SAND2019-1649C

P1417: Lessons learned for linear algebra library standardization

Mark Hoemmen (mhoemme@sandia.gov)

WG21, Kona, Feb. 2019

U.S. DEPARTMENT OF ENERGY NNSA National Nuclear Security Administration

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

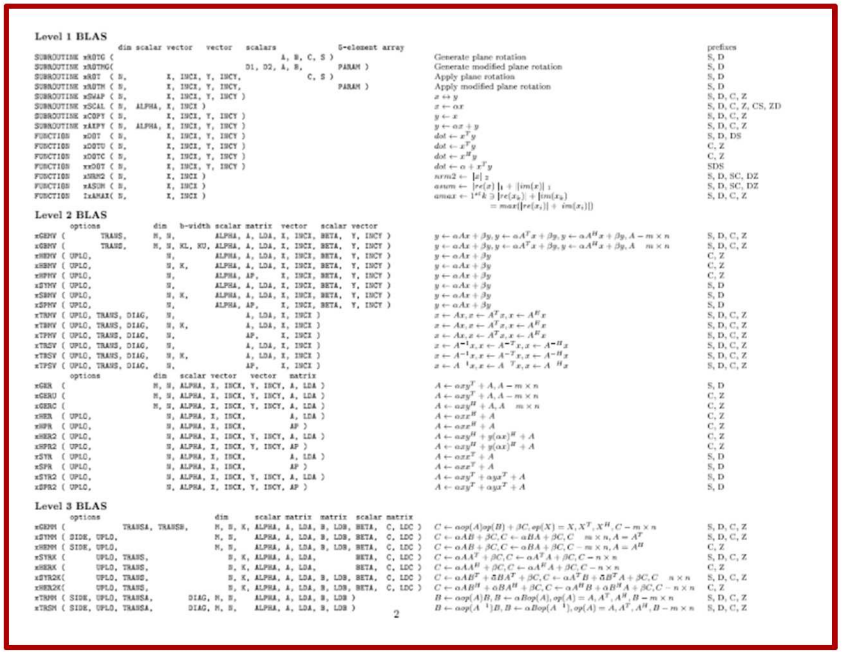
Outline

Sandia National Laboratories

- Over 40 years of standard linear algebra interfaces
- P1417 shares some history & lessons learned
  - Incomplete & can never be fully complete
  - Meant to grow & include your contributions
- This talk: Lessons from BLAS standardization
  - Standardize in layers, bottom up
  - Layer based on expertise & performance portability
  - Start w/ fundamentals least specific to linear algebra
- Lowest-level fundamentals include
  - Multidimensional arrays
  - Multidimensional parallel iteration
  - Explicit SIMD types & ops

Basic Linear Algebra Subprograms

- Standard published 2002
  - 1995 workshop
  - 1996-99 meetings 3x/year (w/ minutes online)
  - Fortran & C interfaces
  - Dense matrix & vector ops(*)
- Developed in levels (1,2,3):
  - Vector-vector (BLAS 1): 1979 (least data reuse)
  - Matrix-vector (BLAS 2): 1988 (constant factor more reuse)
  - Matrix-matrix (BLAS 3): 1990 (most data reuse)



Level 1 BLAS

	dim	scalar	vector	vector	scalars		5-element array		prefixes
SUBROUTINE xROTG (						A, B, C, S)		Generate plane rotation	S, D
SUBROUTINE xROTMG(					D1, D2, A, B,		PARAM)	Generate modified plane rotation	S, D
SUBROUTINE xROT (	N,		X, INCX,	Y, INCY,		C, S)		Apply plane rotation	S, D
SUBROUTINE xROTM (	N,		X, INCX,	Y, INCY,			PARAM)	Apply modified plane rotation	S, D
SUBROUTINE xSWAP (	N,		X, INCX,	Y, INCY)				$x \leftrightarrow y$	S, D, C, Z
SUBROUTINE xSCAL (	N,	ALPHA,	X, INCX)					$x \leftarrow \alpha x$	S, D, C, Z, CS, ZD
SUBROUTINE xCOPY (	N,		X, INCX,	Y, INCY)				$y \leftarrow x$	S, D, C, Z
SUBROUTINE xAXPY (	N,	ALPHA,	X, INCX,	Y, INCY)				$y \leftarrow \alpha x + y$	S, D, C, Z
FUNCTION xDOT (	N,		X, INCX,	Y, INCY)				$dot \leftarrow x^T y$	S, D, DS
FUNCTION xDOTU (	N,		X, INCX,	Y, INCY)				$dot \leftarrow x^T y$	C, Z
FUNCTION xDOTC (	N,		X, INCX,	Y, INCY)				$dot \leftarrow x^H y$	C, Z
FUNCTION xXDOT (	N,		X, INCX,	Y, INCY)				$dot \leftarrow \alpha + x^T y$	SDS
FUNCTION xNRM2 (	N,		X, INCX)					$nrm2 \leftarrow \|x\|_2$	S, D, SC, DZ
FUNCTION xASUM (	N,		X, INCX)					$asum \leftarrow \|re(x)\|_1 + \|im(x)\|_1$	S, D, SC, DZ
FUNCTION IxAMAX(	N,		X, INCX)					$amax \leftarrow 1^{st} k \ni \|re(x_k)\| + \|im(x_k)\| = \max(\|re(x_i)\| + \|im(x_i)\|)$	S, D, C, Z

Level 2 BLAS

	options	dim	b-width	scalar	matrix	vector	scalar	vector	
xGEMV (	TRANS,	M, N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y, y \leftarrow \alpha A^T x + \beta y, y \leftarrow \alpha A^H x + \beta y, A - m \times n$	S, D, C, Z
xGBMV (	TRANS,	M, N, KL, KU,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y, y \leftarrow \alpha A^T x + \beta y, y \leftarrow \alpha A^H x + \beta y, A \quad m \times n$	S, D, C, Z
xHEMV (	UPLO,	N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	C, Z
xHBMV (	UPLO,	N, K,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	C, Z
xHPMV (	UPLO,	N,		ALPHA,	AP,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	C, Z
xSYMV (	UPLO,	N,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	S, D
xSBMV (	UPLO,	N, K,		ALPHA,	A, LDA,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	S, D
xSPMV (	UPLO,	N,		ALPHA,	AP,	X, INCX,	BETA,	Y, INCY)	$y \leftarrow \alpha Ax + \beta y$	S, D
xTRMV (	UPLO, TRANS, DIAG,	N,			A, LDA,	X, INCX)			$x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$	S, D, C, Z
xTBMV (	UPLO, TRANS, DIAG,	N, K,			A, LDA,	X, INCX)			$x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$	S, D, C, Z
xTPMV (	UPLO, TRANS, DIAG,	N,			AP,	X, INCX)			$x \leftarrow Ax, x \leftarrow A^T x, x \leftarrow A^H x$	S, D, C, Z
xTRSV (	UPLO, TRANS, DIAG,	N,			A, LDA,	X, INCX)			$x \leftarrow A^{-1} x, x \leftarrow A^{-T} x, x \leftarrow A^{-H} x$	S, D, C, Z
xTBSV (	UPLO, TRANS, DIAG,	N, K,			A, LDA,	X, INCX)			$x \leftarrow A^{-1} x, x \leftarrow A^{-T} x, x \leftarrow A^{-H} x$	S, D, C, Z
xTPSV (	UPLO, TRANS, DIAG,	N,			AP,	X, INCX)			$x \leftarrow A^{-1} x, x \leftarrow A^{-T} x, x \leftarrow A^{-H} x$	S, D, C, Z

	options	dim	scalar	vector	vector	matrix	
xGER (		M, N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)	$A \leftarrow \alpha xy^T + A, A - m \times n$	S, D
xGERU (		M, N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)	$A \leftarrow \alpha xy^T + A, A - m \times n$	C, Z
xGERC (		M, N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)	$A \leftarrow \alpha xy^H + A, A \quad m \times n$	C, Z
xHER (	UPLO,	N,	ALPHA,	X, INCX,		A, LDA)	$A \leftarrow \alpha xx^H + A$	C, Z
xHPR (	UPLO,	N,	ALPHA,	X, INCX,		AP)	$A \leftarrow \alpha xx^H + A$	C, Z
xHER2 (	UPLO,	N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)	$A \leftarrow \alpha xy^H + y(\alpha x)^H + A$	C, Z
xHPR2 (	UPLO,	N,	ALPHA,	X, INCX,	Y, INCY,	AP)	$A \leftarrow \alpha xy^H + y(\alpha x)^H + A$	C, Z
xSYR (	UPLO,	N,	ALPHA,	X, INCX,		A, LDA)	$A \leftarrow \alpha xx^T + A$	S, D
xSPR (	UPLO,	N,	ALPHA,	X, INCX,		AP)	$A \leftarrow \alpha xx^T + A$	S, D
xSYR2 (	UPLO,	N,	ALPHA,	X, INCX,	Y, INCY,	A, LDA)	$A \leftarrow \alpha xy^T + \alpha yx^T + A$	S, D
xSPR2 (	UPLO,	N,	ALPHA,	X, INCX,	Y, INCY,	AP)	$A \leftarrow \alpha xy^T + \alpha yx^T + A$	S, D

Level 3 BLAS

	options		dim		scalar	matrix	matrix	scalar	matrix	
xGEMM (	TRANSA,	TRANSB,	M, N, K,		ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)	$C \leftarrow \alpha op(A) op(B) + \beta C, op(X) = X, X^T, X^H, C - m \times n$	S, D, C, Z
xSYMM (	SIDE, UPLO,		M, N,		ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)	$C \leftarrow \alpha AB + \beta C, C \leftarrow \alpha BA + \beta C, C \quad m \times n, A = A^T$	S, D, C, Z
xHEMM (	SIDE, UPLO,		M, N,		ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)	$C \leftarrow \alpha AB + \beta C, C \leftarrow \alpha BA + \beta C, C - m \times n, A = A^H$	C, Z
xSYRK (	UPLO,	TRANS,		N, K,	ALPHA,	A, LDA,		BETA,	C, LDC)	$C \leftarrow \alpha AA^T + \beta C, C \leftarrow \alpha A^T A + \beta C, C - n \times n$	S, D, C, Z
xHERK (	UPLO,	TRANS,		N, K,	ALPHA,	A, LDA,		BETA,	C, LDC)	$C \leftarrow \alpha AA^H + \beta C, C \leftarrow \alpha A^H A + \beta C, C - n \times n$	C, Z
xSYR2K(	UPLO,	TRANS,		N, K,	ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)	$C \leftarrow \alpha AB^T + \beta BA^T + \beta C, C \leftarrow \alpha A^T B + \beta B^T A + \beta C, C \quad n \times n$	S, D, C, Z
xHER2K(	UPLO,	TRANS,		N, K,	ALPHA,	A, LDA,	B, LDB,	BETA,	C, LDC)	$C \leftarrow \alpha AB^H + \alpha BA^H + \beta C, C \leftarrow \alpha A^H B + \alpha B^H A + \beta C, C - n \times n$	C, Z
xTRMM (	SIDE, UPLO, TRANSA,	DIAG, M, N,			ALPHA,	A, LDA,	B, LDB)			$B \leftarrow \alpha op(A)B, B \leftarrow \alpha Bop(A), op(A) = A, A^T, A^H, B - m \times n$	S, D, C, Z
xTRSM (	SIDE, UPLO, TRANSA,	DIAG, M, N,			ALPHA,	A, LDA,	B, LDB)			$B \leftarrow \alpha op(A^{-1})B, B \leftarrow \alpha Bop(A^{-1}), op(A) = A, A^T, A^H, B - m \times n$	S, D, C, Z

2

(Fortran) BLAS quick reference:
http://www.netlib.org/blas
(See also Jack Dongarra's oral history)

(*) Sparse etc. too; ask me later

BLAS 1-3 coevolved w/ computers

Seymour Cray w/ Cray 1, circa 1976 (when LINPACK funding started)

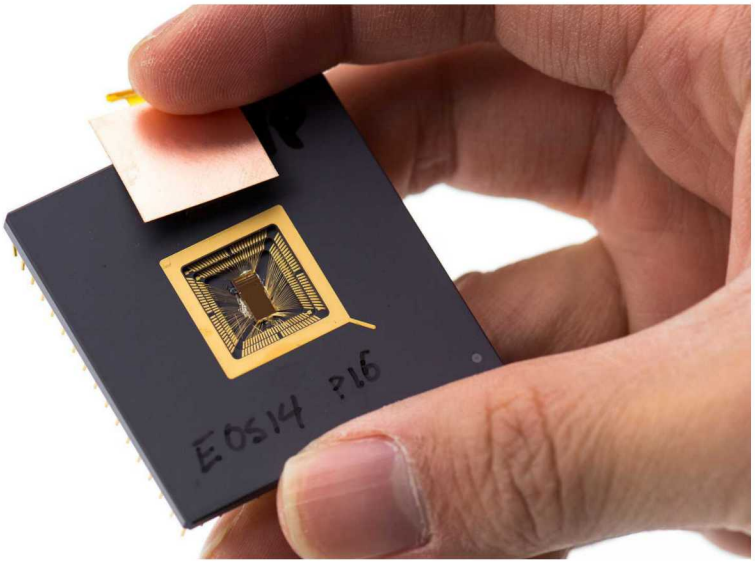
Vector (Cray, NEC)

- Low flop/byte ratio
- Favor long, dependence-free loops & regular data access
- BLAS 1 target

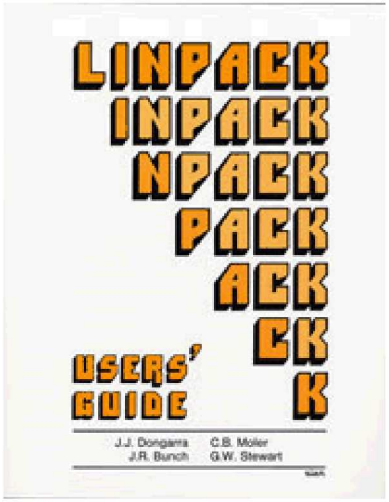
Cache based

- "Killer micros" (see 1991 NYT article)
- High flop/byte ratio
- Favor data reuse
- BLAS 3 target

Yunsup Lee holding RISC V prototype, 2013

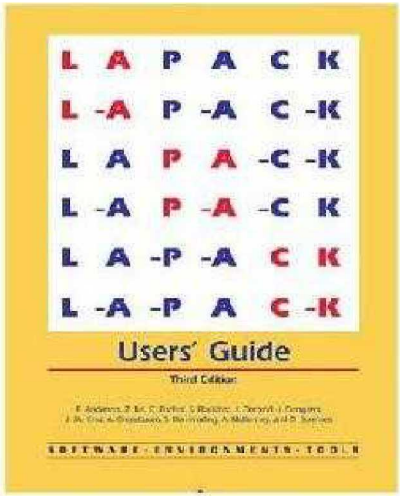


BLAS codesigned w/ algorithms



- LINPACK library: 1979
  - Linear systems (LU, Cholesky)
  - Linear least squares (QR)
  - General dense, symmetric, & banded
  - Designed to use BLAS (1), for good performance on many different computers

- LAPACK: 1990
  - Combines functionality of LINPACK + EISPACK ({eigen,singular} value problems)
  - "Coreleased" w/ BLAS 3, w/ common authors
  - Algorithms that better exploit data reuse
  - BLAS 3 designed for those algorithms



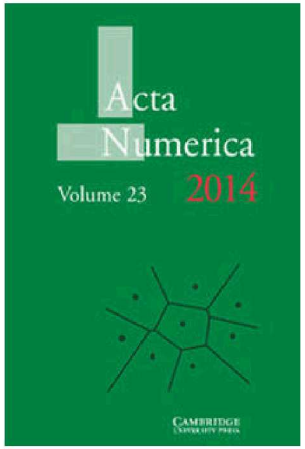
Imitate BLAS' std-ization approach

Sandia National Laboratories

- How it got standardized, not necessarily the interface itself
  - I do not necessarily advocate a "C++ BLAS binding"
  - Fullest development of Ranges may reduce BLAS 1's value
  - BLAS optimized for single large problems, not small & many
  - Larger, different customer base, solving different problems
- Process: Standardize in layers, bottom up
- Layer by expertise
  - BLAS: hardware / performance; LAPACK: numerical analysis
  - Kazushige Goto (talented BLAS optimizer) is self-taught [1]
  - Numerical analysis not typical background for Std Lib developers
- Layer for performance portability
  - Value added in multiple implementations, tuned for specific hardware

Counter & rebuttal

- Counter: BLAS & LAPACK codesigned by numerical analysts
  - Will future algorithms need different fundamental operations?
  - Would a bottom-up approach risk overspecializing the lowest level?
  - Radical changes in computer architecture?
- Rebuttal
  - Fundamentals not specific to linear algebra
  - Last 2 decades: Many theoretical results on algorithm optimality (matrix & tensor), so unlikely to need radical changes like BLAS 1-3
  - Participation by more communities (graphics, machine learning, embedded, …) will reduce risk of overspecification
  - Many of us plugged into SG1 etc., so we're watching architecture changes



See "Communication lower bounds & optimal algorithms for numerical linear algebra," Ballard, Carson, Demmel, Hoemmen, Knight, & Schwarz, 2014

Alternate procedural approaches

Sandia National Laboratories

- What if WG21 rejects / delays standardization?
- Join the BLAS Technical Forum?
  - (+) Meeting minutes show they planned a C++ interface [2]
  - (o) C++ & Java excluded: “time constraints” & possible Java changes
  - (+) CBLAS exists & has support from multiple vendors
  - (-) Invading small (< 20) focused committee, inactive nearly 20 years
  - (-) May reject C++ “binding” if too different (see MPI C++ binding)
- Form our own standard?
  - GraphBLAS: Express graph algorithms in the language of linear algebra
  - Batched BLAS: Do many small dense ops at once, expose parallelism
  - Non-language standards more useful when it’s worth having multiple implementations of the same interface
    - Multiple Engines / back-ends permit this, also encourage interoperability

Fundamental or batteries included?

Sandia National Laboratories

- Referencing debate over Graphics API [3,4]
  - For this talk, I'm not taking a side
  - Fundamental-ness may influence WG21 prioritization
  - But: just as BLAS was codesigned w/ LINPACK & LAPACK, we should design the linear algebra we want, & standardize what we can
- Lowest level of linear algebra: Clearly fundamental, including
  - Multidimensional arrays: P0009r9, mdspan, at LWG level
  - Multidimensional parallel iteration: Papers & ideas by authors here
  - Explicit SIMD types & ops: Voted into N4744 (Parallelism TS 2)
- What makes them fundamental?
  - Need / would benefit from compiler support
  - Vocabulary
  - Multiple use cases outside linear algebra

Multidimensional arrays: mdspan

Sandia National Laboratories

- P0009r9, currently in LWG wording review
- Christian Trott will present on this
- Meant as a zero-overhead abstraction
  - Like Fortran arrays, complete w/ slices
  - Can mix compile- & run-time dimensions
  - No implicit temporaries / allocation (9x)
  - Compiler could optimize indexing
- Polymorphic data layout
  - Needed for performance with tensors & batched small dense
  - Performance portability (tune layout for architecture)
  - Compatibility with different libraries in other languages

Is mdspan a “matrix”?

Sandia National Laboratories

- mdspan is nonowning
  - Generalizes span [views]
  - Can’t create (allocate) matrix
  - Expressions may need temps
  - We use owning Kokkos::View in BLAS-like kokkos-kernels
- mdspan views memory
  - For small dense SIMD-based classes, may prefer values
  - Ask us about combining Kokkos::View & SIMD types
- Not a linear algebra library
  - No NumPy-style broadcast
  - No expressions, arithmetic, …

If we want “Matlab in C++”:

- Owning tensor / mat / vec classes
- Dispatch to mdspan kernels
- mdspan may support expression templates through AccessorPolicy

Multidimensional parallel iteration

Sandia National Laboratories

- Who needs this?
  - Linear & (especially) tensor algebra
  - Representations of physical space (structured grids)
- Why not just 1-D iterators?
  - Iterating 1 dim at a time ignores data locality
  - Problems most naturally expressed in multiple dimensions; "flattening" the iteration space by hand complicates code
  - All these must be optimized together:
    - Algorithm & data layout (hence mdspan's polymorphic layout)
    - Iteration order & parallelization strategy
- Interface options
  - For-each i… over multidimensional index domain (ask us)
  - Einstein tensor index notation (ask other folks here)

Conclusions

Sandia National Laboratories

- If we want to standardize a C++ linear algebra library,
- we should learn from BLAS Standard's successes:
  - Standardize in layers, from the bottom up
  - Layer by expertise: hardware / performance vs. numerical analysis
- Identify fundamentals for any linear algebra / tensor library
  - Explicit SIMD types & ops: Voted into N4744 (Parallelism TS 2)
  - Multidimensional arrays: P0009r9, in LWG wording review
  - Multidimensional parallel iteration: ????, OPPORTUNITY
- Thanks to P1417 coauthors & all of you!

References

- [1] John Markoff, “Writing the Fastest Code, by Hand, for Fun: A Human Computer Keeps Speeding Up Chips,” *New York Times*, Nov. 28, 2005.
- [2] BLAST Forum MINUTES, NIST, Washington, D.C., Oct. 8-9, 1998. Available online: http://www.netlib.org/blas/blast-forum/blast-forum-minutes.oct8-9.98.html [last accessed Feb. 14, 2019].
- [3] Guy Davidson, “Batteries not included: what should go in the C++ standard library?”, *World of hatcat*, Feb. 16, 2018. Available online: https://hatcat.com/?p=16 [last accessed Feb. 10, 2019].
- [4] Titus Winters, “What Should Go Into the C++ Standard Library,” *Abseil Blog*, Feb. 27, 2018. Available online: https://abseil.io/blog/20180227-what-should-go-stdlib [last accessed Feb. 10, 2019].