Formal Proofs of Rounding Error Bounds

Pierre Roux

Noname manuscript No.
(will be inserted by the editor)

Formal Proofs of Rounding Error Bounds

With application to an automatic positive definiteness check.

Pierre Roux

Received: date / Accepted: date

Abstract Floating-point arithmetic is a very efficient solution to perform computations in the real field. However, it induces rounding errors making results computed in floating-point differ from what would be computed with reals. Although numerical analysis gives tools to bound such differences, the proofs involved can be painful, hence error prone. We thus investigate the ability of a proof assistant like Coq to mechanically check such proofs. We demonstrate two different results involving matrices, which are pervasive among numerical algorithms, and show that a large part of the development effort can be shared between them.

Keywords floating-point arithmetic · rounding error · numerical analysis · proof assistant · Coq · matrices · Cholesky decomposition

1 Introduction

Floating-point arithmetic is a very efficient solution to perform computations in the real field $\mathbb{R}$. Unfortunately, intermediate results of computations need to be rounded to fit in the floating-point format used. Due to this rounding errors, final results of computations differ from what would have been obtained by computing in the real field $\mathbb{R}$, although both results usually remain pretty close.

Fortunately, each rounding can only introduce a bounded error. By combining these atomic errors, one can get a bound on the error affecting the final result. Numerical analysis [11] thus aims at bounding these differences between results of numerical algorithms using floating-point or real arithmetic. Using such mathematical properties, rigorous results can be obtained despite the use of floating-point arithmetic [15], ensuring that no disastrous rounding error can happen during the computation.

This work was done while the author was a visiting researcher at LRI, Inria Saclay – Île-de-France.

Pierre Roux
ISAE, ONERA
E-mail: pierre.roux@onera.fr

R **:=** 0;
for j **from** 1 **to** n **do**
  for i **from** 1 **to** $j-1$ **do**
    $R_{i,j}$ **:=** $\left(A_{i,j} - \sum_{k=1}^{i-1} R_{k,i} R_{k,j}\right) / R_{i,i}$;
  od
  $R_{j,j}$ **:=** $\sqrt{M_{j,j} - \sum_{k=1}^{j-1} {R_{k,j}}^2}$;
od

Fig. 1 Cholesky decomposition: from a matrix $A \succeq 0$, computes R such that $A = R^T R$.

More precisely, denoting f a function in the real field $\mathbb{R}$ and $\tilde{f}$ its actually computed floating-point counterpart, we have $\tilde{f}(x) = f(x) + e$. The value e is called the *forward error* and is expected to be negligible in front of $f(x)$. When $\tilde{f}(x) = f(x+d)$, d is called a *backward error*. This paper focuses on proofs of forward error bounds $b(x)$ such that $|e| \leq b(x)$.

Proofs of this kind of mathematical results are hard to automate when they involve an arbitrary number of operations and are therefore mostly done by hand. However, they can be particularly painful and repetitive which make them specially error prone. That's why we want to investigate the ability of a proof assistant, namely Coq [1,6], to check them. Matrices being pervasive in numerical algorithms, we will particularly focus on them.

Formal proofs of error bounds have already been performed with proof assistants such as HOL [10] or Coq [3] or with automatic tools such as Gappa [7]. Yet, to the extent of author's knowledge, those work only address results with a fixed number of basic arithmetic operations whereas algorithms with an arbitrary, parameterized, number of operations are targeted in this paper.

The paper is organized as follows. The remainder of this section first introduces our motivating example that will be used throughout the paper (Section 1.1), then gives basic properties of floating-point arithmetic (Section 1.2) and eventually details a simple proof about summations (Section 1.3). Section 2 then gives the detailed specification of floating-point arithmetic used while Section 3 shows how error terms can be combined. Section 4 eventually details proofs of numerical analysis results involving matrices and Section 5 concludes.

1.1 Motivating Example: Cholesky Decomposition

We will use as motivating example throughout this paper a Cholesky decomposition which is a typical example of numerical algorithm involving matrices. Checking positive definiteness of matrices is one common use of Cholesky decomposition. A matrix $A \in \mathbb{R}^{n \times n}$ is said positive semi-definite $A \succeq 0$ when, for all $x \in \mathbb{R}^n$, $x^T A x \geq 0$.

To prove that a scalar $a \in \mathbb{R}$ is non negative, one can exhibit some $r \in \mathbb{R}$ such that $a = r^2$ (typically $r = \sqrt{a}$). Similarly, one can prove that a matrix $A \in \mathbb{R}^{n \times n}$ is positive semi-definite by exposing a matrix R such that $A = R^T R$ (for $x^T (R^T R) x = (Rx)^T (Rx) = \|Rx\|_2^2 \geq 0$ for all $x \in \mathbb{R}^n$). The Cholesky decomposition is an algorithm

that compute such a matrix R (c.f., Figure 1). It is interesting to notice that the actual value of r or R doesn't matter. It is enough to prove it exists. Indeed, if the Cholesky decomposition of A runs to completion, without ever attempting to take the square root of a negative value or perform a division by zero, hence produce a R, this proves $A \succeq 0$.

This would work perfectly if the algorithm were run using real arithmetic. However, performing it with floating-point arithmetic, it could run to completion while $A \not\succeq 0$, due to rounding errors. But rounding errors remain bounded, so that there exists a $c \in \mathbb{R}$ such that, if the floating-point Cholesky decomposition of A succeeds, then $A + c\,Id \succeq 0$. The successful floating-point Cholesky decomposition of $A - c\,Id$ eventually proves that $A \succeq 0$. Moreover, such a constant c can be easily computed from simple characteristics of A and the floating-point arithmetic format used. The goal of this paper will be to prove this.

1.2 Definitions and Basic Properties

Definition 1.1 $\mathbb{F} \subset \mathbb{R}$ denotes the set of floating point values, round : $\mathbb{R} \to \mathbb{F}$ a rounding function (toward $+\infty$ or to nearest for instance) and $\mathrm{fl}(e) \in \mathbb{F}$ the floating point evaluation of expression e from left to right[1].

Example 1.2 Assuming 1, 2 and 3 are floating-point values, $\mathrm{fl}(1+2+3)$ denotes the value $\mathrm{round}(\mathrm{round}(1+2)+3)$.

Definition 1.3 $\texttt{eps} \in \mathbb{R}$ and $\texttt{eta} \in \mathbb{R}$ are constants, depending from the floating-point format used, such that for all $x \in \mathbb{R}$, $\mathrm{round}(x) \in \mathbb{F}$ satisfies either $|x - \mathrm{round}(x)| \leq \texttt{eps}\,|x|$ or $|x - \mathrm{round}(x)| \leq \texttt{eta}$.

Example 1.4 For the IEEE754 [12] binary64[2] format with round a rounding to nearest, we have $\texttt{eps} = 2^{-53}$ ($\simeq 10^{-16}$) and $\texttt{eta} = 2^{-1075}$ ($\simeq 10^{-323}$).

These constants allow to bound the rounding errors of the basic arithmetic operations.

Property 1.5 For all $x, y \in \mathbb{F}$

$$\begin{aligned}
&\exists \delta \in \mathbb{R}, |\delta| \leq \texttt{eps} \wedge \mathrm{fl}(x \diamond y) = (1+\delta)(x \diamond y), && \text{for } \diamond \in \{+,-\}\\
&\exists \delta, \eta \in \mathbb{R}, |\delta| \leq \texttt{eps} \wedge |\eta| \leq \texttt{eta} \wedge \mathrm{fl}(x \diamond y) = (1+\delta)(x \diamond y) + \eta, && \text{for } \diamond \in \{\times,/\}\\
&\exists \delta \in \mathbb{R}, |\delta| \leq \texttt{eps} \wedge \mathrm{fl}\left(\sqrt{x}\right) = (1+\delta)\sqrt{x}.
\end{aligned}$$

1.3 Simple Example: the Sum

The previous bounds on rounding errors of basic operations can be combined to get bounds on the error of larger expressions, as for instance a summation in the following classic result [11,15].

[1] Order of evaluation matters since floating point operations are not associative.

[2] Usual implementation of the type `double` in C.

Theorem 1.6 *For all $x \in \mathbb{F}^n$*

$$\left| \mathrm{fl}\left(\sum_{i=1}^{n} x_i\right) - \sum_{i=1}^{n} x_i \right| \leq \gamma_{n-1} \sum_{i=1}^{n} |x_i|$$

where $\gamma_{n-1} := \frac{(n-1)\mathtt{eps}}{1-(n-1)\mathtt{eps}}$.

Proof We have by direct application of Property 1.5

$$\mathrm{fl}\left(\sum_{i=1}^{n} x_i\right) = \mathrm{fl}\left(\mathrm{fl}\left(\sum_{i=1}^{n-1} x_i\right) + x_n\right) = (1+\delta_n)\left(\mathrm{fl}\left(\sum_{i=1}^{n-1} x_i\right) + x_n\right)$$

for some $\delta_n \in \mathbb{R}$, $|\delta_n| \leq \mathtt{eps}$. Then

$$\mathrm{fl}\left(\sum_{i=1}^{n} x_i\right) = (1+\delta_n)\left((1+\delta_{n-1})\left(\mathrm{fl}\left(\sum_{i=1}^{n-2} x_i\right) + x_{n-1}\right) + x_n\right)$$

for some $\delta_{n-1} \in \mathbb{R}$, $|\delta_{n-1}| \leq \mathtt{eps}$. By an immediate induction

$$\mathrm{fl}\left(\sum_{i=1}^{n} x_i\right) = \left(\prod_{j=2}^{n}(1+\delta_j)\right) x_1 + \sum_{i=2}^{n}\left(\prod_{j=i}^{n}(1+\delta_j)\right) x_i \tag{1}$$

for some $\delta_j \in \mathbb{R}$, $\left|\delta_j\right| \leq \mathtt{eps}$. Then [11, Lemma 3.1], for all $i \in [\![2,n]\!]$, there exist $\theta_i \in \mathbb{R}$ such that $|\theta_i| \leq \gamma_{n-i+1} := \frac{(n-i+1)\mathtt{eps}}{1-(n-i+1)\mathtt{eps}}$ and $\prod_{j=i}^{n}(1+\delta_j) = 1+\theta_i$, hence

$$\mathrm{fl}\left(\sum_{i=1}^{n} x_i\right) = (1+\theta_2)x_1 + \sum_{i=2}^{n}(1+\theta_i)x_i = \sum_{i=1}^{n} x_i + \left(\theta_2 x_1 + \sum_{i=2}^{n} \theta_i x_i\right). \tag{2}$$

Finally, since for all $i \in [\![2,n]\!]$, $n-i+1 \leq n-1$ and $k \mapsto \gamma_k$ is monotone, we have $|\theta_i| \leq \gamma_{n-i+1} \leq \gamma_{n-1}$, and there exists $\theta \in \mathbb{R}$ such that $|\theta| \leq \gamma_{n-1}$ and

$$\mathrm{fl}\left(\sum_{i=1}^{n} x_i\right) = \sum_{i=1}^{n} x_i + \theta \sum_{i=1}^{n} |x_i| \tag{3}$$

which enables to conclude.

2 Specification of Floating-Point Arithmetic

As seen in Property 1.5 and Theorem 1.6, both definitions and proofs make intensive use of real numbers with a bounded absolute value. To ease the manipulation of these error terms, we use a dependent record `bounded b` packing a real number with a proof that its absolute value is less than a non-negative real number `b`:

```
Record bounded (b : ℝ) :=
{ bounded_val :> ℝ; bounded_prop : |bounded_val| ≤ b }.
```

Since the set of floating-point values $\mathbb{F}$ is a subset of $\mathbb{R}$, we will similarly define floating point values as a value in $\mathbb{R}$ along with a proof that it lies in $\mathbb{F}$:

```
Record Ff format :=
{ F_val :> ℝ; F_prop : format F_val }.
```

where `format` is a predicate over $\mathbb{R}$ identifying real numbers that are in $\mathbb{F}$.

The floating-point arithmetic specification is then given by the following large record which will be used as parameter of all our subsequent developments.

```
Record Float_spec := {
(** format x means that x ∈ ℝ is a floating-point value *)
format : ℝ → Prop;
(** The type of floating-point values (coercible to ℝ). *)
F := Ff format;
(** 0 and 1 must be floating-point numbers. *)
format0 : format 0; format1 : format 1;
(** Bound on the relative error (normalized numbers, no underflow). *)
eps : ℝ; eps_pos : 0 ≤ eps; eps_lt_1 : eps < 1;
(** Bound on the absolute error (denormalized, when underflow occurs). *)
eta : ℝ; eta_pos : 0 < eta;
(** Some rounding. *)
frnd : ℝ → 𝔽; frnd_spec : ∀x : ℝ,
∃d : bounded eps, ∃e : bounded eta, frnd x = (1+d)x+e;
(** Addition. *)
fplus : 𝔽 → 𝔽 → 𝔽; fplus_spec : ∀x y : 𝔽,
∃d : bounded eps, fplus x y = (1+d)(x+y);
(** Opposite. *)
fopp : 𝔽 → 𝔽; fopp_spec : ∀x : 𝔽, fopp x = -x;
(** Subtraction. *)
fminus : 𝔽 → 𝔽 → 𝔽; fminus_spec : ∀x y : 𝔽,
fminus x y = fplus x (fopp y);
fminus_spec2 : ∀x y : 𝔽, 0 ≤ y → fminus x y ≤ x;
(** Multiplication. *)
fmult : 𝔽 → 𝔽 → 𝔽; fmult_spec : ∀x y : 𝔽,
∃d : bounded eps, ∃e : bounded eta, fmult x y = (1+d)(x×y)+e;
fmult_spec2 : ∀x : 𝔽, 0 ≤ fmult x x;
(** Division. *)
fdiv : 𝔽 → 𝔽 → 𝔽; fdiv_spec : ∀x y : 𝔽, y ≠ 0 →
∃d : bounded eps, ∃e : bounded eta, fdiv x y = (1+d)(x/y)+e;
(** Square root. *)
fsqrt : 𝔽 → 𝔽; fsqrt_spec : ∀x y : 𝔽, 0 ≤ x →
∃d : bounded eps, fsqrt x = (1+d)√x;
fsqrt_spec2 : ∀x : 𝔽, 0 < fsqrt x → 0 < x;
}.
```

Having performed our proofs with a proof assistant, we are guaranteed that the above record contains all the hypotheses about floating-point arithmetic used in these proofs. It is interesting to notice that this specification of floating-point arithmetic is really broad. In particular, it encompasses floating-point formats with gradual or abrupt underflow and any rounding mode. Fixed-point arithmetic can even be handled by just setting `eps` to 0, i.e., no relative, only absolute error occur. However, most of our developments are carried on with floating-point arithmetic in mind and the proved bounds might be pretty poor in a fixed-point arithmetic setting.

It is common practice in numerical analysis to ignore underflows [11]. Although this gives good indications on the numerical behavior of algorithms, underflows can appear with any practical implementation of floating-point arithmetic, potentially breaking such results. In our development, they are taken into account, thanks to the `eta` constant.

In our Coq development, the above specification of floating-point arithmetic is proven to hold for the floating point format with gradual underflow and any rounding to nearest modeled in the Flocq library [4] with parameters corresponding to the binary64 format[2] (albeit without NaNs nor overflows). Other formats such as binary32[3] could be obtained by just modifying two constants defining size of the mantissa and minimal exponent.

In contrary to underflows, not handling NaNs and overflows does not constitute an actual issue. In fact, results considering those special values can easily be derived from results in our model with only finite values.

3 Combining Error Terms

3.1 Bounded Error Terms

Values of the type `bounded` defined at beginning of Section 2 are coercible to $\mathbb{R}$ and we developed a few helpful lemmas about them. The two following lemmas can be used to create such values.

Lemma 3.1 (`bounded_le_1`) $\forall x,y \in \mathbb{R}, |x| \leq y \Rightarrow \exists b : \texttt{bounded}\ 1,\ x = b\,y$

Lemma 3.2 (`bounded_scale`) $\forall b,b' \in \mathbb{R}, r : \texttt{bounded}\ b, 0 < b' \Rightarrow \exists r' : \texttt{bounded}\ b',\ r = r'\frac{b}{b'}$

It is often needed to say that a value of type `bounded` b is also of type `bounded` b' for any $b' \geq b$ (for instance, in proof of Theorem 1.6, to state that the $\theta_i : \texttt{bounded}\ \gamma_{n-i+1}$ are all of type `bounded` γ_{n-1}):

Lemma 3.3 (`bounded_widen`) $\forall b,b' \in \mathbb{R}, \forall x : \texttt{bounded}\ b, b \leq b' \Rightarrow \exists x' : \texttt{bounded}\ b',\ x = x'$

It is also common to get bounds of the form $x\,e$ with $x : \texttt{bounded}\ b$ and e a complicated expression we want to replace by a simpler expression $e' \geq e$:

[3] Usual implementation of type `float` in C.

Lemma 3.4 (`bounded_larger_factor`) $\forall b, r_1, r_2 \in \mathbb{R}, \forall x : \texttt{bounded}\, b, |r_1| \leq |r_2| \Rightarrow \exists x' : \texttt{bounded}\, b,\ x\, r_1 = x'\, r_2$

Error terms are compatible with basic arithmetic operations:

Lemma 3.5

$\forall b : \mathbb{R}, \forall x : \texttt{bounded}\, b, \exists x' : \texttt{bounded}\, b,\ x' = -x$

$\forall b_1, b_2 : \mathbb{R}, \forall x_1 : \texttt{bounded}\, b_1, \forall x_2 : \texttt{bounded}\, b_2, \exists x' : \texttt{bounded}\ (b_1 + b_2),\ x' = x_1 + x_2$

$\forall b_1, b_2 : \mathbb{R}, \forall x_1 : \texttt{bounded}\, b_1, \forall x_2 : \texttt{bounded}\, b_2, \exists x' : \texttt{bounded}\ (b_1\, b_2),\ x' = x_1\, x_2$

Finally, probably the most important lemma about error terms allows to factor them and was exemplified between (2) and (3):

Lemma 3.6 (`bounded_distrl`, `big_bounded_distrl`)

$$\forall b, r_1, r_2 \in \mathbb{R}, \forall x_1, x_2 : \texttt{bounded}\, b, \exists x' : \texttt{bounded}\, b,\ x_1\, r_1 + x_2\, r_2 = x'\, (|r_1| + |r_2|)$$

$$\forall b, \in \mathbb{R}, \forall k : \mathbb{N}, \forall r \in \mathbb{R}^k, \forall x : (\texttt{bounded}\, b)^k, \exists x' : \texttt{bounded}\, b,\ \sum_i x_i\, r_i = x' \left(\sum_i |r_i| \right)$$

It is worth noting that this last property involves tuples (r and x) and sums of an arbitrary number of terms ($\sum_i r_i x_i$ for instance). Those are efficiently handled thanks to the `bigop` operator from the SSReflect library [2].

3.2 Accumulating Relative Errors

As already seen, for instance in the proof of Theorem 1.6 (between (1) and (2)), error terms of the form $(1+\delta_1)\dots(1+\delta_n)$, with $|\delta_i| \leq \texttt{eps}$, easily occur when relative errors accumulate. The terms $\gamma_k := \frac{k\texttt{eps}}{1-k\texttt{eps}}$ nicely enable to compact them into $1+\theta_n$, $|\theta_n| \leq \gamma_n$ as will be exposed in this section.

Most of the following lemmas require hypothesis of the form $k\,\texttt{eps} < 1$ for various values of k. In our Coq code, a bunch of small lemmas[4] allow to easily manipulate theses hypothesis so that they do not constitute an annoying burden in practice. First, a few very basic properties of the γ_k are proved. Namely, that they are non negative (provided $k\,\texttt{eps} < 1$), strictly less than 1 (provided $2k\,\texttt{eps} < 1$) and constitute a monotone sequence: for all $k \leq k'$, $\gamma_k \leq \gamma_{k'}$ (provided $k'\,\texttt{eps} < 1$).

We then get some more interesting properties.

Lemma 3.7 *[11, Lemma 3.3]. For all $k, k' \in \mathbb{N}$*

$$k \leq k' \Rightarrow 2k'\,\texttt{eps} < 1 \Rightarrow \gamma_k\, \gamma_{k'} \leq \gamma_k$$
$$(k\,k')\texttt{eps} < 1 \Rightarrow k\, \gamma_{k'} \leq \gamma_{k\,k'}$$
$$(k+k')\texttt{eps} < 1 \Rightarrow \gamma_k + \gamma_{k'} + \gamma_k\, \gamma_{k'} \leq \gamma_{k+k'}$$
$$(k+k')\texttt{eps} < 1 \Rightarrow \gamma_k + \gamma_{k'} \leq \gamma_{k+k'}$$
$$(k+1)\texttt{eps} < 1 \Rightarrow \gamma_k + \texttt{eps} \leq \gamma_{k+1}$$

[4] For instance: $(k+1)\texttt{eps} < 1 \Rightarrow k\,\texttt{eps} < 1$.

Allowing to prove properties about the $\theta_k : \texttt{bounded}\ \gamma_k$.

Lemma 3.8 *[11, Lemma 3.3]. For all $k,k' \in \mathbb{N}$, for all $\theta_k : \texttt{bounded}\ \gamma_k$, $\theta_{k'} : \texttt{bounded}\ \gamma_{k'}$ and $\delta : \texttt{bounded}\ \texttt{eps}$*

$$
\begin{aligned}
&2(k+k')\texttt{eps} < 1 \Rightarrow \\
&\qquad \exists \theta_{k+k'} : \texttt{bounded}\ \gamma_{k+k'},\ (1+\theta_k)(1+\theta_{k'}) = 1+\theta_{k+k'} \\
&2(k+1)\texttt{eps} < 1 \Rightarrow \\
&\qquad \exists \theta_{k+1} : \texttt{bounded}\ \gamma_{k+1},\ (1+\theta_k)(1+\delta) = 1+\theta_{k+1} \\
&k' \leq k \Rightarrow 2(k+k')\texttt{eps} < 1 \Rightarrow \\
&\qquad \exists \theta_{k+k'} : \texttt{bounded}\ \gamma_{k+k'},\ (1+\theta_k)/(1+\theta_{k'}) = 1+\theta_{k+k'} \\
&2(k+2k')\texttt{eps} < 1 \Rightarrow \\
&\qquad \exists \theta_{k+2k'} : \texttt{bounded}\ \gamma_{k+2k'},\ (1+\theta_k)/(1+\theta_{k'}) = 1+\theta_{k+2k'}
\end{aligned}
$$

For a set of values $d_k : \texttt{bounded}\ \texttt{eps}$, we also define $\phi_{i,j,d} := \prod_{k=i}^{j-1}(1+d_k)$ which satisfy the following properties:

Lemma 3.9 *For all $i,j,n \in \mathbb{N}$, $d : (\texttt{bounded}\ \texttt{eps})^n$*

$$
\begin{aligned}
&2(j-i)\texttt{eps} < 1 \Rightarrow \exists \theta_{j-i} : \texttt{bounded}\ (j-i),\ \phi_{i,j,d} = 1+\theta_{j-i} \\
&(j-i)\texttt{eps} < 1 \Rightarrow \exists \theta_{j-i} : \texttt{bounded}\ (j-i),\ \frac{1}{\phi_{i,j,d}} = 1+\theta_{j-i}
\end{aligned}
$$

It is interesting to notice about the division [11, §3.4] that, although (according to Lemma 3.8) $(1+\theta_k)/(1+\theta_{k'}) = 1+\theta_{k+k'}$ only holds for $k' \leq k$, according to the above lemma $\phi_{0,k,d}/\phi_{0,k',d'} = 1+\theta_{k+k'}$ even when $k' > k$.

The notations δ and θ_k, with $|\delta| \leq \texttt{eps}$ and $|\theta_k| \leq \gamma_k$, used in the above lemmas are particularly convenient and popular to carry proofs about error bounds. In fact, like the Landau big O notation, they greatly simplify proofs by enabling the use of simple equalities[5] instead of a bunch of inequalities, or limits. However, it is easy to misuse them or forgot hypotheses, such as $k' \leq k$ in Lemma 3.8. The use of a proof assistant ensures that this does not happen.

3.3 First Applications

Thanks to all the above lemmas, the Theorem 1.6, bounding the rounding error of a sum, is easily proven (c.f., `fsum_l2r_err_abs` in our Coq development). Similar results are also proved for the dotproduct of two vectors of floating point values.

Lemma 3.10 (`fdotprod_l2r_err_abs`) *For all $n \in \mathbb{N}$ and $a,b \in \mathbb{F}^n$, if $2k\,\texttt{eps} < 1$, then*

$$
\left| \mathrm{fl}\left(\sum_{i=1}^{n} a_i b_i \right) - \sum_{i=1}^{n} a_i b_i \right| \leq \gamma_n \left(\sum_{i=1}^{n} |a_i b_i| \right) + 2n\,\texttt{eta}.
$$

[5] See for instance the proof of Theorem 1.6, page 4.

Another example, in case the first operand a is constituted of real numbers, which have to be rounded to floating-point values prior to computation of the dotproduct:

Lemma 3.11 (`fdotprod_l2r_fstr_err`) *For all $n \in \mathbb{N}$, $a \in \mathbb{R}^n$ and $b \in \mathbb{F}^n$, if $2(k+1)\texttt{eps} < 1$, then*

$$\left| \mathrm{fl}\left(\sum_{i=1}^{n} a_i b_i\right) - \sum_{i=1}^{n} a_i b_i \right| \leq \gamma_{n+1}\left(\sum_{i=1}^{n} |a_i b_i|\right) + 2\left(n + \sum_{i=1}^{n} |b_i|\right)\texttt{eta}.$$

Due to the presence of existential quantificators in most intermediate lemmas, proof style in the Coq proof assistant heavily rely on forward proving. This does not appear to add much burden to the proof writing process, as long as proofs are well structured into lemmas of reasonable size[6]. Otherwise, one can first provide some dummy term and later step back to replace it by the, then more obvious, adequate term. Using the `evar` mechanism of Coq might also be a solution, as done with "big enough numbers" for ε, η proofs about limits [5]. However, it is probably not worth translating the latter in our setting since this would be rather complicated to solve what is not an actual problem.

4 Errors on Matrix Operations

4.1 Real Numbers Matrices

As stated in the introduction (Section 1.1), we intend to prove numerical analysis results on algorithms involving matrices. In our Coq development, we borrow matrix algebra to the SSReflect library [9]. But we also need some results which are specific to matrices of real numbers. We therefore introduce some basic definitions and lemmas about pointwise orders and absolute values, dotproducts and quadratic norms.

First, the pointwise extensions of the order $\leq$ and $<$ as well as the absolute value $|.|$ are defined and a bunch of lemmas are proved about them. Most of these lemmas are just lifting of the existing results on the real field $\mathbb{R}$: reflexivity and transitivity of the order $\leq$, compatibility of this order with matrix addition or scaling, triangular inequality of the absolute value ($\forall A, B \in \mathbb{R}^{n\times m}, |A+B| \leq |A| + |B|$),...

In order to deal with quadratic norms, we first define *positive (semi-)definite* matrices. A matrix $P \in \mathbb{R}^{n\times n}$ is said positive semi-definite, written $P \succeq 0$, when for all $x \in \mathbb{R}^n$, $x^T P x \geq 0$ and it is said positive definite, written $P \succ 0$, when for all $x \neq 0$, $x^T P x > 0$. Thus for a symmetric ($P^T = P$) positive definite matrix P, we define the *dotproduct* of two vectors $x, y \in \mathbb{R}^n$ as $x^T P y$ and the quadratic norm $\|x\|_P$ as $\sqrt{x^T P x}$. The dotproduct is then proven to actually be a dotproduct (bilinear, symmetric, definite and non negative). It follows that the quadratic norm is definite non negative and satisfies the scaling property ($\|\lambda x\|_P = |\lambda| \, \|x\|_P$) which eventually enables to prove two usual inequalities: the Cauchy-Schwartz inequality:

$$\forall x, y \in \mathbb{R}^n, \ \left|x^T P y\right| \leq \|x\|_P \, \|y\|_P$$

[6] Which is just good programming practice.

and the triangular inequality:

$$\forall x, y \in \mathbb{R}^n,\ \|x+y\|_P \leq \|x\|_P + \|y\|_P.$$

In the particular case when $P := Id$, the quadratic norm will be written $\|.\|_2$ and a few additional properties are proved: $\forall x, y \in \mathbb{R}^n, \forall c \in \mathbb{R}$,

$$\begin{aligned}
&|x| \leq |y| \Rightarrow \|x\|_2 \leq \|y\|_2\\
&\| |x| \|_2 = \|x\|_2\\
&\left\| [c \dots c]^T \right\|_2 = \sqrt{n}\,|c|\\
&\|x\|_1 \leq \sqrt{n}\|x\|_2
\end{aligned}$$

where $\|x\|_1 := \sum_i |x_i|$.

We eventually needed 101 lemmas. Thanks to the nice SSReflect matrices [9], they are proved using only 426 lines of tactics (hence an average of 4.2 lines of tactic per lemma, the longest proof being the Cauchy-Schwartz inequality with 40 lines of tactic).

4.2 Main Application: Cholesky Decomposition

As explained in Section 1.1, given a matrix M we want to check its positive definiteness, that is to prove $M \succ 0$. This will be done by proving that there exists a constant $c \in \mathbb{R}$ such that when the Cholesky decomposition (c.f., Figure 1) of $M - c\,Id$, performed with floating-point arithmetic, runs to completion without error (square root of negative value or division by zero), then $M \succ 0$. We follow the proof in [14][7].

The first lemmas proved deal with the two "basic blocks" of the Cholesky decomposition: the assignments performed in the inner then the outer loop (c.f., Figure 1, page 2). The two following lemmas are proved with tools similar to the one required for Lemmas 3.10 and 3.11 about floating-point sums and dotproducts.

Lemma 4.1 ([14, Lemma 2.1]) *For all $n \in \mathbb{N}$, $a, b \in \mathbb{F}^n$, $c, d \in \mathbb{F}$, if $d \neq 0$ and $2(n+1)\mathtt{eps} < 1$, then*

$$\left| c - \sum_i a_i b_i - d\,\tilde{y} \right| \leq \gamma_{n+1} \left(\sum_i |a_i b_i| + |d\,\tilde{y}| \right) + 2\mathtt{eta}\,(k+1+|d|)\,.$$

where $\tilde{y} := \mathrm{fl}(c - \sum_i a_i b_i)$.

Lemma 4.2 ([14, Lemma 2.2]) *For all $n \in \mathbb{N}$, $a \in \mathbb{F}^n$, $c \in \mathbb{F}$, if $2(n+2)\mathtt{eps} < 1$ and $\mathrm{fl}\left(c - \sum_i a_i^2\right) \geq 0$, then*

$$\left| c - \sum_i a_i^2 - \tilde{y}^2 \right| < \gamma_{n+2} \left(\sum_i a_i^2 + \tilde{y}^2 \right) + 2\mathtt{eta}\,(k+1)\,.$$

[7] Actually, part of it. We only consider matrices of real numbers whereas [14] also handles complex numbers. [14] also offers improved bounds for sparse matrices and a non-positive-definiteness check.

and

$$\tilde{y}^2 + \sum_i a_i^2 \leq \frac{c + 2\texttt{eta}k}{1 - \gamma_{n+2}}$$

where $\tilde{y} := \text{fl}\left(\sqrt{c - \sum_i a_i^2}\right)$.

Then, given two matrices $A, \tilde{R} \in \mathbb{F}^{n\times n}$ the proposition `cholesky_spec` A $\tilde{R}$ expresses that $\tilde{R}$ is the floating-point Cholesky factor of A and is defined as follows

$$\begin{aligned}&\forall i, j \in [\![1,n]\!], i < j \Rightarrow \tilde{R}_{i,j} = \text{fl}\left(\frac{A_{i,j} - \sum_k \tilde{R}_{k,i}\tilde{R}_{k,j}}{\tilde{R}_{i,i}}\right)\\ &\wedge \forall j \in [\![1,n]\!], \tilde{R}_{j,j} = \text{fl}\left(\sqrt{A_{j,j} - \sum_k \tilde{R}_{k,j}^2}\right).\end{aligned}$$

Then, the proposition `cholesky_success` A $\tilde{R}$ states that the floating-point Cholesky decomposition of A runs to completion without error (and returns $\tilde{R}$):

$$\texttt{cholesky\_spec}\ A\ \tilde{R} \wedge \forall i \in [\![1,n]\!], \tilde{R}_{i,i} > 0.$$

With this specification of the floating-point Cholesky decomposition, the following main theorem can be proved about it.

Theorem 4.3 ([14, Theorem 2.3]) *For all* $n \in \mathbb{N}$ $(n \geq 1)$*, for all* $A, \tilde{R} \in \mathbb{F}^{n\times n}$*,* $m \in \mathbb{R}$*, if* $2(n+2)\texttt{eps} < 1$*,* $A^T = A$*, for all i,* $A_{i,i} \leq m$ *and* `cholesky_success` `A` $\tilde{\texttt{R}}$*, then*

$$\forall x \in \mathbb{R}^n, x \neq 0 \Rightarrow -|x|^T \Delta_A |x| < x^T A x$$

where $\Delta_{A\,i,j} := \alpha_{i,j} d_i d_j + 4\texttt{eta}(n+2+m)$ *with* $\alpha_{i,j} := \gamma_{\min(i,j)+2}$ *and* $d_i := \sqrt{\frac{A_{i,i}+2i\texttt{eta}}{1-\alpha_{i,i}}}$.

This theorem is proved thanks to the above Lemmas 4.1 and 4.2 and the lemmas about matrices of real numbers described in Section 4.1. It is pretty useless by itself but the following corollary looks closer from what we are looking for.

Corollary 4.4 ([14, Corollary 2.4]) *For all* $n \in \mathbb{N}$ $(n \geq 1)$*, for all* $A, \tilde{A} \in \mathbb{F}^{n\times n}$*,* $m, c \in \mathbb{R}$*, if* $2(n+2)\texttt{eps} < 1$*,* $A^T = A$ *and for all i,* $0 \leq A_{i,i} \leq m$ *and*

$$\forall x \in \mathbb{R}^n, \|x\|_2 = 1 \Rightarrow |x|^T \Delta_A |x| \leq c$$

and $\tilde{A}^T = \tilde{A}$ *and*

$$\begin{aligned}&\forall i, j \in [\![1,n]\!], i < j \Rightarrow \tilde{A}_{i,j} = A_{i,j}\\ &\forall i \in [\![1,n]\!], \tilde{A}_{i,i} \leq A_{i,i} - c\end{aligned}$$

then, if there exists $\tilde{R} \in \mathbb{F}^{n\times n}$ *such that* `cholesky_success` $\tilde{\texttt{A}}$ $\tilde{\texttt{R}}$*, we have*

$$A \succ 0.$$

Finally, it is proved that any value larger than

$$\frac{\gamma_{2n+2}}{2}\mathrm{tr}(A)+4\texttt{eta}(n+1)\left(2\left(n+2\right)+m\right)$$

will work as constant c in the above corollary as long as $4(n+2)\texttt{eps}<1$. Thus, an appropriate constant c can easily be computed, for instance with floating-point arithmetic with rounding toward $+\infty$. Then $\tilde{A}$ is computed by subtracting cId to A and the floating point Cholesky decomposition is performed. If it runs to completion without error, this rigorously proves that $A\succ 0$. This automatic positive definiteness check is efficient as it is performed with $\mathrm{O}(n^3)$ floating-point operations for a matrix A of size $n\times n$.

Thanks to the previous corollary, positive definiteness check can be performed on matrices A of floating-point values ($A\in\mathbb{F}^{n\times n}$). However, if the matrix $\tilde{X}$ we want to check has coefficients in the real field ($\tilde{X}\in\mathbb{R}^{n\times n}$), we first have to round them to floating-point values in $\mathbb{F}$ and we will end up checking some matrix $A\in\mathbb{F}^{n\times n}$ such that $\left|\tilde{X}-A\right|_{i,j}\leq R_{i,j}:=\texttt{eps}\left|A\right|_{i,j}+\texttt{eta}$. Such interval matrices are easily handled thanks to the following corollary.

Corollary 4.5 ([14, Corollary 2.7]) *For all $n\in\mathbb{N}$ ($n\geq 1$), for all $A,\tilde{A}\in\mathbb{F}^{n\times n}$, $R\in\mathbb{R}^{n\times n}$, $m,c,r\in\mathbb{R}$, if $2(n+2)\texttt{eps}<1$, $A^T=A$, $R\geq 0$ and for all i, $0\leq A_{i,i}\leq m$ and*

$$\forall x\in\mathbb{R}^n,\left\|x\right\|_2=1\Rightarrow\left|x\right|^T\Delta_A\left|x\right|\leq c$$

and

$$\forall x\in\mathbb{R}^n,\left\|x\right\|_2=1\Rightarrow\left|x\right|^T R\left|x\right|\leq r$$

and $\tilde{A}^T=\tilde{A}$ and

$$\forall i,j\in[\![1,n]\!],i<j\Rightarrow\tilde{A}_{i,j}=A_{i,j}$$
$$\forall i\in[\![1,n]\!],\tilde{A}_{i,i}\leq A_{i,i}-c-r$$

then, if there exists $\tilde{R}\in\mathbb{F}^{n\times n}$ such that `cholesky_success Ã R̃`*, we have*

$$\forall\tilde{X}\in\mathbb{R}^{n\times n},\tilde{X}^T=\tilde{X}\Rightarrow\left|\tilde{X}-A\right|\leq R\Rightarrow\tilde{X}\succ 0.$$

Since $n\max\left\{R_{i,j}\;\middle|\;i,j\in[\![1,n]\!]\right\}$ is a suitable value for r, this gives an effective criterion for positive definiteness of a matrix $\tilde{X}$ with coefficients in the real field $\mathbb{R}$.

The whole Coq development eventually counts 4.3 kloc. Among them, 0.4 are devoted to the specification of floating-point arithmetic (described in Section 2), 0.3 to bounded error terms (Section 3.1), 0.7 to the γ_k terms and their properties (Section 3.2), 0.4 to basic lemmas about sums and dotproducts (Sections 1.3 and 3.3) and 0.9 to matrices of real numbers (Section 4). Finally, the main theorem and corollaries (this section) take 1.3 kloc and the remainder (0.3 kloc) is constituted of miscellaneous lemmas. This appears particularly reasonable, considering the original result is a far from trivial 6 pages long paper proof [14].

```
x := 0;
while true do
   u := ?(−1, 1);      // random value in ℝ between −1 and 1
   x := Ax + Bu;
od
```

Fig. 2 A typical linear controller.

4.3 Another Application: Impact of Rounding Errors on Ellipsoidal Invariants

To assert the reusability of our developments for numerical analysis results involving matrices, we targeted another similar property. We will see that three quarters of the previous development can be directly reused.

Figure 2 displays the code of a typical linear controller. Quadratic Lyapunov functions [8,13] constitute a nice way to prove the stability of such controllers, i.e., that x remains bounded. An invariant ellipsoid $\mathcal{E} := \left\{ x \in \mathbb{R}^n \;\middle|\; x^T P x \leq \lambda \right\}$, for some $P \in \mathbb{R}^{n\times n}$ and $\lambda \in \mathbb{R}$, is then exhibited such that, for all $x \in \mathcal{E}$ and all $u \in \mathbb{R}^p$, if $\|u\|_\infty \leq 1$, then $Ax+Bu \in \mathcal{E}$. We thus get the property

$$\forall x \in \mathbb{R}^n, \forall u \in \mathbb{R}^p, x^T P x \leq \lambda \Rightarrow \|u\|_\infty \leq 1 \Rightarrow (Ax+Bu)^T P (Ax+Bu) \leq \lambda.$$

In practice, there will be some margin and we will get the property

$$\forall x \in \mathbb{R}^n, \forall u \in \mathbb{R}^p, x^T P x \leq \lambda \Rightarrow \|u\|_\infty \leq 1 \Rightarrow (Ax+Bu)^T P (Ax+Bu) \leq \lambda'. \quad (4)$$

for some $\lambda' \leq \lambda$. However, in an actual implementation, $Ax+Bu$ can be computed using floating-point arithmetic and the property we ultimately want to prove becomes

$$\forall x \in \mathbb{R}^n, \forall u \in \mathbb{R}^p, x^T P x \leq \lambda \Rightarrow \|u\|_\infty \leq 1 \Rightarrow \mathrm{fl}(Ax+Bu)^T P \mathrm{fl}(Ax+Bu) \leq \lambda. \quad (5)$$

The following theorem gives sufficient conditions for (4) to imply (5). In particular it characterizes how far λ' must be from λ.

Theorem 4.6 *For all $n, p \in \mathbb{N}$, $A, P \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times p}$, $s, s', \lambda, \lambda' \in \mathbb{R}$, if $2(n+p+1)\mathtt{eps} < 1$, $P^T = P$, $P \succ 0$, $sP - Id \succeq 0$, $s'Id - P \succeq 0$, then for all $x \in \mathbb{R}^n$, $u \in \mathbb{R}^p$ if*

$$x^T P x \leq \lambda,\ \|u\|_\infty \leq 1 \text{ and } (Ax+Bu)^T P (Ax+Bu) \leq \lambda'$$

then

$$\mathrm{fl}(Ax+Bu)^T P \mathrm{fl}(Ax+Bu) \leq \left(\sqrt{\lambda'} + \sqrt{\lambda} a + b\right)^2$$

where $a := \gamma_{n+p+1}\sqrt{ss'}\sqrt{n}\|A\|_F + 2\sqrt{ss'}n\sqrt{n}\mathtt{eta}$ and $b := \gamma_{n+p+1}\sqrt{s'}\sqrt{p}\|B\|_F + 2\sqrt{s'}(n+2p)\sqrt{n}\mathtt{eta}$ where $\|M\|_F$ denotes the Frobenius norm of the matrix M (i.e., $\|M\|_F := \sqrt{\sum_{i,j} M_{i,j}^2}$).

Among the 3.7 kloc needed to prove this theorem, 2.8 are shared with the development performed for the previous Section 4.2. Again, 0.9 kloc of Coq is a reasonably small amount of code fro translating such a non trivial 4 pages long paper proof.

5 Conclusion

We formally proved, using the proof assistant Coq [1,6], two results bounding rounding errors of numerical computations and involving matrices and common numerical analysis tools [11]. Our Coq development is available at `http://cavale.enseeiht.fr/formalbounds2014/`. It indicates that performing such proofs within proof assistants is tractable and that a large part of the proof effort could be reused for similar results. Our floating-point specification is based on the Flocq library of floating-point arithmetic for Coq [4]. It thus involves a constructive definition of floating-point arithmetic, without axioms.

The fact that we were able to translate, far from trivial, multiple pages paper proofs in about 1 kloc of Coq is a very encouraging achievement. It is also worth noting that performing mechanically checked proofs gave the opportunity to fix a few small mistakes in the proofs, thus asserting the interest of formalized proofs.

We eventually hope that a large part of our code can be reused in future developments, for instance about numerical integration of ODEs.

Acknowledgements The author wants to express its deepest thanks to Sylvie Boldo and Guillaume Melquiond as well as to Érik Martin-Dorel and Pierre-Marie Pédrot for their help regarding this work.

References

1. Yves Bertot, Pierre Castéran, Gérard Huet, and Christine Paulin-Mohring. *Interactive theorem proving and program development : Coq'Art : the calculus of inductive constructions*. Texts in theoretical computer science. Springer, Berlin, New York, 2004. Données complémentaires http://coq.inria.fr.
2. Yves Bertot, Georges Gonthier, Sidi Ould Biha, and Ioana Pasca. Canonical big operators. In Otmane Aït Mohamed, César A. Muñoz, and Sofiène Tahar, editors, *TPHOLs*, volume 5170 of *Lecture Notes in Computer Science*, pages 86–101. Springer, 2008.
3. Sylvie Boldo, François Clément, Jean-Christophe Filliâtre, Micaela Mayero, Guillaume Melquiond, and Pierre Weis. Formal proof of a wave equation resolution scheme: The method error. In Matt Kaufmann and Lawrence C. Paulson, editors, *Interactive Theorem Proving, First International Conference, ITP 2010, Edinburgh, UK, July 11-14, 2010. Proceedings*, volume 6172 of *Lecture Notes in Computer Science*, pages 147–162. Springer, 2010.
4. Sylvie Boldo and Guillaume Melquiond. Flocq: A Unified Library for Proving Floating-point Algorithms in Coq. In *Proceedings of the 20th IEEE Symposium on Computer Arithmetic*, pages 243–252, Tübingen, Germany, July 2011.
5. Cyril Cohen. Construction of real algebraic numbers in coq. In Lennart Beringer and Amy P. Felty, editors, *ITP*, volume 7406 of *Lecture Notes in Computer Science*, pages 67–82. Springer, 2012.
6. The Coq development team. *The Coq proof assistant reference manual*, 2012. Version 8.4.
7. Florent de Dinechin, Christoph Quirin Lauter, and Guillaume Melquiond. Assisted verification of elementary functions using gappa. In Hisham Haddad, editor, *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC), Dijon, France, April 23-27, 2006*, pages 1318–1322. ACM, 2006.
8. Éric Féron. From control systems to control software. *Control Systems, IEEE*, 30(6):50 –71, December 2010.
9. Georges Gonthier, Assia Mahboubi, and Enrico Tassi. A Small Scale Reflection Extension for the Coq system. Research Report RR-6455, INRIA, 2008.
10. John Harrison. Floating point verification in HOL. In E. Thomas Schubert, Phillip J. Windley, and Jim Alves-Foss, editors, *Higher Order Logic Theorem Proving and Its Applications, 8th International Workshop, Aspen Grove, UT, USA, September 11-14, 1995, Proceedings*, volume 971 of *Lecture Notes in Computer Science*, pages 186–199. Springer, 1995.
11. Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1996.

12. IEEE Computer Society. IEEE Standard for Floating-Point Arithmetic. *IEEE Standard 754-2008*, 2008.
13. Aleksandr Mikhailovich Lyapunov. Problème général de la stabilité du mouvement. *Annals of Mathematics Studies*, 17, 1947.
14. Siegfried M. Rump. Verification of positive definiteness. *BIT Numerical Mathematics*, 46:433–452, 2006.
15. Siegfried M. Rump. Verification methods: Rigorous results using floating-point arithmetic. *Acta Numerica*, 19:287–449, May 2010.