The Staircase of Signs

Sturm's root-counting theorem, machine-checked in Lean 4

Carles Marín*
Independent researcher
karlesmarin@gmail.com

June 2026

Abstract

I report a complete, `sorry`-free formalization in Lean 4, over Mathlib, of *Sturm's theorem* (1829): for a squarefree real polynomial p and an interval $(a, b]$ whose endpoints are not roots, the number of distinct real roots of p in $(a, b]$ equals

$$\mathrm{V}(a) - \mathrm{V}(b),$$

where $\mathrm{V}(x)$ is the number of sign changes in the *Sturm sequence* p, p', $-(p \bmod p')$, ... evaluated at x (zeros discarded). No root is ever located; two integers are subtracted. The mathematics is entirely classical and the result has been formalized before in other systems (Coq, Isabelle/HOL, HOL Light); to the best of my knowledge— based on searches of Loogle and Mathlib in June 2026 (Section 8)—this is the first proof of Sturm's theorem in Lean. The contribution is therefore the formalization itself and the reusable machinery it required: a small theory of sign variation, an inductive "flank-reduction" relation that decouples the chain's combinatorics from its algebra, and the local-to-global passage from a single root crossing to the interval count. The headline theorem `Sturm.sturm` depends only on the three standard axioms `propext`, `Classical.choice`, `Quot.sound`. A by-product is that Mathlib's existing count of coefficient sign variations (Descartes' rule) and the count used here are, after unfolding, *the same function*—so the toolkit transfers verbatim to Descartes.

What you will find here. This is a short guided tour, not a reference manual. It is organized as four questions, each opening a section and handing the reader the next. The first asks how two integers can possibly count roots (Section 2); the second, why the count moves only at the roots, and by exactly one (Section 3); the third, where the proof genuinely resists being made formal, and what that resistance taught me (Section 4); the fourth, what becomes free once the theorem is in the library (Section 5). The load- bearing lemmas are gathered in one table (Section 7, Table 2); the honest limits—squarefreeness, the ground field, what was *not* proved, and the prior formalizations in other systems—are stated without varnish in Section 8; and the closing section asks, by way of reflection, what this paper still lacks. If you read only two things, read Theorem 1 and Figure 2; the rest is the argument that connects them.

*Formalized with the assistance of an AI system (Claude, Anthropic); the mathematics is Sturm's, and every claim, scope statement, and limitation is the author's responsibility. The boundary of the contribution is set out plainly in Section 8, and the Lean kernel—not the assistant—is what certifies the proofs.

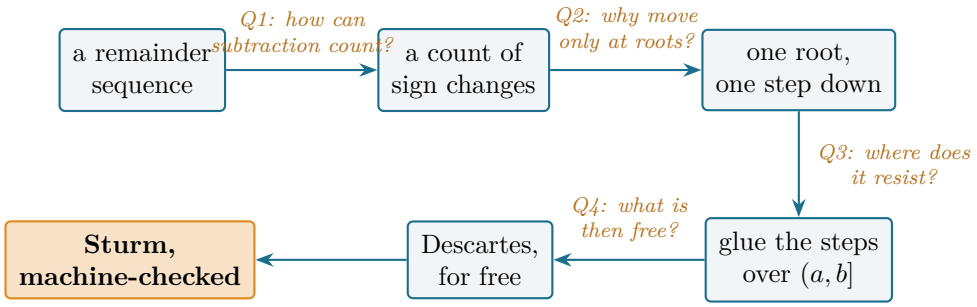


Figure 1: The reader's map. From a sequence of remainders to a certified root count and what it yields downstream; each arrow is the question its section answers. Every node is a block of `sorry`-free Lean.

1 The question that begins it

How many real roots does a polynomial have between here and there—and can you say so without finding a single one of them?

The question is old and was, for a long time, genuinely hard. Descartes' rule of signs (1637) [1] reads a *bound* off the signs of the coefficients; Budan and Fourier (early nineteenth century) [2] sharpened it to a bound for an interval; but a bound is not a count, and the gap between them resisted the best analysts of the age. It was closed in 1829 by a young Charles-François Sturm, who announced—and in 1835 published in full [3]—a procedure that returns the *exact* number of distinct real roots in any interval, with no approximation and no root-finding whatsoever. The method was celebrated at once; it is the ancestor of real-root isolation, of cylindrical algebraic decomposition, and of the decision procedures for the first-order theory of the reals that grew out of Tarski's work. The problem of counting real roots exactly was, in effect, settled by a subtraction.

The mechanism is a small twist on something even older. Euclid's algorithm, run on p and its derivative p', produces a sequence of remainders; Sturm's twist is to negate each remainder as it is formed,

$$p_0 = p, \quad p_1 = p', \quad p_{k+1} = -(p_{k-1} \bmod p_k),$$

stopping when the remainder vanishes. Call this the *Sturm sequence*. For a point x that is not a root of any member, let $\mathrm{V}(x)$ be the number of times the sign changes as one reads $p_0(x), p_1(x), \ldots$ from left to right, ignoring any zeros. Sturm's theorem is the statement that V is, in disguise, a root counter:

$$\#\{\text{distinct real roots of } p \text{ in } (a, b]\} \;=\; \mathrm{V}(a) - \mathrm{V}(b).$$

The result is textbook, and has been machine-checked before: in Coq, as part of Cohen's construction of the real algebraic numbers [5]; in Isabelle/HOL, by Eberl [6] and—in the sharper Sturm–Tarski form, via the Cauchy index—by Li and Paulson [7, 8]; and in HOL Light. What it had not been, until now, is checked in *Lean*. Mathlib, the library on which a large part of contemporary formalized mathematics is built, carries Descartes' rule of signs (the function `Polynomial.signVariations`) but not Sturm's sequence and not Sturm's theorem. This paper fills that gap. It is, I want to be exact, a first-in-Lean and not a first-in-any-system;[1] the value is in

[1] Priority claim based on searches of Loogle and a `grep` of Mathlib as of June 2026; the prior formalizations in Coq, Isabelle/HOL and HOL Light are credited in Section 8.

adding a foundational tool to a library that lacked it, and in the reusable pieces the proof leaves behind.

I came to it not by setting out to prove Sturm's theorem but by asking, repeatedly, which classical results a mature library is still missing—and Sturm's, sitting squarely in real-root counting, was among the most conspicuous absences. The rest of the paper is the road from the absence to the theorem, told as the four questions a careful reader is likely to ask in turn.

2 Q1: how can subtracting two integers count roots?

History bead. The sequence and the counting statement are Sturm's own ([3]; the modern textbook account is Basu–Pollack–Roy [4]). Nothing in this section is new mathematics; the Lean is in `Sturm.lean`, definitions `sturmSeq` and `signVarAt`.

Let us make the two ingredients precise. The Sturm sequence is built by the negated Euclidean recursion above; in Lean it is `sturmSeq p`, a `List` of polynomials produced by an auxiliary `sturmAux` that recurses on the divisor's degree and terminates because each remainder has strictly smaller degree. The sign count is

$$\mathrm{V}(x) \;=\; \texttt{signVarAt (sturmSeq p) x},$$

defined as the number of sign changes in the list of evaluated signs $(\operatorname{sign} p_0(x), \operatorname{sign} p_1(x), \dots)$ after deleting the zeros. (In the Lean source the deletion-and-count is phrased with `List.destutter`; an equivalent and more convenient recursive form, `signChanges`, is proved equal to it and is what the rest of the development uses.)

Theorem 1 (Sturm's theorem, in Lean 4; `Sturm.sturm`)**.** *Let p be a squarefree real polynomial and $a < b$ with $p(a) \neq 0$ and $p(b) \neq 0$. Then*

$$\mathrm{V}(a) - \mathrm{V}(b) \;=\; \#\{x : a < x \leq b,\ p(x) = 0\},$$

the number of distinct *real roots of p in the half-open interval $(a, b]$. The statement is `sorry`-free; `#print axioms Sturm.sturm` reports only `propext`, `Classical.choice`, `Quot.sound`.*

That a difference of two integers should equal a count is, on first sight, a small miracle, so it is worth seeing it happen once by hand.

A hand-checkable case. Take $p(x) = x^3 - x = x(x-1)(x+1)$, with roots $-1, 0, 1$. The negated Euclidean recursion gives the four-term sequence

$$p_0 = x^3 - x, \quad p_1 = 3x^2 - 1, \quad p_2 = \tfrac{2}{3}x, \quad p_3 = 1.$$

Read the signs at the two endpoints of $(-2, 2]$:

x	p_0	p_1	p_2	p_3	V
-2	$-$	$+$	$-$	$+$	3
2	$+$	$+$	$+$	$+$	0

so $\mathrm{V}(-2) - \mathrm{V}(2) = 3 - 0 = 3$, exactly the number of roots in $(-2, 2]$. Notice that no root was computed; only signs were read. (Every entry of this table is re-checked in exact arithmetic by the script accompanying Figure 2.)

The picture behind the table is the one that names this paper. Plot $\mathrm{V}(x)$ as x sweeps the line: it is a *staircase*, flat almost everywhere and dropping by one as x passes each root of p (Figure 2;

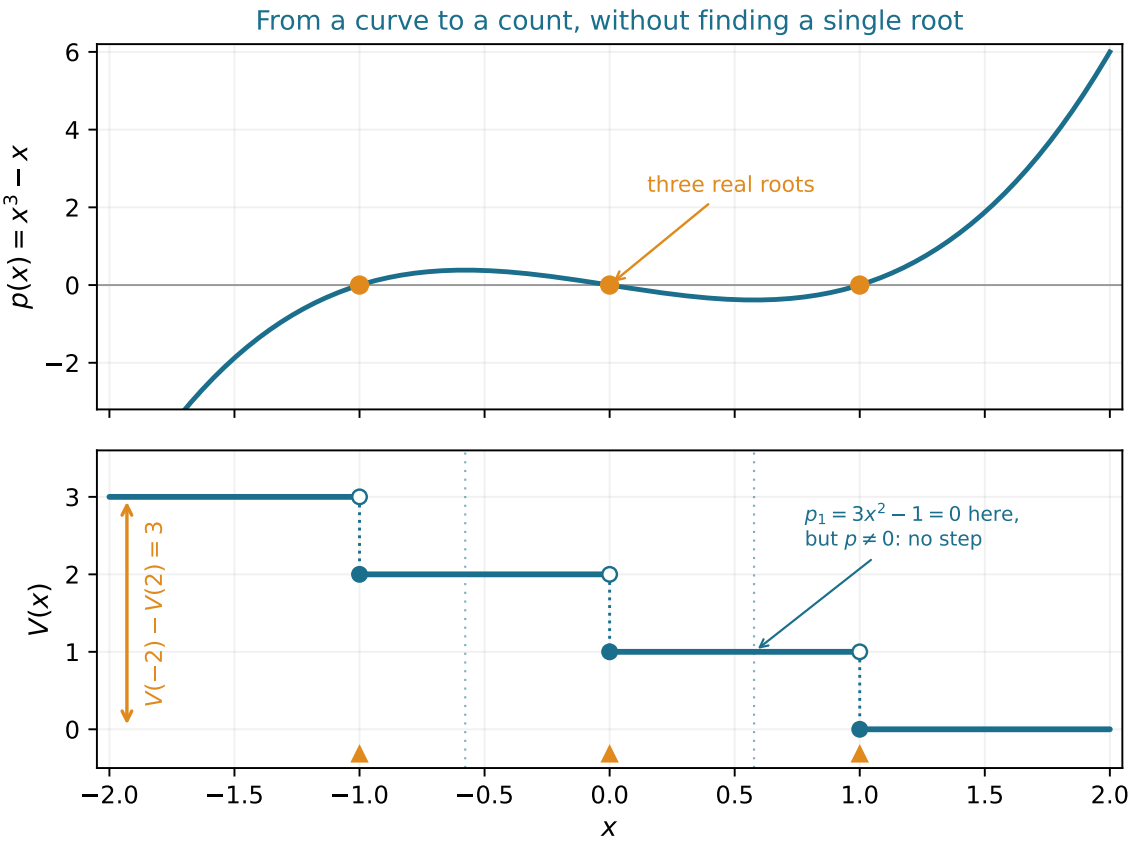


Figure 2: The staircase of signs for $p = x^3 - x$. *Top:* the polynomial and its three real roots. *Bottom:* the count $\mathrm{V}(x)$ is flat except at the roots $-1, 0, 1$, where it drops by exactly one (open circle = the value just to the left, filled circle = the value at the root, on the lower step). At a zero of an *interior* member of the sequence—here $p_1 = 3x^2 - 1$ vanishes at $x = \pm 1/\sqrt{3}$ (dotted guides), points where p itself is nonzero—the staircase does *not* move. The total descent over $(-2, 2]$ (the amber arrow) is the number of roots there. The generating script asserts every plotted sign and count against exact rational arithmetic before drawing.

Table 1 reads it off point by point). Sturm's theorem is then nothing but reading the total descent of the staircase between a and b. The entire difficulty—and the entire content of the next two sections—is in proving that the staircase has exactly this shape: that it is flat except at roots, and that each step has height exactly one.

x	$p_0 = x^3 - x$	$p_1 = 3x^2 - 1$	$p_2 = \frac{2}{3}x$	$p_3 = 1$	$\mathrm{V}(x)$
-2	$-$	$+$	$-$	$+$	3
-1 (root)	0	$+$	$-$	$+$	2
$-\frac{1}{2}$	$+$	$-$	$-$	$+$	2
$-\frac{1}{\sqrt{3}}$ (p_1=0)	$+$	0	$-$	$+$	2
0 (root)	0	$-$	0	$+$	1
$\frac{1}{\sqrt{3}}$ (p_1=0)	$-$	0	$+$	$+$	1
1 (root)	0	$+$	$+$	$+$	0
2	$+$	$+$	$+$	$+$	0

Table 1: The same example read off the signs. Each row evaluates the four chain members and counts the sign changes (zeros skipped). V falls by one at each *root* of p and is unchanged across each zero of the *interior* member p_1 (the $\frac{1}{\sqrt{3}}$ rows)—the two behaviours Section 3 explains. The whole table is regenerated and checked in exact arithmetic by the figure script.

3 Q2: why does the count move only at the roots, and by exactly one?

History bead. The local analysis—what happens to V as x crosses a point—is the heart of every proof of Sturm's theorem since Sturm's. The two facts it turns on (a member and its neighbours cannot vanish together; consecutive members are antipodal at a zero of the one between them) are classical; the Lean is in the lemmas `not_common_root`, `sign_neighbours_opposite_at_interior_root`, and `signVarAt_drop_at_critical_point`.

Call a point *critical* if some member of the sequence vanishes there. Away from critical points every member keeps its sign (a polynomial cannot change sign without a root, by the intermediate value theorem), so V is locally constant: this is `signVarAt_eq_of_no_root`, and it is why the staircase is flat between criticals. All the action is at the critical points, and there are two kinds.

The roots of p. Suppose $p(c) = 0$. Because p is squarefree it shares no root with p', so $p'(c) \neq 0$, and just to the left and right of c the polynomial p takes opposite signs (it crosses zero transversally, with the sign of $p'(c)$ on the right and the opposite on the left). The very first pair of the sequence, (p, p'), therefore changes its number of internal sign agreements by exactly one across c. This is the single step of the staircase, isolated in Lean as the leading-pair drop `signVarAt_cons_head_drop`, fed by the transversality lemmas `sign_eval_eq_sign_deriv_right` and `sign_eval_eq_neg_sign_deriv_left`.

The other critical points. A member p_k with $1 \leq k$ may also vanish at c without p doing so. Here is the mechanism that makes Sturm's theorem work, and it is worth stating as the one indispensable lemma:

Lemma 1 (antipodal neighbours; `sign_neighbours_opposite_at_interior_root`)**.** *If consecutive members p_{k-1}, p_k are coprime and $p_k(c) = 0$, then $p_{k-1}(c)$ and $p_{k+1}(c)$ are both nonzero and have* opposite *signs.*

The reason is a one-line consequence of the defining recursion $p_{k+1} = -(p_{k-1} \bmod p_k)$ evaluated at a root of p_k: there $p_{k-1}(c) = -p_{k+1}(c)$ (lemma `eval_eq_neg_next_of_root`), and neither can be zero because coprime polynomials share no root (`not_common_root`). Consequently, whatever sign p_k adopts on either side of c—and whether it is positive, negative, or exactly zero *at* c—it sits between two fixed, opposite neighbours, and a value squeezed between opposite signs contributes nothing to the count of sign changes, regardless of what it is. The middle member is invisible.

This is why a zero of an interior member moves the staircase not at all (the $p_1 = 0$ marks in Figure 2); coprimality of consecutive members, which propagates along the chain from $\gcd(p, p') = 1$, guarantees it everywhere.

Putting the two kinds together gives the per-point quantum—the height of a single step:

Proposition 1 (single critical point; `signVarAt_drop_at_critical_point`)**.** *If c is the only critical point in $[a, b]$ and $p(a), p(b) \neq 0$, then*

$$\mathrm{V}(a) = \mathrm{V}(b) + \begin{cases} 1 & p(c) = 0, \\ 0 & p(c) \neq 0. \end{cases}$$

The proof is where the formalization earns its keep, and it is the subject of the next section, because making it rigorous required a device the textbook does not need.

4 Q3: where does the proof resist being made formal?

History bead. On paper, "the interior members cancel and only p matters" is a sentence. Made formal, it is the one genuinely structural step, because the cancellation has to be performed *along a list whose shape is itself recursive*. The device that resolves it (`FlankReduce` and `flankReduce_chain_walk`) is, as far as I know, the only part of this development without a direct textbook antecedent—not a new theorem, but a piece of proof engineering.

The difficulty is precise. To compare $\mathrm{V}(a)$ with $\mathrm{V}(b)$ across a single critical point c, one wants to delete from the sign list, at *both* a and b, every interior member that vanishes at c—each is squeezed between opposite neighbours, so deleting it changes nothing (Lemma 1)—and then observe that the two pruned lists differ only in the head p. But "delete each such member" is an induction over the Sturm sequence, and the hypotheses that justify a deletion (the flanking neighbours are nonzero and opposite *at the evaluation point*) are threaded through the chain's own structure. The combinatorics of pruning a list and the algebra of why each pruning is legal are entangled, and an honest proof must separate them.

The separation is an inductive relation. Say a list of signs *flank-reduces* to another when the second is obtained from the first by repeatedly deleting an entry that lies between two nonzero opposite signs:

The decoupling (recipe)

`FlankReduce` $L\ L'$ holds when L' is obtained from L by deleting entries each caught between two nonzero opposite signs: one step takes $\mathit{pre} +\!\!+ [a, m, b] +\!\!+ \mathit{rest}$ to $\mathit{pre} +\!\!+ [a, b] +\!\!+ \mathit{rest}$ whenever $a, b \neq 0$ and $a \neq b$, and the relation is the reflexive closure of iterating it. Two facts then divide the labour cleanly. *Combinatorial:* flank-reduction preserves the number of sign changes (`FlankReduce.signChanges_eq`)—this is pure list algebra, no polynomials. *Structural:* the Sturm chain's evaluated sign list at any point flank-reduces to the chain with its interior c-vanishers removed (`flankReduce_chain_walk`)—this is the induction over the chain, supplying the flank hypotheses from coprimality and from local sign constancy, and nothing else.

With the two halves separated, Proposition 1 becomes assembly: the chain's sign list flank-reduces, at a and at b alike, to the *same* pruned chain p`:` (non-vanishers); on that pruned chain only p can change sign between a and b, so the change in V is the single head-drop when $p(c) = 0$ and

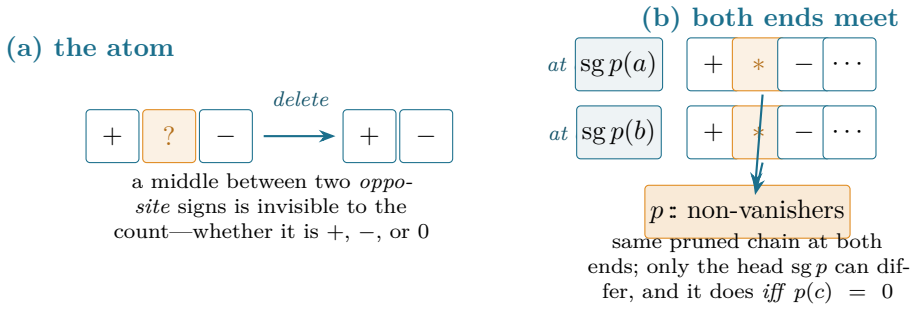


Figure 3: The decoupling of Section 4. *(a)* The combinatorial atom: an entry flanked by opposite nonzero signs ($*$, the interior vanisher) can be deleted without changing the number of sign changes, whatever its own value. *(b)* Applied along the chain at both endpoints, every interior c-vanisher is deleted, so the sign lists at a and at b reduce to the *same* pruned chain; the only entry that can differ is the head $\operatorname{sg} p$. The chain's algebra supplies the "opposite neighbours" hypotheses; the count is moved by `FlankReduce`, which never mentions a polynomial.

nothing otherwise. The chain structure enters only to hand the combinatorial lemma its hypotheses; the count is moved by a relation that knows nothing of polynomials.

What the machine made me state correctly. There is a quieter lesson here, of the kind a proof assistant is good at extracting. The classical proof tacitly assumes "generic position": that no member of the sequence vanishes at the endpoints a, b, and that one may always slide to such points. Formalizing forbids the hand-wave—and it turns out the flank-reduction already handles the awkward cases for free. A member vanishing exactly at an evaluation point is still squeezed between opposite neighbours there, so it is still deleted by the same relation; the pruned chain is the same; the count is unaffected. I therefore proved the per-point lemma (Proposition 1) for a critical point anywhere in the closed interval $[a, b]$, endpoints included, rather than only strictly inside—and the degenerate cases the textbook waves past are discharged by the same line as the main one. The transferable principle is the same one the companion paper on Newton's inequalities [11] drew: phrasing a step at the weakest honest level of generality often dissolves, rather than multiplies, the edge cases.

From one step to the whole staircase. The global statement is then an induction on the number of critical points in $(a, b]$ (`Sturm.sturm`; the criticals are the roots of the product of all members, a nonzero polynomial, hence finite). Peel off the largest critical point c; choose a non-critical separator d just below it—one exists because the criticals are finite and the reals are dense; apply the per-point quantum on $[d, b]$, where c is the only critical point; and recurse on $[a, d]$, which has one fewer. The half-open interval $(a, b]$ splits as $(a, d] \sqcup (d, b]$, the root count adds, and the per-point contribution—one when $p(c) = 0$, zero otherwise—matches the number of p-roots in the peeled piece. The global count is assembled from the per-point steps.

Remark 1 (the pattern behind `FlankReduce`)**.** *`FlankReduce` is worth abstracting from its use here. It is an instance of a recurring schema: a numerical invariant—the sign-change count—preserved under deletions that are licensed by a purely* local *side-condition (an entry between two opposite nonzero signs). One might call this a* count invariant under admissible pruning*. Its value is precisely the decoupling exhibited above: the global combinatorics reduces to (i) a one-step preservation lemma, polynomial-free and proved once (`FlankReduce.signChanges_eq`), and (ii) a structural walk that certifies each deletion's side-condition from the problem at hand (`flankReduce_chain_walk`). The same separation should apply wherever a tally is invariant under locally-justified removals—the*

derivative sequence of Budan–Fourier, or more generally any rewriting system with a conserved count. I prove only the instance Sturm needed and do not develop the general theory; but the shape is general, and naming it—rather than inlining the deletions into the chain induction—is what made the proof modular. Whether "invariance of a tally under locally-licensed deletion" deserves to be a Mathlib abstraction in its own right is the open question I find most worth carrying forward (Section 9).

5 Q4: what becomes free once the theorem is certified?

History bead. Sturm's count of *interval* sign variations and Descartes' count of *coefficient* sign variations (1637) are usually taught as cousins. In Lean they turn out to be, after unfolding, the very same function—a small surprise the formalization surfaced for free. The Lean is `signVariations_eq_signChanges`.

Mathlib already had Descartes' rule of signs, phrased through `Polynomial.signVariations`, the number of sign changes in a polynomial's coefficient list. The count built here for Sturm's theorem, `signChanges`, is defined on an arbitrary list of signs. Unfolding both, they coincide on the nose:

```
P.signVariations = signChanges (P.coeffList.map sign),
```

a proof that is literally `rfl`. The consequence is practical: the toolkit assembled for Sturm—zeros are invisible to the count, an entry between opposite neighbours cancels, the head recursion, the parity (a zero-free list has an even number of sign changes iff its ends agree)—is not Sturm-specific. It transfers verbatim to Descartes, and the two classical sign-variation theorems now share a single combinatorial core in the library rather than two parallel ones. A small unification, but exactly the kind a formal library should accumulate.

Two further things fall out without new work. First, *root isolation*: applying the theorem to $(a, b]$ and to its two halves repeatedly isolates each real root in an interval as small as one likes, the algorithmic use Sturm's method is best known for. Second, the *building blocks* themselves—local sign constancy, the antipodal-neighbour lemma, the structural facts about the sequence (consecutive members coprime, the recursion relating each triple, the final member a nonzero constant)—are reusable for any further work on real-root counting in Lean: the Budan–Fourier theorem, the Sturm–Tarski count with multiplicity [8], or a Cauchy-index development would draw on the same pieces.

6 What a certified root count enables

The mathematics here is two centuries old; the reason to put it in Lean is what it unlocks *inside* the library, where a verified, exact root count had simply not existed. It is worth being concrete about that, because a foundational lemma is valuable in proportion to what rests on it. Four things become reachable that were not before, in rough order of distance from what is proved.

Real-root isolation. The classical algorithmic use of Sturm's theorem is to isolate each real root in an interval as narrow as desired: apply the count to $(a, b]$, bisect, recurse, and stop a branch when its count is 0 (no root) or 1 (exactly one). Because the count here is *exact*, not a bound, the recursion provably terminates with one root per interval. Every ingredient is already proved—the count, local constancy, the finiteness of the critical set—so what remains is the recursive wrapper

and its termination, not new mathematics. This turns the theorem from a statement into a certified *procedure*, the form in which Sturm's method is actually used.

The Budan–Fourier theorem. Budan and Fourier count sign variations in the sequence of *derivatives* $p, p', p'', \ldots$ and bound (with the right parity) the roots in an interval. That is the same sign-variation arithmetic as here, on a different sequence; the toolkit built for Sturm—zeros invisible, the head recursion, and in particular the parity lemma `wallCount_even_iff_endpoints`—transfers directly. Mathlib already had Descartes' rule (`signVariations`); Budan–Fourier sits between Descartes and Sturm, and the pieces to reach it are now in place.

Sturm–Tarski and the Cauchy index. The sharper Sturm–Tarski theorem counts roots of p *weighted* by the sign of a second polynomial q on them—the "Tarski query" $\sum_{p(x)=0} \operatorname{sign} q(x)$—which equals the Cauchy index of $p'q/p$ (the basic case $q = 1$, the Cauchy index of p'/p, being Sturm's own count). This is the engine of root counting *with multiplicity*, of the non-squarefree case, and ultimately of Tarski's quantifier elimination for the reals. The signed-remainder sequence and the sign-variation count assembled here are precisely its substrate; the Isabelle development of Sturm–Tarski [7, 8] is built on the same two objects. A Lean version would extend, not restart, the present file.

Decision procedures over the reals. One-dimensional real-root counting and isolation are the base case of cylindrical algebraic decomposition and of the decision procedures for the first-order theory of real-closed fields (Tarski–Seidenberg). For Lean specifically this is the missing foundation under any future *certified* answer to "does this polynomial have a root in this range, and how many"—a checkable certificate a downstream tactic could trust, rather than an oracle it must believe. None of these four is done here; the point is that each was blocked, in Lean, on exactly the result this paper supplies.

7 The building blocks

The theorem rests on a handful of reusable lemmas, each `sorry`-free with the standard three axioms (Table 2). Three are worth naming for their portability beyond Sturm.

Lemma 2 (local sign constancy; `signVarAt_eq_of_no_root`)**.** *If no member of the sequence has a root in* $[a, b]$*, then* $\mathrm{V}(a) = \mathrm{V}(b)$*.*

The intermediate value theorem, lifted from one polynomial to the list: this is the flatness of the staircase between critical points, and the engine of the global induction's base case.

Lemma 3 (flank-reduction preserves the count; `FlankReduce.signChanges_eq`)**.** *If a sign list flank-reduces to another, the two have the same number of sign changes.*

The combinatorial half of Section 4, depending on no polynomial at all—the cleanest statement of "entries between opposite neighbours do not matter."

Lemma 4 (the chain's structure; `sturmAux_consecutive_coprime`, `sturmAux_consecutive_succ`, `sturmAux_getLast_isUnit`)**.** *Consecutive members of the sequence are coprime; each consecutive triple satisfies* $p_{k+1} = -(p_{k-1} \bmod p_k)$*; and the last member is a nonzero constant.*

These three facts, proved by induction on the defining recursion, are exactly what Lemma 1 and the global induction consume.

How those results depend on one another is easier to see as a tree than as a list (Figure 4): the theorem splits into the per-point quantum and the constancy on the gaps, the quantum into the flank-reduction and the single head-drop, and the flank-reduction into its pure count-invariance and the antipodal-neighbour fact—with the structural facts feeding the global induction underneath.

Lean name	statement	role
`Sturm.sturm`	$\mathrm{V}(a) - \mathrm{V}(b) = \#\{a < x \le b : p(x) = 0\}$	Theorem 1
`sturmSeq`	the negated signed-remainder sequence	definition
`signVarAt`	sign changes of the list evaluated at x	definition
`signVarAt_drop_at_critical_point`	per-point quantum $\Delta\mathrm{V} \in \{0, 1\}$	Prop. 1
`flankReduce_chain_walk`	chain reduces to its non-vanishers	the wall (§4)
`FlankReduce.signChanges_eq`	flank-reduction preserves the count	combinatorial core
`sign_neighbours_opposite_at_interior_root`	antipodal neighbours at a zero	Lemma 1
`signVarAt_cons_head_drop`	a head sign-flip raises V by one	root crossing
`signVarAt_eq_of_no_root`	V constant off the critical set	constancy
`sturmAux_consecutive_coprime`	consecutive members coprime	structure
`sturmAux_consecutive_succ`	$p_{k+1} = -(p_{k-1} \bmod p_k)$	structure
`sturmAux_getLast_isUnit`	last member a nonzero constant	structure
`signVariations_eq_signChanges`	= Descartes' count (by `rfl`)	bridge (§5)

Table 2: The load-bearing results, all `sorry`-free in `Sturm.lean`. The axioms of every entry are `propext`, `Classical.choice`, `Quot.sound` only; no `native_decide`, no custom axiom.

```
Sturm.sturm                        (global induction on the critical set)
|- signVarAt_drop_at_critical_point   (per-point quantum, Sec. 3)
|    |- flankReduce_chain_walk       (prune interior c-vanishers)
|    |    |- FlankReduce.signChanges_eq  (count invariant)
|    |    `- sign_neighbours_opposite... (antipodal neighbours)
|    |- signVarAt_cons_head_drop      (the single step, p(c)=0)
|    `- signVarAt_eq_of_no_root       (no step, p(c)≠0)
|- signVarAt_eq_of_no_root           (flat gaps between criticals)
`- sturmAux_consecutive_{coprime,succ}  (chain structure)
+ sturmAux_getLast_isUnit
```

Figure 4: The dependency tree of `Sturm.sturm`. Reading top to bottom is reading Sections 2–4 in reverse: the pure, polynomial-free `FlankReduce.signChanges_eq` at the bottom, the Sturm-specific assembly at the top.

Cost and reuse

Because a formalization is judged as much by its cost as by its conclusion, here is the development measured (Table 3). The whole proof is a single file, `Sturm.lean`, of about 1,220 lines: 54 lemmas and theorems, 5 definitions, and one inductive relation. It depends on Mathlib alone—twelve modules, listed in the source—and on nothing from the earlier papers in this line; it is self-contained over the library. On a pre-built Mathlib it checks in about nine seconds (Lean `v4.30.0-rc2`), so the iteration loop during development was tight. The reuse is worth quantifying rather than estimating. Of the roughly 1,165 lines inside declarations, about 220 are *pure* sign/list combinatorics that mention no polynomial at all (`signChanges`, `wallCount`, `FlankReduce` and the list-surgery lemmas)—the layer that transfers verbatim to Descartes (Section 5)—and a further ≈ 200 are general polynomial sign-variation lemmas, reusable for any real-root sign argument. The genuinely Sturm-specific remainder is about 750 lines: the sequence's structure and the local-to-global crossing. So roughly a third of the development outlives its original purpose.

measure	value
Lean source	`Sturm.lean`, ≈ 1,220 lines, one file
declarations	54 lemmas/theorems, 5 definitions, 1 inductive
dependencies	Mathlib only (12 modules); none on prior work
compile time	≈ 9 s (file only, against a pre-built Mathlib), Lean `v4.30.0-rc2`
axioms	`propext`, `Classical.choice`, `Quot.sound`
line split	≈ 220 pure sign/list combinatorics + ≈ 200 general polynomial sign-variation (both reusable) + ≈ 750 Sturm-specific

Table 3: The development measured. The compile time is for checking `Sturm.lean` against a Mathlib that is already built; building Mathlib itself is the usual one-time cost and is not part of the contribution. The line split is counted from the declaration bodies (comments and blank lines excluded).

8 The boundary of the contribution

Let me be as plain about the limits as about the result.

The mathematics is Sturm's. I claim no new theorem and no new method. The sequence, the counting statement, the local crossing analysis, and the reduction to a difference of sign counts are all classical (Sturm [3]; the modern treatment in Basu–Pollack–Roy [4]). The contribution is the formalization, together with the reusable infrastructure—the sign-variation toolkit, the `FlankReduce` decoupling, the structural facts about the sequence—that any further real-root development in Lean would want.

Scope, stated plainly. I prove the theorem for *squarefree* p. This is the standard normalization and loses no generality in principle: for arbitrary p one passes to $p/\gcd(p, p')$, which is squarefree with the same distinct roots—but I have *not* formalized that reduction here, so the squarefree hypothesis is a genuine boundary of what is checked, not merely a convenience. I work over $\mathbb{R}$, the natural home of the result and what the intermediate-value and transversality arguments use, not over an abstract real-closed field. The endpoints are assumed non-roots of p; tail members *may* vanish at the endpoints, and that case is handled (Section 4). I count *distinct* roots; for squarefree p distinct and with-multiplicity coincide, so no separate multiplicity statement is needed, but neither is the Sturm–Tarski count with sign-weighted multiplicities [7, 8] attempted.

Where this sits, as a contribution. Judged as pure mathematics the novelty is essentially nil—Sturm's theorem is from 1829. Judged as a formalization it is, I think, worth recording: a foundational classical result genuinely absent from a major library, proved in full with reusable infrastructure and one piece of non-obvious engineering. The natural readership is the formalized-mathematics and verification community rather than the pure-mathematical one, and I have tried to write for it—hence the cost accounting, the reuse analysis, and the comparison that follows.

Prior formalizations, and what differs across systems. The priority claim is narrow and rests on a documented search, not a feeling. Sturm's theorem has been formalized before: in Coq, within Cohen's construction of the real algebraic numbers [5]; in Isabelle/HOL by Eberl [6] (a standalone development with an executable `sturm` proof method) and, in the sharper Sturm–Tarski form via the Cauchy index, by Li and Paulson [7, 8]; and in HOL Light. What I searched for (June 2026) and did not find was any formalization of the Sturm sequence or Sturm's theorem in *Lean*/Mathlib (Loogle and `grep`: `Polynomial.signVariations` for Descartes' rule exists; no `sturm`, no signed-remainder sequence). So: first in Lean, to the best of my knowledge; not first in any system.

A word on what the choice of system cost and saved, since that is what a proof-engineering reader will want.

What Mathlib already provided, making several steps almost free: the Euclidean-domain identity $p = (p \div q)\, q + (p \bmod q)$ with the degree bound $\deg(p \bmod q) < \deg q$, which makes the sequence's definition terminate without a separate measure argument; `Polynomial.Splits`, `roots` and `mem_roots'` for the root set; `separable` together with `PerfectField.separable_iff_squarefree`, which delivers "squarefree ⇒ coprime to the derivative" in one line; the intermediate value theorem (`intermediate_value_Icc`); and `Finset.max'` with the cardinality lemmas for the global induction. The analytic and algebraic substrate, in other words, was already there.

What had to be built from scratch. Absent everywhere in Lean were the sign-variation theory and the whole local-to-global crossing argument. The one genuinely fiddly step—independent of the system—was the list-recursion coupling that `FlankReduce` resolves (Section 4), not the analysis.

On comparing systems. I did not port Cohen's or Eberl's proofs, so I make no claim about relative proof length or difficulty: a fair comparison would mean re-doing the work in each system, which I have not done. What I can say is that the approaches differ in setting—Cohen's is embedded in exact real arithmetic, Eberl's is executable, Li–Paulson's targets the Cauchy index—and that, in Lean, the cost lived in the combinatorics of the count, not in the polynomial analysis Mathlib already supports.

On method. The formalization was carried out with an AI assistant (Claude, Anthropic), under the discipline used throughout this line of work: the worked example verified in exact arithmetic before formalizing (the script behind Figure 2), and every named theorem checked by `#print axioms` to rest only on `propext`, `Classical.choice`, `Quot.sound`. Correctness does not depend on the assistant: the Lean kernel checks every proof, and a reader who trusts the kernel need trust nothing else. The careful accounting in this section—the squarefree boundary named as a boundary, the prior systems credited—is part of what the method, used honestly, should produce.

9 What this paper still lacks

Sturm's theorem is now in the library; the more interesting question is what its absence was hiding, and what this account leaves undone. I close with the questions I would rather carry forward than pretend to have settled—each a place where the present work stops short.

1. *The squarefree boundary.* I leaned on squarefreeness and did not formalize the passage to $p/\gcd(p, p')$. That reduction is routine on paper; is it routine in Lean, or does the gcd of a polynomial and its derivative hide the kind of degree-bookkeeping that, as in Section 4, only looks routine until a machine asks? Closing it would make the theorem unconditional, and is the first thing I would do next.

2. *The two counts, beyond `rfl`.* Descartes' and Sturm's variation counts turned out to be the same function (Section 5). That is a coincidence of *definition*; is there a theorem behind it—a single statement of which Descartes' bound and Sturm's exact count are two instances, and which a library could hold once instead of twice? The shared core is suggestive, but I have only unified the bookkeeping, not the theorems.

3. *From count to certificate.* The proof says *how many* roots lie in $(a, b]$; it does not, as it stands, hand back the roots themselves, nor a checkable certificate of the count for a skeptical downstream tool. The building blocks for root isolation are all here (Section 5); what is the smallest honest bridge from this theorem to a verified root-isolation procedure—and how much of Sturm–Tarski [8] would it want along the way?

4. *What the decoupling really is.* The `FlankReduce` relation separated the combinatorics of pruning a list from the algebra of why each pruning is legal (Section 4). It worked cleanly enough that I suspect it is an instance of something more general—a pattern for any argument where a count is invariant under deletions licensed by local side-conditions. Is "invariance of a tally under locally-justified deletion" a reusable lemma in its own right, or did it merely happen to fit here? I do not yet know, and that uncertainty is the honest state of it.

Reproducing

```
git clone https://github.com/karlesmarin/godsil-gutman-lean
cd godsil-gutman-lean && lake exe cache get
lake env lean Sturm.lean
```

Lean `v4.30.0-rc2`, Mathlib pin `701fb6e9`. (The repository `godsil-gutman-lean` is the umbrella for this series of Lean formalizations; the present work is the single file `Sturm.lean` within it.) The theorem is `Sturm.sturm` in `Sturm.lean`; its supports are in Table 2. It is checked by `#print axioms` to depend only on `propext`, `Classical.choice`, `Quot.sound`. The worked example of Figure 2 is re-derived, and every sign and count asserted against exact rational arithmetic, before the figure is drawn.

References

[1] R. Descartes, *La Géométrie* (1637); the rule of signs bounds the number of positive roots by the number of sign changes of the coefficients.

[2] F. D. Budan de Boislaurent, *Nouvelle méthode pour la résolution des équations numériques* (1807); J. Fourier, *Analyse des équations déterminées* (posth. 1831). The Budan–Fourier theorem bounds the roots in an interval by a sign-variation difference.

[3] C. Sturm, Mémoire sur la résolution des équations numériques, *Mémoires présentés par divers savants à l'Académie Royale des Sciences* **6** (1835) 271–318; presented to the Académie in 1829.

[4] S. Basu, R. Pollack, M.-F. Roy, *Algorithms in Real Algebraic Geometry*, 2nd ed., Algorithms and Computation in Mathematics **10**, Springer, 2006 (Sturm's theorem and sign-variation theory, Ch. 2).

[5] C. Cohen, Construction of real algebraic numbers in Coq, in *Interactive Theorem Proving (ITP 2012)*, LNCS **7406**, Springer, 2012, 67–82.

[6] M. Eberl, Sturm's Theorem, *Archive of Formal Proofs* (2014), https://isa-afp.org/entries/Sturm_Sequences.html.

[7] W. Li, L. C. Paulson, A formal proof of the Sturm–Tarski theorem, *Archive of Formal Proofs* (2014), https://isa-afp.org/entries/Sturm_Tarski.html.

[8] W. Li, The Budan–Fourier theorem and counting real roots with multiplicity, preprint, 2018, arXiv:1811.11093; see also W. Li, G. O. Passmore, L. C. Paulson, Deciding univariate polynomial problems using untrusted certificates in Isabelle/HOL, *J. Automated Reasoning* **62** (2019) 69–91.

[9] The mathlib Community, The Lean mathematical library, in *Certified Programs and Proofs (CPP 2020)*, ACM, 2020, 367–381.

[10] C. Marín, *Random Signs into Matchings: A Godsil–Gutman Identity, Formalized in Lean 4*, Zenodo, 2026, doi:10.5281/zenodo.20517350.

[11] C. Marín, *The Inward Bow of a Real-Rooted Polynomial: Newton's Inequalities, Machine-Checked in Lean 4*, Zenodo, 2026, doi:10.5281/zenodo.20693064.