A Proposal for a Next-Generation BLAS

E. Jason Riedy[1], Greg Henry[2], James Demmel[3], Mark Gates[4], Xiaoye S. Li[5], Ping Tak P. Tang[2]

1: GT, 2: Intel, 3: UCB, 4: UTK, 5: LBNL

We invite feedback: https://goo.gl/D1UKnw

Images from Wikipedia

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.

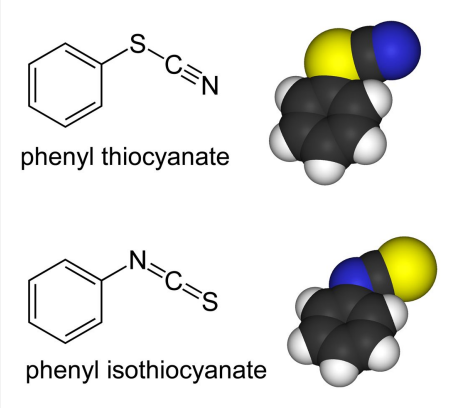
We invite feedback: https://goo.gl/D1UKnw

Naming Rules

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

```
BLAS_<blasFunction>_<typeSequence>[_<precisionLength>[<multiplier>]][_<suppl>](
...parameter list...)
```

- `<blasFunction> := dot | gemm | gemm_out | gemm_batch | trsv | …`
- `<typeSequence> := <type> | <typeSequence>`
- `<type> := <mathType><precisionLength>[<multiplier>]`
- `<multiplier> := x2 | Repro3 | …`
- `<mathType> := C | R | I | …`
- `<precisionLength> := 8 | 16 | 32 | 64 | 80 | …`
- `<suppl> := Repro3 | …`



We invite feedback: https://goo.gl/D1UKnw

Image from Wikipedia

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 7*n* 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.

2011 Atlanta Snowmageddon

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

Image from Brickset on Flickr