A Proposal for a Next-Generation BLAS

E. Jason Riedy\textsuperscript{1}, Greg Henry\textsuperscript{2}, James Demmel\textsuperscript{3}, Mark Gates\textsuperscript{4}, Xiaoye S. Li\textsuperscript{5}, Ping Tak P. Tang\textsuperscript{2}

1: GT, 2: Intel, 3: UCB, 4: UTK, 5: LBNL
Why? (aka Goals)

- XBLAS has 287 routines. Optimizing all of them is not a popular option.
  - Also missing some functionality, e.g. supporting parallel composition, lower precision.
- Establish a uniform naming convention at the library symbol level.
  - Low-level naming, others can provide high-level interfaces.
  - Mixed precision, extended precision, reproducibility, batched, fixed-point…
  - Ensure parallel routines can be built on top!
- Provide clear platform optimization targets and minimal needed interface.
  - Libraries need not support all possible names.
  - Many routines can be built on a few tuned microkernels (as in BLIS).
- (Eventually) Provide a reference implementation.
  - Also, example higher level C++ & Fortran interfaces.

We invite feedback: [https://goo.gl/D1UKnw](https://goo.gl/D1UKnw)
Naming Rules

Provide the low-level *naming* for high-level languages and dispatching.

\[
\text{BLAS}_{<\text{blasFunction}>_{<\text{typeSequence}>[_{<\text{precisionLength}>[<\text{multiplier}>]}]}_{<\text{suppl}>}}(\ldots \text{parameter list} \ldots)
\]

- \(<\text{blasFunction}> := \text{dot} | \text{gemm} | \text{gemm\_out} | \text{gemm\_batch} | \text{trsv} | \ldots\)
- \(<\text{typeSequence}> := <\text{type}> | <\text{typeSequence}>\)
- \(<\text{type}> := <\text{mathType}><\text{precisionLength}>[<\text{multiplier}>]\)
- \(<\text{multiplier}> := x2 | \text{Repro3} | \ldots\)
- \(<\text{mathType}> := C | R | I | \ldots\)
- \(<\text{precisionLength}> := 8 | 16 | 32 | 64 | 80 | \ldots\)
- \(<\text{suppl}> := \text{Repro3} | \ldots\)
Naming Examples

- **Traditional:**
  - DGEMM(...) => BLAS_GEMM_R64(...)
  - (The original BLAS names will remain for compatibility.)

- **CUDA’s cuBLAS:**
  - cublasGemmEx(..., CUDA_R_16F, ..., CUDA_R_32F, ..., CUDA_R_32F, ..., CUDA_R_64F, ...);

- **Our implementation of cublasGemmEx:**
  - Could dispatch to BLAS_GEMM_R16R32R32_64
  - … if it exists …
  - … taking 16-bit real A, 32-bit real B & C …
  - … using 64-bit internal precision.
  - Scalars would be 32-bit reals. (Although CUDA’s would be 64-bit?)

We invite feedback: https://goo.gl/D1UKnw
Reproducibility & Mixed-Extended Precision

- **Reproducibility:**
  - Compute bitwise identical answers independent of summation order / hardware resources.
  - Essential for some massive-scale computing applications that require computing the same result no matter the machine configuration.

- **Cost (ReproBLAS version):**
  - Summing $n$ numbers reproducibly costs $7n$ flops, uses a 6 word "reproducible accumulator".
  - Could be accelerated by the following operations...

- **IEEE-754 is recommending augmented arithmetic operations.**
  - New names for twoSum $(a + b \Rightarrow h, t)$, twoProd.
  - Supports both ReproBLAS and "double double" style arithmetic.

We invite feedback: https://goo.gl/D1UKnw
Remaining Big, Big Question

- Many little things have been added. Incomplete list:
  - _OUT routines that do not overwrite matrix C, useful for PBLAS.
  - Specify minimal required support for extended precision, reproducibility.
  - Type inference rules for the scalars.
  - Consistent exceptional value handling.
  - Error reporting through a return value.
- One huge question remains:

  **Should level 2 & 3 BLAS support row strides?**

  - Row strides support row- & column-major orders, packed double-double.
  - Could optimize dense tensor slices…

We invite feedback: [https://goo.gl/D1UKnw](https://goo.gl/D1UKnw)