Document Number: P 1910R0
Date: 2019-10-07
Authors: Michael Wong
Project: Programming Language C++, SG14 Games Dev/Low Latency/Financial Trading/Banking/Simulation/Embedded
Reply to: Michael Wong <michael@codeplay.com>

SG14: Linear Algebra SIG Meeting Minutes 2019/08/07-2019/10/02

Contents

Minutes for 2019/08/07 SG14 Conference Call

Bob Steagall
Mark Hoemann
Javier Cabezas
Jayesh Badwail
Klaus Iglberger
Mateusz Nowak
Matthew BUtler
Richard Dosselmann
William Tambellini
Michael Wong
Cem Bassoy
Nevin Liber
Graham Lopez
Jens Maurer

BS: discussion with Christian David Jayesh, Bob
had lunch for collaboration
working implementation mdspan and mdarray, and is aiming to create a test implementation
Guy and Bob feels it is a complimentary proposal
DOE proposal is a layer upon which Bob/Guy one can be
ship vehicle for Bob/Guy aiming for C++23
MH: Mark will want to aim for C++23
public github repo from DOE LA

MH: DOE proposal
LEWGi: come back after exploring concepts and ranges
instead of using concrete type
ranges and lazy evaluation
uncomfortable writing an expression template library
due to caution from Eigen
Mdspan is not owning
expression template hard sell to the committee due to valarray
valarray was abandoned by original author, when David proposed full expression template treatment for it,
P1674 talked about this
interest in lazy evaluation in ranges style, this can solve problems with arbitrary arithmetic evaluation witout multiple passes
this solves it for large class Blas-1, but still cannot fuse the matri and matrix multiply
even if we dont ship expression template library in the future
they build their expression template ontop of blas

KI: scale a vector with an expression template, so where do you set your boundary, where do you support view? yes we debated this a lot, we just want to introduce it and see where people go with it
major complaint to Blas is it takes too many arguments
KI:where you draw the line with transforming view: 0 layer is a drop in replacement for at least what existign Blas has, and not preclude that; but still need transform and scale to do what existign Blas can do
We were also thining of ways to reduce the number of arguments with BLAS
KI: But in your proposal I think there is one part where you were trying to do more. What part should we expand? scale with 2 positions ..; yes we have an alpha nad beta so I think it is there
KI: Is that for matrix vector? Yes, where you can have the plus y at the end. Yes I thought that was a bit of extending, but I like it; ok it is in GEMV
The only difference in functionality is that alpha nad beta means different
There would not be ABI problems in future as we are changign the accessor policy for mdspan
A triangular matrix is not an accessor

BS: Bob/Guy proposal:
returnig temporaries, expression templates, we are neutral on this, provdie an interface for implementator to decide
our interface can adopt for the future inclusing of expression templates,

MW: what games deveoper do - either proposal will work well to gamer
MW: will solver be part of the proposal: neither will support solver

BS: if you want customized operator, what is the complexity? If allowed this customization, they may not take the best advantage as possible
Andrew Lumsdaine mentioned that we shoudl think in terms of algorithms, concepts first, that we should not get ahead of the game
my thinking is to use the imperfect analogy of standard algorithm and containers, 1673 has wrapper aroudn blas which are algo, mdspan is iterator to provide access to data, matrix type holds the data
we created mdarray as a view cannot create a matrix, so mdarray, then we need a way to iterate over them
this is actually a good analogy
single vector type vs row and column vector, havng a seingle vector type is messy, from c++now and Klaus also mentioned this
wait until LEWG review to decide on this
KI: Klaus believes we can expresss thing better with 2 different vector types, this keeps it more generic, inner or outer allow people to do more things
also believe this helps people to debug their code more easily, at compile you can detect it and 2 vector types helps that
MH: when people do outer product, they are really aiming for a matrix

update,
so its not like you dont need it, but it is a different context for inner product
GL: What are the situation beside multiplication that you wan tto differentiate? expand is generic, row vector into a row matrix, or column to column matrix, 2 different functions is not generic
the more generic the operations are the better I can express the things I want to have
for multiplication and expand, could impose a cognitive overhead for most situation (other then these), so this could make it harder to decide which to use
also dividing the type into row and column does not cover the degeneracy, now you can describe 2 different kinds of multiplication but I can think of 5 different kinds of multiplication like permutation expand
GL: are you sure in future we dont need 2 different kinds of vectors? mathematically never need 2 types of vector, can always express with one type of vector
e.g. a dot product is different then an outer product? yes vectors are objects and you do operations on them, mathematically are not coupled ways, I am worry that in math we never talk about row and column vectors
MH: some people talk about covariance and contravariance?
GL: yes but that does
not say much about their orientation in space and in most cases I dont care about the type differentiation
BS: by imposing 2 different types, we use the language to impose a specification we want, its like shapes squares and rectangles
??: dont believe this is significant cognitive overload, people are already exposed to but at advanced level

JM: a slightly more abstract question, both proposals are forwarded to LEWG both libraries are not small, will they have enough expertise
games use 3-d vectors
B/G proposal: uses fixed sized can be hand coded SSD, while dynamically sized can be more BLAS like things
1673 has a lot of in out parameters so how is this passe thorugh vector, so a vector x scalar operation will be done in register
DOE proposal thinks can be done in mdarray, in/out as a reference parameter may not be current convention, a ref is a ptr to memory internally
What Jens is concerned how many thing you can return in register

MW: can we show same examples with both interface
or qualities comparison (though this could be dangerous)
not trying for a runoff but to support your claim that these are complimentary and could build on top of each other
mdspan can have more then 2 orders, while B/G proposal has submatrixes with

2 dimensions like a view
MH: left off batch linear algebra

JM: remember the valarray, afraid of having deadcode
1673 is in markdown, should be html or pdf
BS: Michael Parks has a system for creating standards paper, whch starts with markdown and change to tex and pdf
wg21 document creator

LM: 1673 is an updating matrix vector product, has 2 declarations shown before going on with requires, why? I forgot to delete the first declaration, the execution policy should not to be there
all uses of execution policy were for algo, now using matrices or vectors, can mdspan skip? yes it is polymorphic on the unique layout: mapping from 2 dim to 1 dim is 1-1
symmetric is an example as in if you want to multiply all with scalar

at the end of proposal is a reason for thin blas, this is important for floating point accuracy, where I use a class scalar template argument. But if I want a better guarantee, does it tell me it is a scalar type? dot can have intermediate storage, but does not say the evaluation accuracy we would like dot and dotc to use ...
If you go for vector and matrix then have a dot product taking a scalar, i.e.e separate dot product is a good thing: one function that does both in a single pass

thinks plain C++ wrapper around Blas may not deliver enough
is it really interesting to have the low level as well? Vendors like to have it this low, vendors want even lower
if this is just to add this bag of LA algorithm, is that still not C++ enough? Yes
yes Mathias Kretz was in SG14 on SIMD question and it may be orthogonal, then if all else fails then we need
not trying for tensor library
Lewgi also interested multidimensional ranges
see a lot of unconstraint template parameters: new library needs concepts: But LEWG has not settled on that yet, because once set they are hard to change.
These Algorithms take concrete types
Runtime parameter of this is bad, we actually have a layout called transpose view that abstract away so if you intend to call the BLAS that is what you have to do

Dot product has both conjugated and non conjugated, this is a good example where you should make a better effort to return things by value instead of scalar, similar to std:accumulate; but reduce is not order dependent

Minutes for 2019/09/04 SG14 Conference Call

04 Sep 2019

SG 14 linear algebra conference call

Present:

- Mark Hoemmen (scribe)
- Michael Wong
- 18655743073 (unidentified)
- Bob Steagall
- Jayesh Badwaik
- Klaus Igelberger
- Marco Foco
- Paul Preney
- William Tambellini
- Nevin Liber
- Matthew Butler

MW: Today I would mainly like to discuss

1. the possibility of integrating/merging P1385 and P1673, and

2. any implementation experience with the two proposals.

Also, on Wed 18 Sep. at CppCon, SG14 will meet. If you'll be there, let's discuss the linear algebra proposals. The idea is to give them more dissemination, esp. since there will be more gamers there.

Bob and Mark will be at CppCon and will have a slot on Wed. Matthew Butler will also be there. There will be other SG14 proposals, but I want to make sure linear algebra gets exposure, since it looks like it's a likely candidate for C++23.

BS: After the August linear algebra conference call, Mark and I had a conversation about integration and the layering of the two proposals. We are in complete agreement about how things should be layered and how to proceed. Mark says they will begin implementing headers and providing API, so we can begin including and using them in our implementation. That work will pick up the pace after CppCon and we'll have something to review at Belfast. Guy can't make it today and sends his regrets. Recent thought about minor change to improve interface; I might bring it up at CppCon and Belfast.

MH: We are working on implementing our proposal (P1673). Also, I had a phone conversation earlier this week with Andrew Lumsdaine to discuss our proposal. Andrew walked through some MTL3 (Matrix Template Library, version 3) interfaces and examples. His singular

value decomposition (SVD) example showed the value of nonowning "views" of parts of a matrix or vector. In general, I'm convinced that a usable library would need both owning and nonowning matrix and vector data structures. This is partly why we added mdarray (P1684).

BS: We have owning matrices -- could be dynamically allocated memory or part of built-in array -- as well as nonowning view types.

MW: I'm glad you talked to Andrew. For everyone: talk to John Mcfarlane about (discounted) SG14 tickets.

BS: If you read the reflector for SG21, John posted a no-cost registration link there.

MW: The next level of review your proposals will meet is LEWG directly. It's fine if the two proposals go as separate proposals, but anticipate that LEWG will ask why we need two different proposals. Also, to confirm: You're not thinking of a TS, right? You're targeting C++23?

BS: Yes.

MH: Yes.

MW: If you attempt integrating your two proposals, I applaud it, but I'm not trying to push you. If you do integrate, the resulting proposal will be large. Thus, try to have optional sections that you think are more controversial. This gives reviewers a "shopping list." Putting wording in might be premature. LWG committees will be doing a lot of reviews for C++20. We don't know what changes LWG would make to your proposal. The idea is to have something ready. Just be prepared for a lot of rework.

BS: I don't think we intend to have complete and final wording done, but we want to practice the wording and have tables that are expected, even though they might change.

MW: Let's lay out a schedule. If you optimistically get through LEWG in 2 sessions, then going to Prague already, before you get to LWG. Even in Prague, LWG will _still_ be busy. Best bet for LWG review is now Bulgaria. Worst case is Hawaii, a year after that, but that's golden. If you can get it in by Feb 2021, it's a shoe-in for 2023. Door closes for 2023 at beginning of 2022.

NL: I think 2 is optimistic.

MW: Good if you can do it in 5.

NL: Big paper takes longer to go through LWG, and they are not domain experts. This is like getting Ranges through.

MW: Average is 5-8, which means you'll be tight. You might not get everything in until end of 2022.

MH: I'm willing to be patient. The time scale for some of our codes is 40 years.

MW: I want this to happen, my company wants it, etc.

PP: Did you get my last e-mail, BS, about GCC 9 build?

BS: Yes, I got it, thanks, I will try it on a different system.

Minutes for 2019/10/02 SG14 Conference Call

Call cancelled.