Pedagogical Uncertainty Propagation and Significant Figures in Lean 4

Alexander Meiburg

September 8, 2025

Abstract

We present a library for the Lean 4 theorem prover for numeric types with uncertainty propagation under exact, and crucially, inexact models, for precise modeling of semantics used in applied mathematical settings. The Lean 4 theorem prover provides a basis for verifiable computation, and is increasingly finding a role in the classroom as a companion to textbook and lecture material. It has some support for verifiable interval arithmetic, which has rigorous guarantees on error bounds. But scientific or educational contexts often define distinct semantics for uncertainties that treat errors less conservatively, such as the notion of "significant figure" tracking, and for Lean to be used pedagogically these semantics must be modeled. This library permits rigorous verification of textbook problems with unique, inexact answers.

1 Introduction

Uncertainty propagation is a fundamental aspect of scientific computation, and science in general: given input data with uncertainty, how do we define a meaningful result after applying mathematical operations? While informal rules for significant figures are taught in early science education, a formal, mechanized treatment is rare. Consider the following problem:

> A cylinder has height equal to its two times its diameter, and its volume is 4.86m^3. What is its height?

An exact answer would be $h = \sqrt[3]{\frac{16\times 4.86}{\pi}}\text{m}$, and this rounds at three decimal places to 2.91m. While that exact answer might be accepted by a teacher, the mathematically equal expression $6\sqrt[3]{\frac{9}{25\pi}}$ would likely not, and the answers 3m or 2.9143081m would often be considered incorrect for misrepresenting the accuracy. In math classrooms the topic of uncertainty is often secondary, but in science classrooms this is often the subject of several lessons; in science publications, an incorrect presentation of uncertainty verges on misconduct.

There is something implied by the expression $\sqrt[3]{\frac{16\times 4.86}{\pi}}\text{m}$ beyond its numerical value, that is clear. The Lean 4 theorem prover[7] and its companion library Mathlib[6] define rational and real numbers and permit the computer verification of proofs about these quantities. For instance, $2.91 < \sqrt[3]{\frac{16\times 4.86}{\pi}} < 2.92$ is a provable theorem in Lean – as are, for instance, the facts that its exact representation is a non-repeating decimal, and that its irrationality measure is at most fifteen. But that 2.91 would be the unique answer we expect this to be equal to, and that 2.910 represents something else, is an aspect of the *semantics* of the uncertainty that is as-of-yet not present.

We present a software package `SigFigs`, building on Lean and Mathlib, which defines these semantics. We build on Mathlib's existing interval arithmetic library, to give e.g. interpretation of

the number 4.86 as the interval $[4.855, 4.865]$, and similar syntax for easily applying mathematical operations to intervals. We then provide two other less rigorous models of uncertainty propagation, that are still popular used in the classroom setting: linear propagation of uncorrelated errors, and the popular "count the significant figures" method, which gives its name to the package.

2 Background

Formal verification is the process of using computer-assisted proof systems to rigorously check the correctness of mathematical statements, algorithms, and even entire software systems. The motivation for formal verification arises from the limitations of traditional mathematical practice: while human-written proofs are the foundation of mathematics, they are susceptible to errors, omissions, and ambiguities. As mathematics and computation have grown in complexity, so too has the need for tools that can provide higher degrees of certainty.

The Lean theorem prover[7], originally developed at Microsoft Research and now maintained by a vibrant open-source community, is a modern system designed to be both powerful and user-friendly. Lean's language is expressive enough to formalize advanced mathematics, while its automation and libraries make it practical for large-scale formalization. The Lean mathematical library, Mathlib[6], is one of the largest collections of formalized mathematics in existence, covering topics from basic algebra to advanced analysis and topology.

It is relatively recently that Lean has started to find room in the classroom.[1] Theorem provers, including Lean, have a famously steep learning curve compared to a typical programming language. Using Lean comfortably requires knowing not only the basics of *type theory*, but also functional programming, a deep hierarchy of mathematical definitions, and the conventions needed to navigate Mathlib's huge list of existing theorems. Lean is mostly used in courses whose objective is to teach computer-verified proof. But it has been tested in courses teaching basic mathematical logic, or even generic mathematical courses with a Lean companion such as topology[5, 4], cryptography[2], and real analysis[8]. The company Harmonic has announced[3] an educational app for mathematical fields, backed by Lean verification. Within this context, it seems necessary to equip Lean with the necessary tools for less precise fields of math.

Formal verification is often associated with stringent guarantees of correctness, and disastrous consequences in case of failure – a typical example is verifying that the software running a nuclear reactor cannot deadlock. In this sense, formal verification in a pedagogical setting may seem unnecessary. But formal verification has great pedagogical value: it forces us to make all assumptions explicit, to clarify definitions, and to confront edge cases that might otherwise be overlooked. In the context of uncertainty propagation, this means precisely specifying how uncertainty is represented, how it propagates through different types of operations, and what guarantees can be made about the results. The formalization described in this paper is an example of how modern proof assistants can be used to bring clarity and rigor to even the most practical side of scientific computation.

3 Pedagogical vs. Mathematical Models: Operational Semantics of Error Propagation

In scientific education, the rules for propagating uncertainty—such as those taught in grade school or introductory university courses—are often quite different from the mathematically exact models used in rigorous analysis. The pedagogical model is designed for clarity, intuition, and ease of use, rather than for capturing all possible sources of error or providing strict bounds. For example, the rules for significant figures are simple to state and apply, but they are not mathematically

robust: addition and multiplication are not associative, and the results are often only heuristically justified. In contrast, mathematicians and numerical analysts are concerned with precise definitions, rigorous error bounds, and the preservation of mathematical properties such as associativity and distributivity.

This distinction highlights the importance of *operational semantics* in formalization. By operational semantics, we mean a model of notation and numbers that reflects the intended meaning of the problem as it is presented to students or practitioners. In other words, the semantics of a calculation are determined not just by the underlying mathematics, but by the conventions and rules that govern how numbers and uncertainty are represented and manipulated in practice. Classroom science often prioritizes rules that are easy to remember and apply, even if they are not universally valid, while mathematicians seek models that are internally consistent and generalizable.

Consider the following example from the `IntervalExamples.lean` file:

```
#check (1.23 : ℝRange) --when type-ascripted, the decimal is interpreted as an interval [1.225
#check 3.21 --note that this parses as a `Float` on its own!
```

In a classroom setting, a number like 1.23 is often understood to mean "1.23 with uncertainty in the last digit," i.e., 1.23 ± 0.005. The operational semantics in Lean can be set up to reflect this, so that the type of the number determines how it is interpreted. In contrast, a mathematician might insist on specifying the interval explicitly, or distinguishing between exact and approximate values.

Another example is the addition of uncertain quantities. In the pedagogical model, the rule is often "round the result to the least precise decimal place," as in:

```
example (A B : ℝRange) (hA : A = 100 ± 3) (hB : B = 30 ± 15) :
  A + B = 130 ± 18 := by
  sorry
```

Here, the uncertainties are simply added, and the result is rounded to match the least precise input. This is easy to teach and apply, but does not always reflect the true propagation of error. In a more exact mathematical model, one might add the uncertainties in quadrature (i.e., take the square root of the sum of the squares), or use interval arithmetic to compute the full range of possible values.

Nonlinear functions provide another point of divergence. In the `FOBallExamples.lean` file, uncertainty is propagated through functions using a first-order (differential) approximation:

```
#check (letI x : FOBall := Real.pi; ↑Real.sin (7 * x + 6.7))
```

This approach, common in classroom science, uses the derivative to estimate how uncertainty in the input affects the output. It is fast and intuitive, but only accurate for small uncertainties and well-behaved functions. The interval model, by contrast, would compute the exact image of the interval under the function, which is mathematically rigorous but often impractical for hand calculation or teaching.

The ability to reflect these different operational semantics in Lean is crucial for both pedagogy and research. It allows us to formalize textbook problems in a way that matches their intended meaning, to check the correctness of solutions under the rules actually taught, and to compare the outcomes of different models. For example, a teacher can encode a problem using the significant figures model to match classroom expectations, while a mathematician can use interval arithmetic for rigorous bounds. By providing multiple models of uncertainty—each with its own operational semantics—our framework enables users to choose the approach that best fits their needs, whether for rigorous proof, practical computation, or educational clarity.

4 Models of Uncertainty

We provide three models, each with different trade-offs between rigor, usability, and pedagogical value.

4.1 Interval Arithmetic ℝRange

In interval arithmetic, each value is represented as an interval $[a, b]$ containing all possible values. All operations are defined to ensure the result interval contains the true value. This model is mathematically rigorous and suitable for downstream theorem proving, but can be overly conservative in practice.

```
#check (1.23 : ℝRange) -- [1.225, 1.235]
#check 7 ± 1           -- [6, 8]
```

As an example of applicability to a physics setting:

```
example (g t height : ℝRange) (hg : g = 9.8) (ht : t = 5.18) (h_height : height = 150.0) :
    height - (1/2) * g * t^2 ≈ 19. := by
  sorry
```

By first declaring the variables g, t, and $height$ as ℝRanges the subsequent hypotheses automatically interpret the numerical literals as intervals. For instance, g is interpreted as $[9.75, 9.85]$. This inaccuracy propagates through the subsequent calculations. The value $\frac{1}{2}$ is interpreted precisely as a rational number, in contrast to 0.5 which would mean $[0.495, 0.505]$.

The final expression $height - \frac{1}{2} * g * t^2$ evaluates to a particular interval, $[17.545191875, 19.494453125]$, or equivalently $18.5198225 \pm 0.974630625$. A true theorem would then be to equate these intervals:

```
example (g t height : ℝRange) (hg : g = 9.8) (ht : t = 5.18) (h_height : height = 150.0) :
    height - (1/2) * g * t^2 = 18.5198225 ± 0.974630625 := by
  sorry
```

while equating with 19 or 19.0 would be incorrect. Due to Lean's stringent notion of equality, the two sides must be exactly equal, including their uncertainty, which does not reasonably reflect the intent of a textbook problem. This highlights the trade-off between rigor and pedagogical value in this model. But, as two ways to reasonably express the relationship of an answer, we do offer two other notions of a "correct answer" that are available for stating a problem. One, as seen above, is an approximate equality, $\approx 19.$. This means is defined to mean that the interval $19. := [18.5, 19.5]$ overlaps with the interval on the other side, and that the associated uncertainties to these intervals are within an order of magnitude difference. This forces there to be a unique correct scientific notation on the right-hand side that satisfies the $\approx$ relation.

The other, much weaker notion is simply expressing membership in the interval:

```
example (g t height : ℝRange) (hg : g = 9.8) (ht : t = 5.18) (h_height : height = 150.0) :
    : 19 ∈ height - (1/2) * g * t^2 := by
  sorry
```

This merely means that the (exact real number) 19 is consistent with the uncertainties. But 19.192082, as well as $16 + \pi$, would be satisfy this membership relation as well.

4.2 First-Order Ball Arithmetic (`FOBall`)

First-order ball arithmetic represents a value as a center and a radius, interpreted as a sensitivity or standard deviation. Errors are added in quadrature, and uncertainties are propagated using derivatives (the delta method).

```
#check 1 ± 157       -- 1 with uncertainty 157
#check (1.5 : FOBall) -- 1.5 ± 0.05
#check (1.50 : FOBall) -- 1.5 ± 0.005
```

4.3 Significant Figures (`SigFigs`)

The significant figures model represents a value as a center and a number of significant digits. This is the most intuitive for students, but is the least mathematically robust: addition and multiplication are non-associative, and all bounds are heuristic.

5 Examples

5.1 Physics Problem: Free Fall

Problem: You drop a ball from a height of 150.0 m. After 5.18 seconds, how far is it above the ground? Use $h = h_0 - \frac{1}{2}gt^2$.

```
example (g t height : SigFig) (hg : g = 9.8) (ht : t = 5.18) (h_height : height = 150.0) :
    height - (1/2) * g * t^2 = 19. := by
  sorry
```

Here, the operator $\approx$ indicates that the output is an approximation to the correct accuracy, and the types ensure that uncertainty is propagated.

5.2 Adding Uncertain Quantities

Problem: Alice has a box of chocolates that weighs 100 ± 3 g, and Bob has a box that weighs 30 ± 15 g. How much do they have together?

```
example (A B : ℝRange) (hA : A = 100 ± 3) (hB : B = 30 ± 15) :
  A + B = 130 ± 18 := by
  sorry
```

6 Implementation in Lean 4

The package overloads arithmetic operations for each uncertainty model, so that textbook-style problems can be written naturally in Lean. Numeric literals are interpreted with the correct uncertainty, and notations such as $\pm$ are available.

7 Conclusion and Future Work

This formalization enables both rigorous and pedagogically useful reasoning about uncertainty in Lean. Future work includes:

- Extending support for all field operations and nonlinear functions.
- Theorems for converting between models.
- Integration with computable real and interval packages for more automation.

Acknowledgments

We thank the Lean and Mathlib communities for their support.

References

[1] Courses using Lean. `https://leanprover-community.github.io/teaching/courses.html`. [Accessed 31-08-2025].

[2] Matthew Ballard. Intro to Cryptography — 587.f24.matthewrobertballard.com. `https://587.f24.matthewrobertballard.com/`. [Accessed 31-08-2025].

[3] Harmonic Fun. Aristotle achieves gold medal-level performance at the international mathematical olympiad, ios app beta launch. `https://harmonic.fun/news`, 2025. [Accessed 31-08-2025].

[4] Miguel Marco. Consulta de Guías Docentes. `https://sia.unizar.es/doa/consultaPublica/look[conpub]MostrarPubGuiaDocAs?entradaPublica=true&idiomaPais=es.ES&_anoAcademico=2023&_codAsignatura=27008`. [Accessed 31-08-2025].

[5] Miguel Marco. GitHub - miguelmarco/topologia_general_lean: Material auxiliar en Lean3 Para un curso de topología general. `https://github.com/miguelmarco/topologia_general_lean`. [Accessed 31-08-2025].

[6] The mathlib Community. The lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs*, CPP 2020, page 367–381, New York, NY, USA, 2020. Association for Computing Machinery.

[7] Leonardo de Moura and Sebastian Ullrich. The lean 4 theorem prover and programming language. In André Platzer and Geoff Sutcliffe, editors, *Automated Deduction – CADE 28*, pages 625–635, Cham, 2021. Springer International Publishing.

[8] Terence Tao. GitHub - teorth/analysis: A Lean companion to Analysis I. `https://github.com/teorth/analysis`, 2025. [Accessed 31-08-2025].