Improving Exception Handling in the BLAS and LAPACK

Jim Demmel, Jack Dongarra, Mark Gates,
Greg Henry, Xiaoye Li, Jason Riedy,
Weslley Pereira, Julien Langou, Piotr Luszczek

Examples of handling exceptions badly

https://www.reddit.com/r/formula1/comments/jk9jrg/ot_roborace_driverless_racecar_drives_straight/gai295l/

"During this initialization lap something happened which apparently cause the steering control signal to go to NaN"

Outline

- Basic high level goals: handle exceptions "consistently"
- Goals for BLAS
- Goals for LAPACK
- Comments welcome!

High level goals

- Handle exceptions “consistently”
  - Always terminate, despite exceptions
  - Report exceptions for which problem “ill-posed”
    - Ex: eig(NaN), not inv(Inf)
  - Propagate exceptions “consistently” (see later examples)
- Do not change default interface semantics
  - Allow options for more detailed reporting
    - NaN and/or Inf appear in inputs, outputs, or internally
- Do not slow down (much)
- Accommodate inconsistent building blocks
  - Eg: How compilers implement max or complex/complex, how vendors optimize BLAS, summation order, …

“Bug” 1/3 in BLAS: IxAMAX

- IxAMAX returns index of first entry of largest “absolute value”
- ISAMAX:
  - ISAMAX([0,NaN,2]) = 3 and ISAMAX([NaN,0,2]) = 1
  - NaNs do not propagate consistently
- ICAMAX
  - OV = overflow threshold
  - ICAMAX([OV + i*OV, Inf + i*0]) = 1
  - ICAMAX points to finite entry instead of Inf

“Bug” 2/3 in BLAS: GER and SYR

- GER computes $A = A + \alpha x y^T$
- GER checks if $y(i) = 0$, does not multiply by it
  - Inf/NaN in x does not propagate to column i of A
  - If all $y(i) = 0$, no Infs/NaNs in x propagate
  - No checking for zeros in x
- SYR computes $A = A + \alpha x x^T$ when $A = A^T$
  - Can update upper or lower triangle of A
  - Code only checks for 0 in x^T, so can get different answer for upper and lower triangle

“Bug” 3/3 in BLAS: TRSV

- TRSV solves $T * x = b$ or $T^T * x = b$
- TRSV checks for zeros in x like GER and SYR
- Ex: $T = \begin{vmatrix} 1 & NaN & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{vmatrix}, b = \begin{vmatrix} 2 \\ 1 \\ 1 \end{vmatrix}$ yields $x = \begin{vmatrix} 1 \\ 0 \\ 1 \end{vmatrix}$
- NaN does not propagate
- Solving $(T^T)^T * x = b$ does not check for zeros, so NaN does propagate
- BLAS Bugs 1,2 and 3 combine so that SGESV does not propagate NaNs

Not Bugs in BLAS

- C = 0*A*B + beta*C = beta*C: expected semantics
- Different rules for complex*complex in C vs Fortran: live with it
- Different orders of summation, algorithms (Strassen, Gauss's complex*complex) may cause different exceptions: live with it
- Goals:
  - Provide new reference BLAS that propagates NaN and Inf "consistently",
  - Provide test code for exception handling
  - Encourage vendor adoption

”Bug” in SGESV

- Assume version that calls GER to update Schur complement, not newer recursive version that uses GEMM
- Solve $\begin{bmatrix} 1 & 0 \\ NaN & 2 \end{bmatrix} * x = \begin{vmatrix} 0 \\ 1 \end{vmatrix}$
- ISAMAX chooses 1 as pivot, not NaN
- GER updates 2 – NaN*0 = 2, NaN does not propagate
- TRSV does not multiply by 0 in x, NaN does not propagate, get x = [0; .5]

Goals for LAPACK (1/2)

- Use INFO for all exception reporting
- Modify LAPACK drivers that already compute norm(A) to report Inf and/or NaN in inputs, possibly return immediately if ill-posed (eg eig)
  - Add option to CLANGE
  - Consistency with LAPACKE

Goals for LAPACK (2/2)

- Some users want more reporting and control over exception handling, some want no changes
- Provide wrappers that allow more detailed reporting options using INFO:
  1. INFO behaves as usual (except for last slide)
  2. Check inputs and outputs for NaNs/Infs, report first one found
  3. Also report if any internal subroutine reported NaNs/Infs (and no input/output NaN/Inf to report)
- Provide test code for exception handling

Example of LAPACK Exception Handling

- Example: Solving Ax=B with

  SGESV(N, NRHS, A, LDA, IPIV, B, LDB, INFO)

  1. INFO = 0 means no error (current practice)
  2. INFO = -1 if N<0 (current)
  3. INFO = -2 if NRHS < 0 (and INFO not already set, current)
  4. INFO = -4 if LDA < max(1,N) (ditto)
  5. INFO = -7 if LDB < max(1,N) (ditto)
  6. INFO = k, $1 \leq k \leq$ N, if k is first zero pivot (ditto)
  7. INFO = -3 if A contains NaN/∞ on input (and INFO not set, **new**)
  8. INFO = -6 if B contains NaN/∞ on input (ditto)
  9. INFO = N+3 if A contains NaN/∞ on output (ditto)
  10. INFO = N+6 if B contains NaN/∞ on output (ditto)
  11. INFO = N+9 if SGETRF reports a NaN/∞ (ditto)
  12. INFO = N+10 if SGETRS reports a NaN/∞ (ditto)

How to build test code

- BLAS
  - Test both examples with NaN/Inf inputs, and unexceptional inputs that generate Infs (at least) internally
- LAPACK
  - Harder to generate unexceptional inputs that lead to Infs or NaNs at selected locations internally
  - Possible solution: “Fuzzing”, artificially insert Infs or NaNs at selected locations during execution
    - “Fluzzing”?

- More details available at:

https://people.eecs.berkeley.edu/~demmel/Exception_Handling_for_the_BLAS_and_LAPACK_130721.pdf

Comments welcome!