Formally Verified Numerical Methods

Andrew Appel
Princeton University

David Bindel
Cornell University

https://VeriNum.org

Machine-checked proofs of numerical accuracy and program correctness,
end-to-end from foundational specifications of C language semantics,
IEEE floating point, to high-level problem specification

General Approach

Mathematical specification to be approximated

Coq proof: bound the **method error**

Real-valued functional model of algorithm

Coq proof: bound the floating-point roundoff error

Floating point functional model of algorithm

Coq proof: C program exactly refines float functional model

C program

Coq: compose all proofs into main theorem

Example: Differential Equation

$$\frac{dp}{dt} = -\omega^2 q \qquad \frac{dq}{dt} = p$$

method error proof

$\mathrm{stepR}(h:\mathbb{R})((p,q):\mathbb{R}\times\mathbb{R}):\mathbb{R}\times\mathbb{R} :=$
$\text{let } q' := q + h\cdot p + \frac{1}{2}(h\cdot h)\cdot(-q) \text{ in}$
$\text{let } p' := p + \frac{1}{2}h\cdot(-q + -q') \text{ in } \ (p',q')$

roundoff error proof

$\mathrm{stepF}(h:\mathbb{F})((p,q):\mathbb{F}\times\mathbb{F}):\mathbb{F}\times\mathbb{F} :=$
$\text{let } q' := q + h\cdot p + \frac{1}{2}(h\cdot h)\cdot(-q) \text{ in}$
$\text{let } p' := p + \frac{1}{2}h\cdot(-q + -q') \text{ in } \ (p',q')$

program + data structures proof

```
for (n = 0; n < N; n++) {
   a = -(s->q);
   s->q = s->q + h * s->p + (0.5f * (h * h)) * a;
   s->p = s->p + (0.5f *  h) * (a + -(s->q));
   t = t + h;
}
```

Coq: combine errors using triangle inequality

Theorem (proved in Coq):

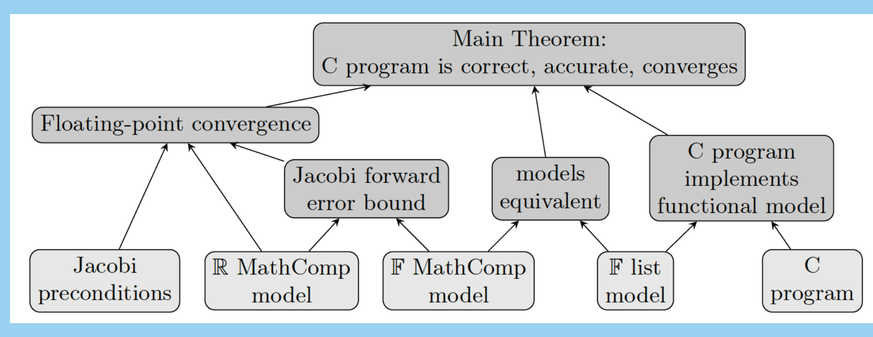
After *N* time steps,
the absolute error in
position or momentum
will be within the
specified tolerance.

Ariel E. Kellison & Andrew W. Appel,
Verified Numerical Methods for Ordinary Differential Equations,
NSV'22: Workshop on Numerical Software Verification, 2022.

Example: Jacobi iteration

Find x such that $Ax=b$

Method: let D be the diagonal of A, let $N=A-D$,
iterate: $x_{n+1} = D^{-1}(b - N\,x_n)$



Theorem (proved in Coq):

Given matrix A and vector b
satisfying our preconditions,
the C program **will** converge
within k iterations;
and the residual **will** be
within the tolerance.

Mohit Tekriwal, Andrew W. Appel, Ariel E. Kellison, David Bindel, and Jean-Baptiste Jeannin.
Verified correctness, accuracy, and convergence of a stationary iterative linear solver: Jacobi method
16th Conference on Intelligent Computer Mathematics, September 2023.

Example: Cholesky decomposition

Let A be positive definite,
find R upper triangular such $R^{\mathrm{T}}R = A$

```
void cholesky (unsigned n, double A[N][N], double R[N][N]) {
  unsigned i,j,k; double s;
  for (j=0; j<n; j++) {
     for (i=0; i<j; i++) {
       s = A[i][j];
       for (k=0; k<i; k++) s -= R[k][i]*R[k][j];
       R[i][j]=s/R[i][i];
     }
     s = A[j][j];
     for (k=0; k<j; k++) {
        double rkj = R[k][j];
        s -= rkj*rkj;
     }
     R[j][j] = sqrt(s);
  }
}
```

Floating-point accuracy proof in Coq: based on,
Pierre Roux, Formal Proofs of Rounding Error Bounds,
Journal of Automated Reasoning, Volume 57, pages 135–156, (2016).

C program correctness proof in Coq: work in progress, using VST.

Libraries and Tools

Verified Software Toolchain

Andrew W. Appel *et al.*
Foundational program logic and
proof system for verifying C programs,
2011-2024

VCFloat2
Floating-point
roundoff error
analysis tool
in Coq

Andrew W. Appel & Ariel E. Kellison
VCFloat2: Floating-point Error Analysis in Coq,
CPP'24: ACM SIGPLAN International Conference on Certified Programs and Proofs, 2024

LAProof
Verified
linear algebra
library

Ariel E. Kellison, Andrew W. Appel, Mohit Tekriwal, and David Bindel
LAProof: a library of formal accuracy and correctness proofs for sparse linear algebra programs. *30th IEEE International Symposium on Computer Arithmetic,* 2023.

Why3
Intermediate
language for
verification

Joshua M. Cohen and
Philip Johnson-Freyd.
A Formalization of
Core Why3 in Coq.
POPL 2024: 51st ACM SIGPLAN Symposium on Principles of Programming Languages, 2024.

NumFuzz
Floating-point
roundoff analysis
via typechecking

Ariel E. Kellison & Justin Hsu
Numerical Fuzz: A Type System
for Rounding Error Analysis.
PLDI'24: ACM SIGPLAN Conf. on Programming Language Design and Implementation, 2024

Broader Impact:

One trend in computer architecture motivates accuracy guarantees more than ever:
Supercomputers are no longer designed for scientific computing (with 64-bit, 128-bit floating point); they are sold for machine learning (with 32-bit, 16-bit, 8-bit floating point).
Can no longer aim for accuracy by just throwing extra bits of precision at the problem!

The 2024 Formal Methods in the Field PI Meeting (2024 FMitF PI Meeting)
November 12-13, 2024 | The University of Iowa | Iowa City, Iowa