Exception handling for the BLAS and LAPACK

Weslley da Silva Pereira, University of Colorado Denver

September 14, 2022

XBLAS meeting

Coauthors [1]

- Jim Demmel, UC Berkeley,
- Jack Dongarra, U Tennessee, Knoxville,
- Mark Gates, U Tennessee, Knoxville,
- Greg Henry, Intel Corp.,
- Julien Langou, U Colorado, Denver,
- Xiaoye Li, LBNL,
- Piotr Luszczek, U Tennessee, Knoxville,
- Weslley Pereira, U Colorado, Denver,
- Jason Riedy, Lucata Corp.,
- Cindy Rubio-Gonzalez, UC Davis.

[1] Proposed Exception Handling for the BLAS and LAPACK: `https://www.arxiv.org/abs/2207.09281`

Examples of bad exception handling

- Ariane 501, 1996. "[An] exception was caused during execution of a data conversion from 64-bit floating point to 16-bit signed integer value." [2]
- Roborace, 2020. "the system somehow managed to produce a NaN (not a number) value and all verification logic was designed to work only with numbers." [3]

[2] https://www-users.cse.umn.edu/~arnold/disasters/ariane.html

[3] https://www.reddit.com/r/formula1/comments/jk9jrg/comment/gai2951

Agenda

1. Inconsistencies in current BLAS and LAPACK libraries
2. Proposed exception handling and preliminary tests
3. Concluding remarks

One of the goals of this talk:
Seek feedback

Inconsistencies in the BLAS

IxAMAX: Return the index of the largest entry.

- Inconsistency #1:
  `isamax([ 0, NaN, 2 ]) = 3`,
  `isamax([ NaN, 0, 2 ]) = 1`,
- Inconsistency #2:
  `icamax([ OV+i*OV, Inf+i*0 ]) = 1`,
  `icamax([ .6*OV+i*.6*OV, .7*OV+i*.7*OV ]) = 1`,
  where `OV` is the overflow threshold.
- Proposed consistent fix:

Return index to the first NaN,
else to the first Inf,
else to the first largest finite entry.

Inconsistencies in the BLAS

xGER: $A := A + \alpha x y^T$.

- Current Standard BLAS:
  If $\alpha = 0$, return A.
  If $y_j = 0$ then $A_{ij} := A_{ij}$.
  There is no check for $x_i = 0$.
- Proposed consistent fix:

  Keep check for $\alpha = 0$ but not for $y_j = 0$ (or $x_i = 0$).

Inconsistencies in LAPACK

xGETF2 + xGETRS: Solve $Ax = b$ for x.

- Example: $A = \begin{bmatrix} 1 & 0 \\ NaN & 2 \end{bmatrix}$ and $b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.
- Combination of BLAS inconsistencies:
  - `IxAMAX` chooses $A_{11} = 1$, not $A_{21} = NaN$, as pivot.

$$A = \begin{bmatrix} 1 & 0 \\ NaN & 2 \end{bmatrix}$$

Inconsistencies in LAPACK

xGETF2 + xGETRS: Solve $Ax = b$ for x.

- Example: $A = \begin{bmatrix} 1 & 0 \\ NaN & 2 \end{bmatrix}$ and $b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.
- Combination of BLAS inconsistencies:
  - `IxAMAX` chooses $A_{11} = 1$, not $A_{21} = NaN$, as pivot.
  - `xGER` finds 0 in A_{12} and do not use the NaN.

$$A_{22} := A_{22} - NaN \cdot 0$$

Inconsistencies in LAPACK

xGETF2 + xGETRS: Solve $Ax = b$ for x.

- Example: $A = \begin{bmatrix} 1 & 0 \\ NaN & 2 \end{bmatrix}$ and $b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.
- Combination of BLAS inconsistencies:
  - `IxAMAX` chooses $A_{11} = 1$, not $A_{21} = NaN$, as pivot.
  - `xGER` finds 0 in A_{12} and do not use the NaN.
  - `xGETRS` sees the LU factorization:

$$L = \begin{bmatrix} 1 & 0 \\ NaN & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

Inconsistencies in LAPACK

xGETF2 + xGETRS: Solve $Ax = b$ for x.

- Example: $A = \begin{bmatrix} 1 & 0 \\ NaN & 2 \end{bmatrix}$ and $b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.
- Combination of BLAS inconsistencies:
  - `IxAMAX` chooses $A_{11} = 1$, not $A_{21} = NaN$, as pivot.
  - `xGER` finds 0 in A_{12} and do not use the NaN.
  - `xGETRS` sees the LU factorization:

$$L = \begin{bmatrix} 1 & 0 \\ NaN & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

  - The first call for `xTRSV` solves $Ly = b$ and finds $y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$. In the second iteration, $y_2 := 1 - NaN \cdot y_1$ and $y_1 = 0$.

Inconsistencies in LAPACK

xGETF2 + xGETRS: Solve $Ax = b$ for x.

- Example: $A = \begin{bmatrix} 1 & 0 \\ NaN & 2 \end{bmatrix}$ and $b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.
- Combination of BLAS inconsistencies:
  - `IxAMAX` chooses $A_{11} = 1$, not $A_{21} = NaN$, as pivot.
  - `xGER` finds 0 in A_{12} and do not use the NaN.
  - `xGETRS` sees the LU factorization:

$$L = \begin{bmatrix} 1 & 0 \\ NaN & 1 \end{bmatrix}, \quad U = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$$

  - The first call for `xTRSV` solves $Ly = b$ and finds $y = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.
  - The final solution is $x = \begin{bmatrix} 0 \\ 0.5 \end{bmatrix}$
- **NaN does not propagate!**

Examples of unavoidable inconsistencies

IEEE 754-2019 defines `min` and `max` that propagate NaNs.
Compilers need time to adopt those rules. [4]

```
min( qNaN, 1.0 ) = qNaN ! gfortran 12.2
min( 1.0, qNaN ) =  1.0 ! gfortran 12.2
```

Absolute value, product, and quotient of complex numbers:

- They are not specified by IEEE 754-2019.
- There is space for inconsistencies, e.g., [5]

```
cmplx(Inf,0.0)*cmplx(Inf,Inf) = cmplx(Inf,Inf) ! ifort 2021
cmplx(Inf,0.0)*cmplx(Inf,Inf) = cmplx(NaN,NaN) ! gfortran 12

abs(subNormal)              = subNormal ! ifort 2021
abs(cmplx(0.0,subNormal)) = subNormal ! ifort 2021 (-O0)
abs(cmplx(0.0,subNormal)) = 0.0       ! ifort 2021 (-O3)
```

[4] https://www.godbolt.org/z/1sfceYbao
[5] https://www.godbolt.org/z/nvWWeToY6

Proposed exception handling

If NaNs or Infs are inputs or created internally,

1. The program will still terminate.
2. Moreover, either:
   - NaNs and Infs propagate to the output.
   - NaNs and Infs are dealt with internally.
   - NaNs and Infs **do not propagate** to the output **in special cases**, e.g., $C := 0 \cdot AB + 0 \cdot C$ in GEMM.
3. LAPACK routines will report (using INFO and more).

Current design [6] takes into account:

- Several user requests.
- Different stakeholders.
- Relevant xSDK Community Policies for Library Development.

[6] `https://www.arxiv.org/abs/2207.09281`

The EC (Error Checking) interfaces

- Example:
  `SGESV_EC( N, NRHS, A, LDA, IPIV, B, LDB, INFO, FLAG_REPORT, INFO_ARRAY, CONTEXT )`
- 3 more arguments at end:
  - FLAG_REPORT(1:2)
    FLAG_REPORT(1): WHAT to report.
    FLAG_REPORT(2): HOW to report.
  - INFO_ARRAY(:) - can report INFO + details on Infs and NaNs on input, and internal calls.
  - CONTEXT - opaque argument. Can report, for example, exceptions on different threads.
- Legacy interface will be maintained as a wrapper.

Some early testBLAS [8] results

Expected output from IxAMAX: Return index to the first NaN, else to the first Inf, else to the first largest finite entry.

BLAS library	I{S,D}AMAX	I{C,Z}AMAX
Apple Accelerate 12.2.1	DNFPS [7]	DNFPS
BLIS 0.9.0	Pass	DNFPS
IBM ESSL 6.3.0	DNFPS	DNFPS
Intel MKL 2022.1.0	Pass	DNFPS
LAPACK 3.9.1	DNFPS	DNFPS
LAPACK 3.10.1	DNFPS	DNFPS
LAPACK 3.11-beta	Pass	Pass
OpenBLAS 0.3.8	DNFPS	DNFPS

[7] DNFPS: Does not follow proposed standard.

[7] Apple Accelerate on macOS Monterey 12.2.1 using Apple clang version 13.1.6. IBM ESSL on Summit node using GNU compiler v9.1.0. Others on Ubuntu 20.04.4 LTS using GNU compiler v9.4.0.

[8] `https://www.github.com/tlapack/testBLAS`

Some early testBLAS [10] results

Expected output from xNRM2: NaN if in input, else Inf if in input, else "accurate answer" (possibly Inf).

BLAS library	{S,D}NRM2 (finite input)	{S,D}NRM2
Apple Accelerate 12.2.1	DNFPS [9]	DNFPS
BLIS 0.9.0	DNFPS	DNFPS
IBM ESSL 6.3.0	Pass	Pass
Intel MKL 2022.1.0	Pass	Pass
LAPACK 3.9.1	DNFPS	DNFPS
LAPACK 3.10.1	Pass	Pass
LAPACK 3.11-beta	Pass	Pass
OpenBLAS 0.3.8	Pass	Pass

[9] DNFPS: Does not follow proposed standard.

[9] Apple Accelerate on macOS Monterey 12.2.1 using Apple clang version 13.1.6. IBM ESSL on Summit node using GNU compiler v9.1.0. Others on Ubuntu 20.04.4 LTS using GNU compiler v9.4.0.

[10] `https://www.github.com/tlapack/testBLAS`

Proposed tasks

1. During installation of LAPACK, check `abs`, multiplication and division of complex, and `min` and `max` behave as expected. (partially done in LAPACK 3.10.1)
2. Modify LAPACKE drivers, so that they return an error flag if input has NaNs or Infs.
3. Modify LAPACK, so that routines that compute norms of complex matrices signal NaNs and Infs on input.
4. Fix Reference BLAS and implement test code.
5. Validate the EC interfaces for a few LAPACK routines.
6. Design more general test code.
7. Implement the EC interfaces to the rest of LAPACK.

Thank you!

Questions?

Proposed Exception Handling for the BLAS and LAPACK:
`https://www.arxiv.org/abs/2207.09281`

Funded by:
NSF under the project Basic ALgebra LIbraries for Sustainable Technology with Interdisciplinary Collaboration (BALLISTIC).

National
Science
Foundation

Meanings of WHAT and HOW

- WHAT $\leq$ -1: check nothing.
- WHAT $=$ 0: legacy INFO checking.
- WHAT $=$ 1: also, check input/output args for Infs/NaNs.
- WHAT $\geq$ 2: do the above throughout call tree.

- HOW $\leq$ 0: only use INFO.
- HOW $=$ 1: also use INFO_ARRAY.
- HOW $=$ 2: also, if INFO $\neq$ 0, call
  `REPORT_EXCEPTIONS ( CONTEXT, ROUTINE_NAME, INFO_ARRAY)`
- HOW $=$ 3: do the above throughout call tree.
- HOW $\geq$ 4: call
  `GET_FLAGS_TO_REPORT( CONTEXT, FLAG_REPORT )`

testBLAS [13]

Tests corner cases and Inf/NaN propagation on BLAS packages.

- Uses the C++ BLAS interface of ⟨T⟩LAPACK [11].
- Uses BLAS++ [12] wrappers as main interface to vendor BLAS.
- Currently:
  - Test corner cases on BLAS routines.
  - Specific Inf and NaN propagation tests for `iamax`, `nrm2`, `trsv` and `trsm`.

[11] `https://github.com/tlapack/tlapack`
[12] `https://bitbucket.org/icl/blaspp`
[13] `https://www.github.com/tlapack/testBLAS`