CaSo

Automatically improving floating point code
Scientists Write Code

Every scientist needs to write code
  Analyze data
  Simulate models
  Control experiments
Scientists Write Code

Every scientist needs to write code
  Analyze data
  Simulate models
  Control experiments

They have little computer science training
How scientific code is written

1. Come up with mathematical formula

\[ f(x) = \sqrt{x + 1} - \sqrt{x} \]
How scientific code is written

1. Come up with mathematical formula

\[ f(x) = \sqrt{x + 1} - \sqrt{x} \]

2. Write as floating-point code

\[ f(x) := \text{sqrt}(x + 1) - \text{sqrt}(x) \]
How scientific code is written

$$f(x) := \sqrt{x + 1} - \sqrt{x}$$

3. Test code

$$f(1) = 0.41421...$$

$$f(1) = 0.41421...$$
How scientific code is written

\[ f(x) := \sqrt{x + 1} - \sqrt{x} \]

3. Test code

\[ f(1) = 0.41421... \]

4. Publish
But try a few other values...

\[ f(3141592653589793) = 7.451 \times 10^{-9} \]

\[ f(3141592653589793) \approx 8.921 \times 10^{-9} \]
But try a few other values...

\[ f(3141592653589793) = 7.451 \times 10^{-9} \]

\[ f(3141592653589793) \approx 8.921 \times 10^{-9} \]

That’s a 16% error!
Outline

Why did this happen?

How can we fix it?

How does Casio help?
Outline

Why did this happen?

How can we fix it?

How does Casio help?
Why did this happen?

“Because floating-point is imprecise!”
Why did this happen?

“Because floating-point is imprecise!”

We want something more constructive.
Why did this happen?

“Because floating-point is imprecise!”

We want something more constructive.

“Rounding”
Why did this happen?

“Because floating-point is imprecise!”

We want something more constructive.

“Rounding”

Let’s try to be a bit more precise...
Let’s try to be a bit more precise...

177.24559232...
- 177.24277136...

0.00282006...
Let’s try to be a bit more precise…

177.24559  232
- 177.24277  136

00282  006

Rounding error
Let’s try to be a bit more precise...

\[
\begin{array}{cc}
177.24559 & 232 \\
- 177.24277 & 136 \\
\hline
.00282 & 006
\end{array}
\]

→ \(.000001\%\) error

→ \(.000001\%\) error

→ \(.03\%\) error

Rounding error
Let’s try to be a bit more precise...

\[
\begin{array}{cc}
177.24559 & 232 \\
-177.24277 & 136 \\
\hline
0.00282 & 006 \\
\end{array}
\]

→ .000001% error
→ .000001% error
→ .03% error

Error in output proportional to size of input.
For our example

\[ \sqrt{x + 1} - \sqrt{x} \approx \frac{1}{2\sqrt{x}} \]
For our example

\[\sqrt{x + 1} - \sqrt{x} \approx \frac{1}{2\sqrt{x}}\]

\[f(x) \approx f(x) + \sqrt{x} \cdot \epsilon\]
For our example

\[
\sqrt{x + 1} - \sqrt{x} \approx \frac{1}{2\sqrt{x}}
\]

\[
f(x) \approx f(x) + \sqrt{x} \cdot \epsilon
\]

\[
(f(x) - f(x)) \approx \sqrt{x} \cdot \epsilon \approx 2x f(x) \epsilon
\]
For our example

\[
\sqrt{x + 1} - \sqrt{x} \approx \frac{1}{2\sqrt{x}}
\]

\[
f(x) \approx f(x) + \sqrt{x} \cdot \epsilon
\]

\[
(f(x) - f(x)) \approx \sqrt{x} \cdot \epsilon \approx 2xf(x)\epsilon
\]

error \approx 2x\epsilon
So, in summary

Code is numerically imprecise

The subtraction is the culprit

Figuring out why was hard
Outline

Why did this happen?

How can we fix it?

How does Casio help?
How can we fix it?

Run with higher precision?
  Software floating point is slow.

Add correction terms?
  Very hard to do; very error-prone
How can we fix it?

Better idea: rephrase the program

Compute the same thing in a different way.

Somehow get rid of the subtraction
One way of rephrasing it

\[ \sqrt{x + 1} - \sqrt{x} \]
One way of rephrasing it

\[ \sqrt{x+1} - \sqrt{x} = (\sqrt{x+1} - \sqrt{x}) \frac{\sqrt{x+1} + \sqrt{x}}{\sqrt{x+1} + \sqrt{x}} \]
One way of rephrasing it

\[
\sqrt{x + 1} - \sqrt{x} = (\sqrt{x + 1} - \sqrt{x}) \frac{\sqrt{x + 1} + \sqrt{x}}{\sqrt{x + 1} + \sqrt{x}} = \frac{\sqrt{x+1}^2 - \sqrt{x}^2}{\sqrt{x + 1} + \sqrt{x}}
\]
One way of rephrasing it

\[
\sqrt{x+1} - \sqrt{x} = (\sqrt{x+1} - \sqrt{x}) \frac{\sqrt{x+1} + \sqrt{x}}{\sqrt{x+1} + \sqrt{x}} \\
= \frac{\sqrt{x+1}^2 - \sqrt{x}^2}{\sqrt{x+1} + \sqrt{x}} = \frac{x+1 - x}{\sqrt{x+1} + \sqrt{x}}
\]
One way of rephrasing it

\[
\sqrt{x+1} - \sqrt{x} = (\sqrt{x+1} - \sqrt{x}) \frac{\sqrt{x+1} + \sqrt{x}}{\sqrt{x+1} + \sqrt{x}} \\
= \frac{\sqrt{x+1}^2 - \sqrt{x}^2}{\sqrt{x+1} + \sqrt{x}} \\
= \frac{x+1 - x}{\sqrt{x+1} + \sqrt{x}} \\
= \frac{1}{\sqrt{x+1} + \sqrt{x}} 
\]
Implementing this rephrasing

\[ f(x) = \frac{1}{\sqrt{x + 1} + \sqrt{x}} \]

This version has effectively no error.
Implementing this rephrasing

\[ f(x) = \frac{1}{\sqrt{x + 1} + \sqrt{x}} \]

This version has effectively no error.

But it’s a harder to understand:

Is this computing the right function?
So, in summary

Problem solved

Algebraic ingenuity required

Fixing it was hard
Outline

Why did this happen?

How can we fix it?

How does Casio help?
How does Casio help?

Diagnosing and fixing precision problems is hard
How does Casio help?

Diagnosing and fixing precision problems is hard

Casio automatically...

Computes error
Finds better code
How does Casio help?

Diagnosing and fixing precision problems is hard

Casio automatically...
  Computes error
  Finds better code

Resulting expression is the same over the reals but better over the floats
Automatically computing error

Compute exact answers with arbitrary precision

Use program analysis tools
to find bad subexpressions
Automatically improving code

Small database of mathematical identities
  e.g. \( a - a = 0 \)
  e.g. \( a^2 - b^2 = (a + b)(a - b) \)

Apply identities to the problem subexpression
  recursive goal-directed rewrite

Evaluate resulting code versus exact answer
Inner Loop

- Equation
- Input Distribution
Inner Loop

Equation → Computed Output
Input Distribution → Exact Output

Arbitrary Precision
Inner Loop

Equation

Input Distribution

Computed Output

Exact Output

Bad Subexpressions

Rewrite Database

Candidate Programs

Arbitrary Precision

Program Analysis

Rewrite Generation
Inner Loop

Equation → Computed Output
Input Distribution → Exact Output

Arbitrary Precision → Program Analysis

Bad Subexpressions → Candidate Programs
Rewrite Database → Best Candidates

Rewrite Generation → Ranking
Domain Knowledge

Simplification

Inferring branch conditions

Periodicity analysis
> (improve
  '(\x (- (sqrt (+ x 1)) (sqrt x)))
  3)
> (improve
  '(
    \(x\) (- (sqrt (+ x 1)) (sqrt x))
  )
  3)

(\(x\)
  (if (< x 0.0007021373107872404)
    (- (sqrt (+ x 1)) (sqrt x))
    (/ 1 (+ (sqrt x) (sqrt (+ x 1)))))))
> (improve
   '(\x (-(sqrt (+ x 1)) (sqrt x)))
   3)

(\x
   (if (< x 0.0007021373107872404)
     (-(sqrt (+ x 1)) (sqrt x))
     (/ 1 (+ (sqrt x) (sqrt (+ x 1))))))

Improvement by an average of 23.2 bits
Future work

Extracting floating point computation from code

More domain knowledge

Provide explanation of what Casio did

Unsound rewrites