A Comprehensive Study of Real-World Numerical Bug Characteristics

Anthony Di Franco, **Hui Guo**, and Cindy Rubio-Gonzalez

University of California, Davis

ASE'17, Urbana-Campaign IL, November 1st, 2017

Motivation

Precision Loss

fun1 = **lambda** x: $\sqrt{x+1} - \sqrt{x}$

fun2 = **lambda** x: $\dfrac{1}{\sqrt{x+1}+\sqrt{x}}$

x = 1e16

exact_value(fun1(x))

Out : '0'

exact_value(fun2(x))

Out : '5.000000000000000104612804150642363376633170446439 18..E-9'

Note: *"exact_XXX"* is implemented with python module "decimal".

Precision Loss

fun1 = **lambda** x: $\sqrt{x+1} - \sqrt{x}$

fun2 = **lambda** x: $\frac{1}{\sqrt{x+1}+\sqrt{x}}$

x = 1e16

exact_value(fun1(x))

Out : '0'   Cancellation happens.

exact_value(fun2(x))

Out : '5.00000000000000010461280415064236337663317044643918..E-9'

exact_computation(fun1(x))

Out : '5.0000000000E-9'

Note: *"exact_XXX"* is implemented with python module "decimal".

Precision Loss

fun1 = **lambda** x: $\sqrt{x+1} - \sqrt{x}$

fun2 = **lambda** x: $\frac{1}{\sqrt{x+1}+\sqrt{x}}$

x = 1e16

exact_value(fun1(x))

Out : '0'

Cancellation happens.

exact_value(fun2(x))

Out : '5.00000000000000010461280415064236337663317044643918..E-9'

exact_computation(fun1(x))

Out : '5.0000000000E-9'

Floating-point computations have to be carefully designed to avoid large precision loss in the result.

Note: *"exact_XXX"* is implemented with python module "decimal".

Challenges of Numerical Software

- Precision loss
    - A large numerical error in the output
- Numerical instability
    - A slight change in the input leads to a large change in the output
- Floating-point exceptions
    - Overflow / Underflow, NaNs
- Path divergence
    - Roundoff errors make the program to take a different branch
- ...

Empirical Questions to Answer

- Bug Categorization
  - What kinds of numerical bugs exist in real-world numerical software? How frequent are these bugs?
- Bug Detection
  - What are the symptoms of numerical bugs? Can we automate the detection of these bugs?
- Bug Fixing
  - How are numerical bugs fixed? Can we automatically fix these bugs?

Methodology

Selection of Numerical Libraries

Library	Language	Start Year	Bug Tracker	LOC	# Commits
NumPy	Python	2001	GitHub	15,366	15,731
SciPy	Python	2001	GitHub	823,446	17,012

Selection of Numerical Libraries

Library	Language	Start Year	Bug Tracker	LOC	# Commits
NumPy	Python	2001	GitHub	15,366	15,731
SciPy	Python	2001	GitHub	823,446	17,012
Elemental	C++	2010	GitHub	778,156	3,721

Selection of Numerical Libraries

Library	Language	Start Year	Bug Tracker	LOC	# Commits
NumPy	Python	2001	GitHub	15,366	15,731
SciPy	Python	2001	GitHub	823,446	17,012
Elemental	C++	2010	GitHub	778,156	3,721
LAPACK1	C/Fortran	2000	Website	1,613,856	NA
LAPACK2	C/Fortran	2008	GitHub	1,776,339	1,249

Selection of Numerical Libraries

Library	Language	Start Year	Bug Tracker	LOC	# Commits
NumPy	Python	2001	GitHub	15,366	15,731
SciPy	Python	2001	GitHub	823,446	17,012
Elemental	C++	2010	GitHub	778,156	3,721
LAPACK1	C/Fortran	2000	Website	1,613,856	NA
LAPACK2	C/Fortran	2008	GitHub	1,776,339	1,249
GSL	C/C++	2007	Savannah	278,617	5,596

Selection of Numerical Libraries

Library	Language	Start Year	Bug Tracker	LOC	# Commits
NumPy	Python	2001	GitHub	15,366	15,731
SciPy	Python	2001	GitHub	823,446	17,012
Elemental	C++	2010	GitHub	778,156	3,721
LAPACK1	C/Fortran	2000	Website	1,613,856	NA
LAPACK2	C/Fortran	2008	GitHub	1,776,339	1,249
GSL	C/C++	2007	Savannah	278,617	5,596

Identification of Numerical Bugs

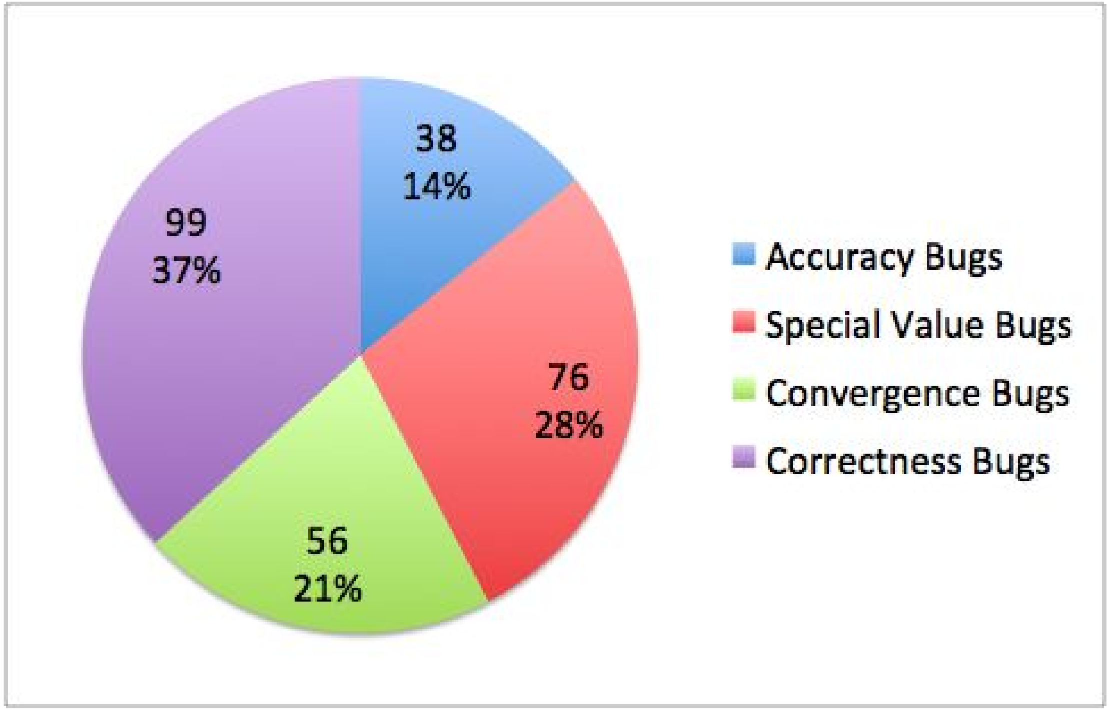
- Bug reports or GitHub issues/pull requests
- Filtered by:
  - Status : *Closed*
  - Associated Commits/Patches : *Number of (commits/patches) >= 1*
  - Labels : *w/o build, documentation or other conflicting labels with numeric*
  - Keyword Filters : *nan*, *exception*, *overflow*, *underflow*, *infinity*, *infinite*, *precision*, *unstable*, *instability*, *ringing*, *unbounded*, *roundoff*, *truncation*, *rounding*, *diverge*, *cancellation*, *cancel*, *accuracy*, *accurate*
- Additional random sampling if needed
- Manual inspection

Identification of Numerical Bugs

Library	#Bugs	Filter/Sample	Inspected	Identified
NumPy	9008	Sample	100	24
SciPy	7345	Sample	100	71
Elemental	230	Status Filter	185	9
LAPACK1	148	Commit Filter	135	38
LAPACK2	140	Label Filter	90	4
GSL	218	None	218	123
Total	**17089**	**NA**	**828**	**269**

Main Findings

Numerical Bug Categorization



Out of the 828 bugs manually inspected, 269 are numerical bugs

Example of Accuracy Bug

```
>>>  import numpy as n
>>>  a = n.ones( (1000, 1000), dtype=n.float32 ) * 132.00005
>>>  a.min()
132.000045776
>>>  a.max()
132.000045776
>>>  a.mean()
133.96639999999999
```

Insufficient Precision Type

Problem: *The "float" accumulator loses precision*

Solution: *Increase precision for the summation*

Accuracy Bugs

- Accuracy bugs correspond to large precision loss
- 14% (38/269) are accuracy bugs in our dataset with a symptom of wrong results
- Most of the bugs are related to
  - *Insufficient precision types*
  - *Buggy arithmetic expressions*
  - *Ill-conditioned problems*
- Bug exposure: to generate inputs for automated detection remains a challenge

Example of Special Value Bug

```
>>> import numpy as n
>>> np.max ( np.array( [-1, np.nan, -2] ) )
-2
```

Missing NaN Check

Problem: *Comparison against* NaN *evaluates to false*

Solution: *Check for* NaN *inputs*

Bugs Related to Special Values

- Special values are *signed zero, subnormal numbers, infinities and NaNs*
- We found 28% (76/269) of numerical bugs are related to special values
- Common symptoms include NaN or other wrong results, infinite loops
- Most of these bugs involve:
  - *Missing or buggy NaN checks*
  - *Overflow/underflow*
- Bug exposure:
  - *Insert NaNs to inputs to test missing/buggy NaN checks*
  - *To generate fair inputs to test overflow/underflow remains a challenge*

Convergence Bugs

- Iterative or series approximation diverges or converges too slowly due to magnified roundoff or truncation errors
- 21% (56/269) of numerical bugs are convergence bugs
- Common convergence bugs include:
  - Problematic approximation formula that yield wrong result
  - Problematic iterative approximation causes infinite loop
  - NaNs cause divergent behavior
- Detecting and fixing these bugs requires domain knowledge
  - Usually requires finding a better approximation technique

Correctness Bugs

- 37% (99/269) of numerical bugs are correctness bugs
- Common correctness bugs include:
  - Typographical errors when transcribing formulas
  - Using approximations outside a function's domain
  - Compiler optimizations violating semantics of mathematical operations
- The most common symptom was wrong results
- Bug exposure: users often compare against less efficient implementations

Lessons Learned

- Few patterns for bug finding and bug fixing
  - Still these could be applied to other code bases
- A large number of bugs are domain specific
  - Techniques such as differential testing may be useful to find them
- Special value bugs, accuracy bugs, and convergence bugs could be found applying program analysis techniques
- In general, automated bug fixing is very challenging for numerical bugs
- No indication during manual inspection that users or developers use tools for bug finding or bug fixing

Conclusions

- We identified and examined *269 numerical bugs* out of 828 bugs from a diverse set of numerical libraries: NumPy, SciPy, Elemental, LAPACK, and GSL
- We found that numerical bugs can be largely categorized into four groups: *accuracy bugs, special-value bugs, convergence bugs, and correctness bugs*
- We studied the *symptoms* and *fixes* of the four bug categories and discussed the opportunities to *automate* detection and fixing

Acknowledgments

NSF

Hellman Fellows Fund

UC DAVIS
UNIVERSITY OF CALIFORNIA

Questions?

higuo@ucdavis.edu

https://hguo15.github.io/huiguo.github.io/

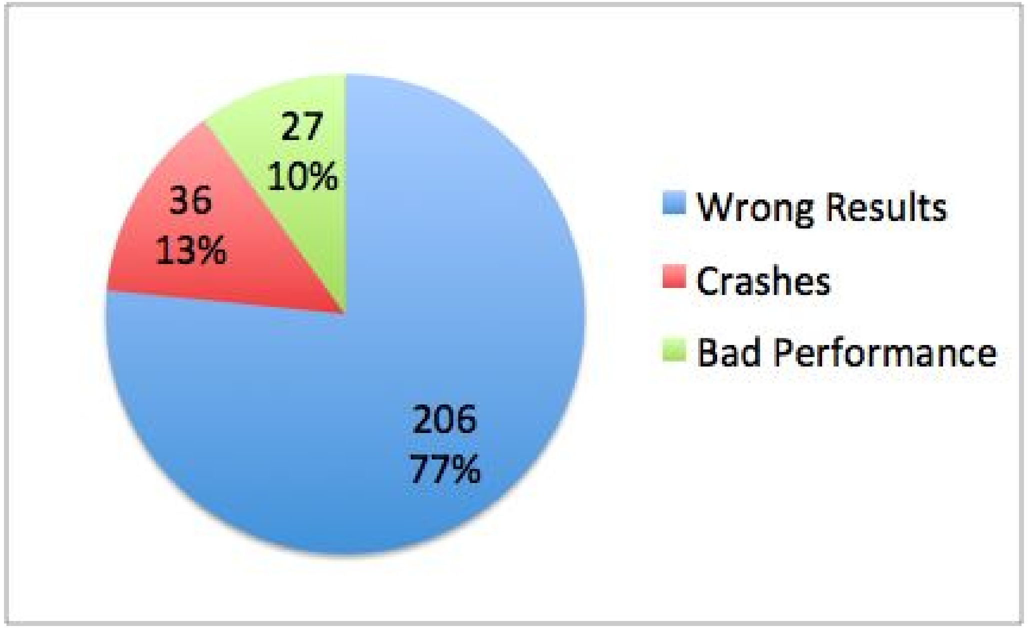
Backup Slides

Bug Categorization

32% (*269/828*) of the bugs examined are numerical bugs,

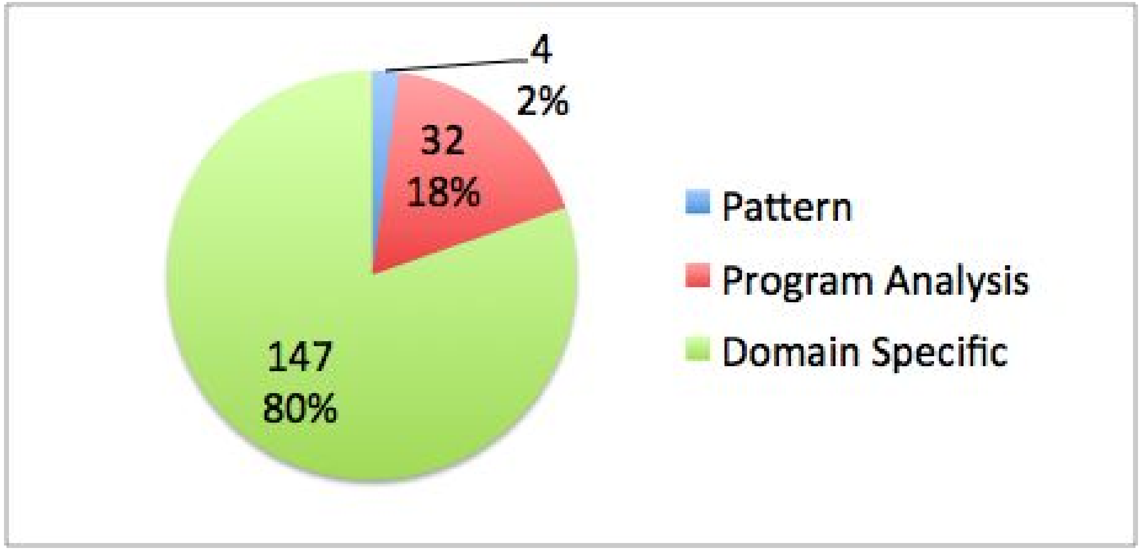
Library	Accuracy	Special Values	Convergence	Correctness	Total
NumPy	5	16	0	3	24
SciPy	8	27	6	30	71
Elemental	0	0	0	9	9
LAPACK	11	11	11	9	42
GSL	14	22	39	48	123
Total	38	76	56	99	**269**

Symptoms of Numerical Bugs



The most common symptom of numerical bugs are wrong results

Bug Fixing Strategies



Automating bug fixing for numerical bugs may be difficult in most cases