Undebuggability of Big Floating-Point Programs for Scientific and Engineering Computations

Inordinate effort and time are being expended on attempts,
often unsuccessful,
to debug floating-point programs,
most of them presumed already debugged,
whose application to some ostensibly innocuous data,
not necessarily test data,
has produced results that arouse suspicion
perhaps undeserved.

Wasted Time: Instances have occurred when a bug was never found before the underlying system was upgraded and the bug went elsewhere or away.

How are floating-point programs worse than others ?

Though heir to the same ills as others, these suffer three more:

•1) **Roundoff :** What you see is not what you get, and
what you get is not what you asked for.

•2) **Floating-Point Exceptions :** Over/Underflow, Invalid Operations, ...; no *flags* to expose them; ... .

•3) **Overly Aggressive Compiler "Optimizations" :**
O.K. for integers but not Flt. Pt. because of •1) & •2).

Would you like to go back to the years of my youth when floating-point was deemed refractory to error-analysis, thus undebuggable?

Undebuggability of Big Floating-Point Programs

•1) Roundoff **•2)** Exceptions **•3)** Over-Optimization

Exploitation of parallelism worsens our situation :

To minimize communications costs (*cf.* J.W. Demmel & *al.*) we use novel algorithms that have not yet been (and may never be) proved numerically stable for all innocuous data. Hence more obscure bugs.

Two Palliatives: (No complete cure exists.)

•I) To greatly attenuate damage from roundoff and exceptions, carry ***by default*** extravagantly excessive precision and range during computation; *cf.* pre-1980 Kernighan-Ritchie ***C*** .

•II) To diminish time spent debugging, we need aids:

- Compiler support for modes (*e.g.*, directed roundings) and flags as scoped variables, perhaps like *APL*'s System Variables *CT*, ... ,
- Linker-planted *Milestones* for flags' & NaNs' *Retrospective Diagnostics*.
- Compiler-Debugger collaboration to inject breaks *etc*. in object modules too

Current efforts towards that end at U.C. Berkeley:

<eecs.berkeley.edu/~grevy/publications/files/pdf/BaDeKaSe10.pdf>
supported by Sun Microsystems, and The MathWorks

See too my web page, <eecs.berkeley.edu/~wkahan/... for ...

- History: .../7094II.pdf>, .../BASCD08K.pdf>
- (Counter)Proposals: .../7Oct09.pdf>, .../Mindless.pdf>

The Challenge: Can we collect the necessary

Coalition of Competencies ?

Hardware, Compilers, Link-and-Loaders, Debuggers, Environments