LAProof: A Library of Formal Proofs of Accuracy and Correctness for Linear Algebra Programs

Ariel Kellison\textsuperscript{1,2}, Andrew W. Appel\textsuperscript{3}, Mohit Tekriwal\textsuperscript{4}, and David Bindel\textsuperscript{1}

1. Dept. of Computer Science, Cornell University
2. Sandia National Laboratories, Livermore, CA
3. Dept. of Computer Science, Princeton University
4. Dept. of Aerospace Engineering, University of Michigan
What is LAProof?

LAProof is a library of machine-checked accuracy proofs for basic linear algebra operations.
What is LAProof?

LAProof is a library of machine-checked accuracy proofs for basic linear algebra operations. The accuracy proofs from LAProof

- assume only a low-level formal model of IEEE-754 arithmetic,
What is LAProof?

LAProof is a library of machine-checked accuracy proofs for basic linear algebra operations. The accuracy proofs from LAProof

- assume only a low-level formal model of IEEE-754 arithmetic,
- are mixed (backward-forward) rounding error bounds that account for underflow,
What is LAProof?

LAProof is a library of machine-checked accuracy proofs for basic linear algebra operations.

The accuracy proofs from LAProof

- assume only a low-level formal model of IEEE-754 arithmetic,
- are mixed (backward-forward) rounding error bounds that account for underflow,
- capture low order error terms exactly – not approximating as $O(u^2)$,
What is LAProof?

LAProof is a library of machine-checked accuracy proofs for basic linear algebra operations.

The accuracy proofs from LAProof

- assume only a low-level formal model of IEEE-754 arithmetic,
- are mixed (backward-forward) rounding error bounds that account for underflow,
- capture low order error terms exactly – not approximating as $O(u^2)$,
- can be used to formally verify the accuracy of programs implementing operations defined by the basic linear algebra subprograms (BLAS) specification, and
What is LAProof?

LAProof is a library of machine-checked accuracy proofs for basic linear algebra operations.

The accuracy proofs from LAProof

- assume only a low-level formal model of IEEE-754 arithmetic,
- are mixed (backward-forward) rounding error bounds that account for underflow,
- capture low order error terms exactly – not approximating as $O(u^2)$,
- can be used to formally verify the accuracy of programs implementing operations defined by the basic linear algebra subprograms (BLAS) specification, and
- are developed entirely within the Coq proof assistant.

VeriNum / LAProof
Why verify the accuracy of programs implementing BLAS?
BLAS implementations are vital in *numerical analysis*

“...our mission is to compute quantities that are typically uncomputable, from an analytic point of view, and to do it with lightning speed.” - Trefethen 1992
BLAS implementations are vital in *numerical analysis*

“...our mission is to compute quantities that are typically uncomputable, from an analytic point of view, and to do it with lightning speed.” - Trefethen 1992

- continuous mathematical problem
- discrete algorithm
- application software
BLAS implementations are vital in numerical analysis

“...our mission is to compute quantities that are typically uncomputable, from an analytic point of view, and to do it with lightning speed.” - Trefethen 1992
Accuracy proofs give us confidence in these algorithms and their implementations.
Accuracy proofs give us confidence in these algorithms and their implementations
Accuracy proofs give us confidence in these algorithms and their implementations.
Accuracy proofs give us confidence in these algorithms and their implementations.
In high-consequence settings, machine-checked proofs are the gold standard
In high-consequence settings, machine-checked proofs are the gold standard.
In high-consequence settings, machine-checked proofs are the gold standard
What is a machine-checked proof?

- A formal derivation in a formal logical system, checked by a proof-checking program
What is a machine-checked proof?

- A formal derivation in a formal logical system, checked by a proof-checking program
- A common tool for developing machine checked proofs is a **proof assistant**
- Proof assistants provide
  - a DSL (domain-specific language) for building proofs
  - a program that verifies whether proofs are valid derivations in a formal logic
  - libraries of definitions, theorems, and programs for proof automation

*adapted from Ringer et al, 2019.*
What is a machine-checked proof?

- A formal derivation in a formal logical system, checked by a proof-checking program
- A common tool for developing machine checked proofs is a proof assistant
- Proof assistants provide
  - a DSL (domain-specific language) for building proofs
  - a program that verifies whether proofs are valid derivations in a formal logic
  - libraries of definitions, theorems, and programs for proof automation

![Diagram]

* The Coq proof assistant – free, open-source software; development largely supported by INRIA (The National Institute for Research in Digital Science and Technology) since 1989.

* adapted from Ringer et al, 2019.
Which linear algebra operations does LAProof provide machine-checked accuracy proofs for?
What operations does LAProof provide?

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>LAPROOF Vector Operations</th>
<th>TABLE II</th>
<th>LAPROOF Matrix-Vector Operations</th>
<th>TABLE III</th>
<th>LAPROOF Matrix Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOT</td>
<td>$r \leftarrow x \cdot y$</td>
<td>MV</td>
<td>$r \leftarrow Ax$</td>
<td>sMat</td>
<td>$R \leftarrow \alpha A$</td>
</tr>
<tr>
<td>sVec</td>
<td>$r \leftarrow \alpha x$</td>
<td>sMV</td>
<td>$r \leftarrow \alpha Ax$</td>
<td>MM</td>
<td>$R \leftarrow AB$</td>
</tr>
<tr>
<td>SUM</td>
<td>$r \leftarrow \sum_i x_i$</td>
<td>GEMV</td>
<td>$r \leftarrow \alpha Ax + \beta y$</td>
<td>MatAdd</td>
<td>$R \leftarrow A + B$</td>
</tr>
<tr>
<td>VecAdd</td>
<td>$r \leftarrow x + y$</td>
<td></td>
<td></td>
<td>sMM</td>
<td>$R \leftarrow \alpha AB$</td>
</tr>
<tr>
<td>VecAXPBY</td>
<td>$r \leftarrow \alpha x + \beta y$</td>
<td></td>
<td></td>
<td>MatAXPBY</td>
<td>$R \leftarrow \alpha X + \beta Y$</td>
</tr>
<tr>
<td>VecNRM1</td>
<td>$r \leftarrow</td>
<td></td>
<td>x</td>
<td></td>
<td>_1$</td>
</tr>
<tr>
<td>VecNRM2</td>
<td>$r \leftarrow</td>
<td></td>
<td>x</td>
<td></td>
<td>_2$</td>
</tr>
</tbody>
</table>
What operations does LAProof provide?

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>LAProof Vector Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOT</td>
<td>$r \leftarrow x \cdot y$</td>
</tr>
<tr>
<td>sVec</td>
<td>$r \leftarrow \alpha x$</td>
</tr>
<tr>
<td><strong>SUM</strong></td>
<td>$r \leftarrow \sum_i x_i$</td>
</tr>
<tr>
<td>VecAdd</td>
<td>$r \leftarrow x + y$</td>
</tr>
<tr>
<td>VecAXPBY</td>
<td>$r \leftarrow \alpha x + \beta y$</td>
</tr>
<tr>
<td>VecNRM1</td>
<td>$r \leftarrow |x|_1$</td>
</tr>
<tr>
<td>VecNRM2</td>
<td>$r \leftarrow |x|_2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE II</th>
<th>LAProof Matrix-Vector Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>MV</td>
<td>$r \leftarrow Ax$</td>
</tr>
<tr>
<td>sMV</td>
<td>$r \leftarrow \alpha Ax$</td>
</tr>
<tr>
<td>GEMV</td>
<td>$r \leftarrow \alpha Ax + \beta y$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE III</th>
<th>LAProof Matrix Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>sMat</td>
<td>$R \leftarrow \alpha A$</td>
</tr>
<tr>
<td>MM</td>
<td>$R \leftarrow AB$</td>
</tr>
<tr>
<td>MatAdd</td>
<td>$R \leftarrow A + B$</td>
</tr>
<tr>
<td>sMM</td>
<td>$R \leftarrow \alpha AB$</td>
</tr>
<tr>
<td>MatAXPBY</td>
<td>$R \leftarrow \alpha X + \beta Y$</td>
</tr>
<tr>
<td>GEMM</td>
<td>$R \leftarrow \alpha AX + \beta Y$</td>
</tr>
</tbody>
</table>
What operations does LAProof provide?

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>LAPROOF VECTOR OPERATIONS</th>
<th>TABLE II</th>
<th>LAPROOF MATRIX-VECTOR OPERATIONS</th>
<th>TABLE III</th>
<th>LAPROOF MATRIX OPERATIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOT</td>
<td>$r \leftarrow x \cdot y$</td>
<td>MV</td>
<td>$r \leftarrow Ax$</td>
<td>sMat</td>
<td>$R \leftarrow \alpha A$</td>
</tr>
<tr>
<td>sVec</td>
<td>$r \leftarrow \alpha x$</td>
<td>sMV</td>
<td>$r \leftarrow \alpha Ax$</td>
<td>MM</td>
<td>$R \leftarrow AB$</td>
</tr>
<tr>
<td>SUM</td>
<td>$r \leftarrow \sum_i x_i$</td>
<td>GEMV</td>
<td>$r \leftarrow \alpha Ax + \beta y$</td>
<td>MatAdd</td>
<td>$R \leftarrow A + B$</td>
</tr>
<tr>
<td>VecAdd</td>
<td>$r \leftarrow x + y$</td>
<td></td>
<td></td>
<td>sMM</td>
<td>$R \leftarrow \alpha AB$</td>
</tr>
<tr>
<td>VecAXPBY</td>
<td>$r \leftarrow \alpha x + \beta y$</td>
<td></td>
<td></td>
<td>MatAXPBY</td>
<td>$R \leftarrow \alpha X + \beta Y$</td>
</tr>
<tr>
<td>VecNRM1</td>
<td>$r \leftarrow |x|_1$</td>
<td></td>
<td></td>
<td>GEMM</td>
<td>$R \leftarrow \alpha AX + \beta Y$</td>
</tr>
<tr>
<td>VecNRM2</td>
<td>$r \leftarrow |x|_2$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
What operations does LAProof provide?

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>TABLE II</th>
<th>TABLE III</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAPROOF Vector Operations</td>
<td>LAPROOF Matrix-Vector Operations</td>
<td>LAPROOF Matrix Operations</td>
</tr>
<tr>
<td>DOT</td>
<td>MV</td>
<td>sMat</td>
</tr>
<tr>
<td>sVec</td>
<td>r ← Ax</td>
<td>MM</td>
</tr>
<tr>
<td>SUM</td>
<td>sMV</td>
<td>MatAdd</td>
</tr>
<tr>
<td>VecAdd</td>
<td>GEMV</td>
<td>sMM</td>
</tr>
<tr>
<td>VecAXPY</td>
<td>r ← αx + βy</td>
<td>MatAXPY</td>
</tr>
<tr>
<td>VecNRM1</td>
<td></td>
<td>GEMM</td>
</tr>
<tr>
<td>VecNRM2</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>r ← x · y</td>
<td></td>
</tr>
<tr>
<td></td>
<td>r ← αx</td>
<td>R ← αA</td>
</tr>
<tr>
<td></td>
<td>r ← ∑ᵢ xᵢ</td>
<td>R ← AB</td>
</tr>
<tr>
<td></td>
<td>r ← x + y</td>
<td>R ← A + B</td>
</tr>
<tr>
<td></td>
<td>r ← αx + βy</td>
<td>R ← αAB</td>
</tr>
<tr>
<td></td>
<td>r ←</td>
<td></td>
</tr>
<tr>
<td></td>
<td>r ←</td>
<td></td>
</tr>
</tbody>
</table>
What operations does LAProof provide?

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>LAProof Vector Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOT</td>
<td>$r \leftarrow x \cdot y$</td>
</tr>
<tr>
<td>sVec</td>
<td>$r \leftarrow \alpha x$</td>
</tr>
<tr>
<td>SUM</td>
<td>$r \leftarrow \sum_i x_i$</td>
</tr>
<tr>
<td>VecAdd</td>
<td>$r \leftarrow x + y$</td>
</tr>
<tr>
<td>VecAXPBY</td>
<td>$r \leftarrow \alpha x + \beta y$</td>
</tr>
<tr>
<td>VecNRM1</td>
<td>$r \leftarrow |x|_1$</td>
</tr>
<tr>
<td>VecNRM2</td>
<td>$r \leftarrow |x|_2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE II</th>
<th>LAProof Matrix-Vector Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>MV</td>
<td>$r \leftarrow Ax$</td>
</tr>
<tr>
<td>sMV</td>
<td>$r \leftarrow \alpha Ax$</td>
</tr>
<tr>
<td>GEMV</td>
<td>$r \leftarrow \alpha Ax + \beta y$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE III</th>
<th>LAProof Matrix Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>sMat</td>
<td>$R \leftarrow \alpha A$</td>
</tr>
<tr>
<td>MM</td>
<td>$R \leftarrow AB$</td>
</tr>
<tr>
<td>MatAdd</td>
<td>$R \leftarrow A + B$</td>
</tr>
<tr>
<td>sMM</td>
<td>$R \leftarrow \alpha AB$</td>
</tr>
<tr>
<td>MatAXPBY</td>
<td>$R \leftarrow \alpha X + \beta Y$</td>
</tr>
<tr>
<td>GEMM</td>
<td>$R \leftarrow \alpha AX + \beta Y$</td>
</tr>
</tbody>
</table>
What operations does LAPRooft provide?

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>LAPRooft Vector Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOT</td>
<td>$r \leftarrow x \cdot y$</td>
</tr>
<tr>
<td>sVec</td>
<td>$r \leftarrow \alpha x$</td>
</tr>
<tr>
<td>SUM</td>
<td>$r \leftarrow \sum_i x_i$</td>
</tr>
<tr>
<td>VecAdd</td>
<td>$r \leftarrow x + y$</td>
</tr>
<tr>
<td>VecAXPBY</td>
<td>$r \leftarrow \alpha x + \beta y$</td>
</tr>
<tr>
<td>VecNRM1</td>
<td>$r \leftarrow |x|_1$</td>
</tr>
<tr>
<td>VecNRM2</td>
<td>$r \leftarrow |x|_2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE II</th>
<th>LAPRooft Matrix-Vector Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>MV</td>
<td>$r \leftarrow Ax$</td>
</tr>
<tr>
<td>sMV</td>
<td>$r \leftarrow \alpha Ax$</td>
</tr>
<tr>
<td>GEMV</td>
<td>$r \leftarrow \alpha Ax + \beta y$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE III</th>
<th>LAPRooft Matrix Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>sMat</td>
<td>$R \leftarrow \alpha A$</td>
</tr>
<tr>
<td>MM</td>
<td>$R \leftarrow AB$</td>
</tr>
<tr>
<td>MatAdd</td>
<td>$R \leftarrow A + B$</td>
</tr>
<tr>
<td>sMM</td>
<td>$R \leftarrow \alpha AB$</td>
</tr>
<tr>
<td>MatAXPBY</td>
<td>$R \leftarrow \alpha X + \beta Y$</td>
</tr>
<tr>
<td>GEMM</td>
<td>$R \leftarrow \alpha AX + \beta Y$</td>
</tr>
</tbody>
</table>
What operations does LAProof provide?

<table>
<thead>
<tr>
<th>TABLE I</th>
<th>LAProof Vector Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>DOT</td>
<td>$r \leftarrow x \cdot y$</td>
</tr>
<tr>
<td>sVec</td>
<td>$r \leftarrow \alpha x$</td>
</tr>
<tr>
<td>SUM</td>
<td>$r \leftarrow \sum_i x_i$</td>
</tr>
<tr>
<td>VecAdd</td>
<td>$r \leftarrow x + y$</td>
</tr>
<tr>
<td>VecAXPBY</td>
<td>$r \leftarrow \alpha x + \beta y$</td>
</tr>
<tr>
<td>VecNRM1</td>
<td>$r \leftarrow |x|_1$</td>
</tr>
<tr>
<td>VecNRM2</td>
<td>$r \leftarrow |x|_2$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE II</th>
<th>LAProof Matrix-Vector Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>MV</td>
<td>$r \leftarrow Ax$</td>
</tr>
<tr>
<td>sMV</td>
<td>$r \leftarrow \alpha Ax$</td>
</tr>
<tr>
<td>GEMV</td>
<td>$r \leftarrow \alpha Ax + \beta y$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>TABLE III</th>
<th>LAProof Matrix Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>sMat</td>
<td>$R \leftarrow \alpha A$</td>
</tr>
<tr>
<td>MM</td>
<td>$R \leftarrow AB$</td>
</tr>
<tr>
<td>MatAdd</td>
<td>$R \leftarrow A + B$</td>
</tr>
<tr>
<td>sMM</td>
<td>$R \leftarrow \alpha AB$</td>
</tr>
<tr>
<td>MatAXPBY</td>
<td>$R \leftarrow \alpha X + \beta Y$</td>
</tr>
<tr>
<td>GEMM</td>
<td>$R \leftarrow \alpha AX + \beta Y$</td>
</tr>
</tbody>
</table>
Matrices and vectors are defined using polymorphic lists.

Definition matrix (T : Type) := list (list T).
Definition vector (T : Type) := list T.

- Coq's axiomatic reals
- Flocq's [Boldo & Melquiond, 2011]
  IEEE-754 binary floats at any precision
How are LAProof operations defined?

Matrices and vectors are defined using polymorphic lists.

```
Definition matrix (T : Type) := list (list T).
Definition vector (T : Type) := list T.
```

Operations are simple higher-order polymorphic functional programs.

```
Variable dot : vector T → vector T → T.

Definition MV : vector T := map (fun a ⇒ dot a v) A.
```
How do we ensure the correctness of LAProof operations?

Extension to the Mathematical Components (MathComp) Library [Mahboubi et al., 2022]

The Mathematical Components repository

The Mathematical Components Library is an extensive and coherent repository of formalized mathematical theories. It is based on the Coq proof assistant, powered with the Coq/SSReflect language.

- Originally developed for formal proofs of the four color theorem and the odd order theorem
- Contains well developed and maintained libraries for real analysis and basic linear algebra
How do we ensure the correctness of LAProof operations?
How do we ensure the correctness of LAProof operations?

Extension to the Mathematical Components (MathComp) Library [Mahboubi et al., 2022]

- Define injections from LAProof matrices and vectors over Coq’s axiomatic reals to MathComp’s matrices and vectors [Cohen et al., 2022]
- Prove that injections from LAProof operations to MathComp operations are correct
- Provides added confidence and functionality
How do we ensure the correctness of LAProof operations?

Extension to the Mathematical Components (MathComp) Library [Mahboubi et al., 2022]

- Define injections from LAProof matrices and vectors over Coq’s axiomatic reals to MathComp’s matrices and vectors [Cohen et al., 2022]
- Prove that injections from LAProof operations to MathComp operations are correct
- Provides added confidence and functionality

Why not use MathComp in the first place?

- Coq lists are easier to use in proofs of program correctness [Cohen et al., 2022]
- More Coq users are familiar with Coq lists than MathComp (and SSReflect)
What accuracy theorems does LAProof provide?
What accuracy theorems does LAProof provide?

- Mixed (backward-forward) rounding error bounds that account for underflow
- Low order error terms captured exactly, not approximating as $O(u^2)$
- Assume only a low-level formal model of IEEE-754 arithmetic provided by the Flocq library
  [Boldo & Melquiond, 2011]
LAProof accuracy theorems rely on the correctness of the standard rounding error model

Flocq theorem: for IEEE arithmetic,

\[ \text{fl}(a \text{ op } b) = (a \text{ op } b)(1 + \delta) + \epsilon \]

\[ |\delta| \leq u, \quad |\epsilon| \leq \eta, \quad \delta \epsilon = 0, \quad \text{op } \in \{+, -, \times, /, \sqrt{\cdot} \} \]

unit roundoff  \quad underflow unit
What accuracy theorems does LAProof provide?

- Backward error bounds when possible (e.g., summation).

\[ \phi_{F_{\text{p},e}}(x) = \phi_{R}(x + \Delta x) \]

Floating-point operation: precision \( p \), maximum exponent \( e \)

Rounding error is attributed to a small change in the input.
What accuracy theorems does LAProof provide?

- Mixed (backward-forward) rounding error bounds that account for underflow.

\[
\phi_{p,e}(x) = \phi_{\mathbb{R}}(x + \Delta x) + \hat{\delta}
\]

Floating-point operation: precision \( p \), maximum exponent \( e \)

Rounding error is attributed to a small change in the input, plus some small term that accounts for underflow.
What accuracy theorems does LAProof provide?

- Forward rounding error bounds are derived from mixed (or backward) error bounds.

\[ F \triangleq |\phi_\mathbb{R}(x) - \phi_{F_p,e}(x)| \]

Floating-point operation:
precision \( p \), maximum exponent \( e \)
An example: accuracy of matrix-vector product

- Mixed (backward-forward) rounding error bounds that account for underflow.

\[ \phi_{p,e}(x) = \phi(x + \Delta x) + \hat{\delta} \]

**Definition**

\[ \text{MV} : \text{vector } T \]
\[ := \text{map } (\text{fun } a \Rightarrow \text{dot } a \text{ v}) \text{ A.} \]

Rounding error is attributed to a small change in the input, plus some small term that accounts for underflow.
An example: accuracy of matrix-vector product

**Theorem 3 (bfMV).** For any vector \( v \in \mathbb{F}_{p,e}^n \), and matrix \( A \in \mathbb{F}_{p,e}^{m \times n} \), there exists a matrix \( \Delta A \in \mathbb{R}^{m \times n} \) and vector \( \eta \in \mathbb{R}^n \) such that

\[
fl(Av) = (A + \Delta A)v + \eta,
\]

where every element of the vector \( \eta \) respects the bound \( |\eta| \leq g(n, n) \) and each element of the matrix \( \Delta A \) respects the bound \( |\Delta A| \leq h(n)|A| \).
An example: accuracy of matrix-vector product

**Theorem 3 (bfMV).** For any vector $v \in \mathbb{F}_{p,e}^n$ and matrix $A \in \mathbb{F}_{p,e}^{m \times n}$, there exists a matrix $\Delta A \in \mathbb{R}^{m \times n}$ and vector $\eta \in \mathbb{R}^n$ such that

$$fl(Av) = (A + \Delta A)v + \eta,$$

where every element of the vector $\eta$ respects the bound $|\eta| \leq g(n, n)$ and each element of the matrix $\Delta A$ respects the bound $|\Delta A| \leq h(n)|A|$. 

$$h(n) = (1 + u)^n - 1$$

$$g(n, m) = n\eta(1 + h(m))$$
What does the theorem look like in Coq?

```coq
Variable (A: @matrix (ftype t)).
Variable (v: @vector (ftype t)).

Hypothesis Hfin : is_finite_vec (A *f v).
Hypothesis Hlen: forall row, In row A -> length row = length v.

Lemma mat_vec_mul mixed_error:
  exists (E : matrix) (eta : vector),
  A *fr v = (Ar +m E) *r vr +v eta.
  /
  (forall i j, (i < m)%nat -> (j < n)%nat ->
   Rabs (E _ (i,j)) <= g n * Rabs (Ar _ (i,j))).
  /
  (forall k, In k eta -> Rabs k <= g1 n n).
  /
  eq_size E A.
  /
  length eta = m.
```
What does the theorem look like in Coq?

```
Variable (A: @matrix (ftype t)).
Variable (v: @vector (ftype t)).

Hypothesis Hfin : is_finite_vec (A *f v).
Hypothesis Hlen: forall row, In row A -> length row = length v.

Lemma mat_vec_mul_mixed_error:
  exists (E : matrix) (eta : vector),
  A *fr v = (Ar +m E) *r vr +v eta.
  /
  (forall i j, (i < m)%nat -> (j < n)%nat ->
   Rabs (E _(i,j)) <= g n * Rabs (Ar _(i,j))).
  /
  (forall k, In k eta -> Rabs k <= gl n n).
  /
  eq_size E A.
  /
  length eta = m.
```
What does the theorem look like in Coq?

```coq
Variable (A : @matrix (ftype t)).
Variable (v : @vector (ftype t)).

Hypothesis Hfin : is_finite_vec (A *f v).
Hypothesis Hlen: forall row, In row A -> length row = length v.

Lemma mat_vec_mul_mixed_error:
  exists (E : matrix) (eta : vector),
  A *fr v = (Ar +m E) *r vr +v eta.
  \ (forall i j, (i < m)%nat -> (j < n)%nat ->
      Rabs (E (i,j)) <= g n * Rabs (Ar (i,j))).
  \ (forall k, In k eta -> Rabs k <= gl n n).
  \ eq_size E A.
  \ length eta = m.
```
What does the theorem look like in Coq?

```coq
Variable (A : @matrix (ftype t)).
Variable (v : @vector (ftype t)).

Hypothesis Hfin : is_finite_vec (A *f v).
Hypothesis Hlen: forall row, In row A -> length row = length v.

Lemma mat_vec_mul_mixed_error:
  exists (E : matrix) (eta : vector),
  A *fr v = (Ar +m E) *r vr +v eta.
  /
  (forall i j, (i < m)%nat -> (j < n)%nat ->
    Rabs (E _<(i,j)) <= g n * Rabs (Ar _<(i,j))).
  /
  (forall k, In k eta -> Rabs k <= g! n n).
  /
  eq_size E A.
  /
  length eta = m.
```
What does the theorem look like in Coq?

```coq
Variable (A: @matrix (ftype t)).
Variable (v: @vector (ftype t)).

Hypothesis Hfin : is_finite_vec (A *f v).
Hypothesis Hlen: forall row, In row A -> length row = length v.

Lemma mat_vec_mul_mixed_error:
  exists (E : matrix) (eta : vector),
  A *fr v = (Ar +m E) *r vr +v eta.
  \ (forall i j, (i < m)%nat -> (j < n)%nat ->.
    Rabs (E _.{(i,j)}) <= g n * Rabs (Ar _.{(i,j)}).
  \ (forall k, In k eta -> Rabs k <= g1 n n).
  \ eq_size E A.
  \ length eta = m.
```
What does the theorem look like in Coq?

```coq
Variable (A: @matrix (ftype t)).
Variable (v: @vector (ftype t)).

Hypothesis Hfin : is_finite_vec (A *f v).
Hypothesis Hlen: forall row, In row A -> length row = length v.

Lemma mat_vec_mulmixed_error:
  exists (E : matrix) (eta : vector),
  A *fr v = (Ar +m E) *r vr +v eta.
  \forall (forall i j, (i < m)%nat -> (j < n)%nat ->.
  Rabs (E _ (i,j)) <= g n * Rabs (Ar _ (i,j))).
  \forall (forall k, In k eta -> Rabs k <= gl n n).
  \forall eq_size E A.
  \forall length eta = m.
```
What does the theorem look like in Coq?

```
Variable (A: @matrix (ftype t)).
Variable (v: @vector (ftype t)).

Hypothesis Hfin : is_finite_vec (A *f v).
Hypothesis Hlen: forall row, In row A -> length row = length v.

Lemma mat_vec_mul_mixed_error:
  exists (E : matrix) (eta : vector),
  A *fr v = (Ar +m E) *r vr +v eta.
  \ (forall i j, (i < m)%nat -> (j < n)%nat ->.
    Rabs (E _ (i,j)) <= g n * Rabs (Ar _ (i,j))).
  \ (forall k, In k eta -> Rabs k <= g1 n n).
  \ eq_size E A.
  \ length eta = m.
```
What does the theorem look like in Coq?

```
Variable (A: @matrix (ftype t)).
Variable (v: @vector (ftype t)).

Hypothesis Hfin : isFiniteVec (A *f v).
Hypothesis Hlen: forall row, In row A -> length row = length v.

Lemma mat_vec_mul_mixed_error:
  exists (E : matrix) (eta : vector),
  A *fr v = (Ar +m E) *r vr +v eta.
  \ (forall i j, (i < m)%nat -> (j < n)%nat ->
    Rabs (E _\((i,j)) <= g n * Rabs (Ar _\((i,j)))).
  \ (forall k, In k eta -> Rabs k <= g1 n n).
  \ eq_size E A.
  \ length eta = m.
```

- Formal proof ~120 lines of code
How can we connect error bounds from LAProof to concrete programs?
Example from the paper: prove the correctness of the function csr_mv_multiply, which implements matrix-vector multiplication using a compressed sparse row (CSR) format.

```c
void csr_mv_multiply (struct csr_matrix *m,
          double *v,  double *p) {
  unsigned i, rows = m → rows;
  double *val = m → val;
  unsigned *col_ind = m → col_ind;
  unsigned *row_ptr = m → row_ptr;
  unsigned next=row_ptr[0];
  for (i = 0; i < rows; i++) {
    double s = 0.0;
    unsigned h = next;
    next = row4_ptr[i+1];
    for (h = 0; h < next; h++) {
      double x = val[h];
      unsigned j = col_ind[h];
      double y = v[j];
      s = fma(x,y,s);
    }
    p[i]=s;
  }
}
```
Example from the paper: prove the correctness of the function `csr_mv_multiply`, which implements matrix-vector multiplication using a compressed sparse row (CSR) format.

- Write a specification of matrix-vector multiplication using the LAProof operation.
- Prove (in Coq) that `csr_mv_multiply` complies with this specification.
Example: Prove the correctness of `csr_mv_multiply` using VST

The Verified Software Toolchain (VST) [Appel et al., 2011]

- A collection of verification tools for the C language
- Implements (in Coq) a program logic for reasoning about the correctness of C programs
- Proved sound with respect to the CompCert C compiler [Leroy et al., 2008]

```coq
Definition csr_mv_spec :=
DECLARE _csr_mv_multiply
WITH π₁: share, π₂: share, π₃: share,
    m: val, A: matrix Tdouble, v: val,
    x: vector Tdouble, p: val
PRE [ tptr t_csr, tptr tdouble, tptr tdouble ]

POST [ tvoid ]
EX y: vector Tdouble,
    PROP(Forall2 (eq y (MVF A x)))
RETURN()
SEP (csr_rep π₁ A m;
    data_at π₂ (tarray tdouble (Zlength x))
    (map Vfloat x) v;
    data_at π₃ (tarray tdouble (matrix_rows A))
    (map Vfloat y) p).
```
Example: Prove the correctness of csr_mv_multiply using VST

A function is specified by its precondition and its postcondition.
Example: Prove the correctness of \texttt{csr_mv_multiply} using VST

A function is specified by its precondition and its postcondition

- \(A, x\): formal models of the matrix and vector begin multiplied.

\begin{verbatim}
Definition csr_mv_spec :=
DECLARE _csr_mv_multiply
WITH \(\pi_1\): share, \(\pi_2\): share, \(\pi_3\): share,
  \(m\): val, \(A\): matrix Tdouble, \(v\): val,
  \(x\): vector Tdouble, \(p\): val
PRE [ tptr t_csr, tptr tdouble, tptr tdouble ]

POST [ tvoid ]
EX \(y\): vector Tdouble,
  PROP(Forall2 eq \(y\) (MVF \(A\ \ x\)))
RETURN()
SEP (csr_rep \(\pi_1\) \(A\ \ m\);
  data_at \(\pi_2\) (tarray tdouble (Zlength \(x\)))
    (map Vfloat \(x\)) \(v\);
  data_at \(\pi_3\) (tarray tdouble (matrix_rows \(A\)))
    (map Vfloat \(y\)) \(p\).
\end{verbatim}
A function is specified by its **precondition** and its **postcondition**

- **A, x**: formal models of the matrix and vector begin multiplied.
- **m**: address where CSR representation of A is stored
- **p**: address where vector x is stored

**Example: Prove the correctness of csr_mv_multiply using VST**

```plaintext
Definition csr_mv_spec :=
DECLARE _csr_mv_multiply
WITH π₁: share, π₂: share, π₃: share,
   m: val, A: matrix Tdouble, v: val,
   x: vector Tdouble, p: val
PRE [ tptr t_csr, tptr tdouble, tptr tdouble ]

POST [ tvoid ]
EX y: vector Tdouble,
   PROP(Forall2 feq y (MVF A x))
RETURN()
SEP (csr_rep π₁ A m;
   data_at π₂ (tarray tdouble (Zlength x))
   (map Vfloat x) v;
   data_at π₃ (tarray tdouble (matrix_rows A))
   (map Vfloat y) p).
```
A function is specified by its **precondition** and its **postcondition**

- A, x: formal models of the matrix and vector begin multiplied.
- m: address where CSR representation of A is stored
- p: address where vector x is stored
- **postcondition**: the vector of y of double precision floats exists, and...

---

**Example: Prove the correctness of csr_mv_multiply using VST**

```plaintext
Definition csr_mv_spec :=
DECLARE _csr_mv_multiply
WITH π₁: share, π₂: share, π₃: share,
  m: val, A: matrix Tdouble, v: val,
  x: vector Tdouble, p: val
PRE [ tptr t_csr, tptr tdouble, tptr tdouble ]

POST [ tvoid ]
EX y: vector Tdouble,
PROP(Forall2 #eq y (MVF A x))
RETURN()
SEP (csr_rep π₁ A m;
  data_at π₂ (tarray tdouble (Zlength x))
  (map Vfloat x) v;
  data_at π₃ (tarray tdouble (matrix_rows A))
  (map Vfloat y) p).
```
Accuracy and correctness proofs compose

Theorem [accuracy and correctness]: the function csr_mv_multiply correctly and accurately implements matrix-vector multiplication using a compressed sparse row format.

```c
void csr_mv_multiply (struct csr_matrix *m,
                      double *v,
                      double *p) {
  unsigned i, rows = m->rows;
  double *val = m->val;
  unsigned *col_ind = m->col_ind;
  unsigned *row_ptr = m->row_ptr;
  unsigned next=row_ptr[0];
  for (i = 0; i < rows; i++) {
    double s = 0.0;
    unsigned h = next;
    next = row_ptr[i+1];
    for (h = 0; h < next; h++) {
      double x = val[h];
      unsigned j = col_ind[h];
      double y = v[j];
      s = fma(x,y,s);
    }
    p[i]=s;
  }
}
```
In summary,

LAProof provides machine-checked proofs of accuracy for basic linear algebra operations and these accuracy proofs can be connected to concrete programs implementing BLAS.

- Accuracy proofs assume only a low-level formal model of IEEE-754 arithmetic.
- The rounding error bounds in the accuracy proofs are mixed (backward-forward) error bounds that account for underflow.
- Rounding error bounds capture low order error terms exactly, not approximating as $O(u^2)$. 
Thanks for listening!

My co-authors are Andrew W. Appel, Mohit Tekriwal, and David Bindel

We acknowledge the generous support of

[Logos of PRINCETON UNIVERSITY, UNIVERSITY OF MICHIGAN, and Cornell University]