0 to CoCoALib-0.99850

#13 - 21 Jan 2024 20:01 - John Abbott

In comment 8 Anna says that practically all uses could be replaced by **FirstRows(M,n)**, so it makes a lot of sense to add this function (and probably **FirstCols** for symmetry).

I am unhappy about **all** because it is too short; or is there a way to achieve what we want without blocking the global name all? Maybe we could define a function void all(FunnyType), and then have signature submat(ConstMatrixView M, void (*all)(FunnyType), ...) Would this work? Or is it a "nasty C++ hack"?

Alternatively, we could have two more functions **SelectRows(M, v)** and **SelectCols(M,v)** instead of **all**.

#14 - 08 Mar 2024 09:15 - Anna Maria Bigatti

- Related to Feature #202: MatrixView/function for viewing a single row or column (RowMat, ColMat) added

#15 - 08 Mar 2024 09:16 - Anna Maria Bigatti

- Related to Feature #312: LongRange(a,b) returning vector of long a..b (included) added

#16 - 08 Mar 2024 10:02 - Anna Maria Bigatti

- Status changed from In Progress to Resolved

- % Done changed from 10 to 80

I went through all the calls of submat in CoCoALib and indeed most of them are just the first rows.
We could have another MatrixView class (called FirstRowsCols(M, long r, long c)?) whose implementation could be quite a bit simpler than submat.

For all the other cases, with the "new" function LongRange, we have done enough on this topic. So maybe I should "reject" this issue (opening the FirstRowsCols issue)

#17 - 08 Mar 2024 10:12 - Anna Maria Bigatti

- Related to Feature #1788: New MatrixView/function "FirstRows/FirstCols"? added

#18 - 08 Mar 2024 16:52 - Anna Maria Bigatti

- Tracker changed from Bug to Design

- % Done changed from 80 to 10

#19 - 08 Mar 2024 17:13 - Anna Maria Bigatti

This issue, as described, was rejected by answer 2 (#64#note-2).
For the other questions mentioned here, we have follow-ups in dedicated issues.
Closing.

#20 - 08 Mar 2024 17:17 - Anna Maria Bigatti

- Status changed from Resolved to Closed

- Assignee set to John Abbott

- % Done changed from 10 to 100

- Estimated time set to 2.06 h