SUBDIVISION ALGORITHMS AND REAL ROOT ISOLATION

Simon SEPIOL-DUCHEMIN   Joshua SETIA

Supervised by Mohab SAFEY EL DIN

May 2025

Outline

Table of Contents

The Problem: Finding Polynomial Roots

- Fundamental in CS, vision, robotics, computational geometry.
- Applications often involve high-degree polynomials, large coefficients.
- Abel-Ruffini: No general algebraic solution (radicals) for degree ≥ 5.

The Problem: Finding Polynomial Roots

- Fundamental in CS, vision, robotics, computational geometry.
- Applications often involve high-degree polynomials, large coefficients.
- Abel-Ruffini: No general algebraic solution (radicals) for degree ≥ 5.
- **Our Goal:** Real Root Isolation.
  - Input: Square free univariate polynomial $f(x) \in \mathbb{Z}[x]$.
  - Output: Disjoint intervals $[p_i, q_i]$ with rational endpoints, each real root is isolated in one interval.

Table of Contents

Descartes' Rule of Signs

Let $f(x) = \sum_{i=0}^{n} f_i x^{b_i}$ (with $0 = b_0 < b_1 < \cdots < b_n$, $f_i \neq 0$).

- $V(f)$: Number of sign changes in the sequence of coefficients $(f_0, f_1, \ldots, f_n)$.
- $Z_+(f)$: Number of positive real roots of f (counting multiplicity).

Theorem (Descartes' Rule of Signs)

$Z_+(f) = V(f) - 2k$, for some non-negative integer k.

Crucial Implications for Isolation:

- If $V(f) = 0 \implies Z_+(f) = 0$ (no positive roots).
- If $V(f) = 1 \implies Z_+(f) = 1$ (exactly one positive root).

Table of Contents

Algorithm Overview (1/2)

1. **Compute bound B**
2. **Normalize:** Compute $Q(x) = f(B \cdot x)$
3. **Bisection algorithm:** Isolate roots on $[0, 1]$

Bounds on Real Roots

- **Lagrange's Bound**:

$$|z| \leq \max\left\{1, \sum_{i=0}^{n-1} \left| f_i / f_n \right| \right\}.$$

- **Cauchy's Bound**:

$$|z| \leq 1 + \max_{0 \leq i < n} \left| f_i / f_n \right|.$$

- **Local-Max Linear Bound** [Vigklas, 2010]:

$$|z| = \max_{\{k | f_k < 0\}} \left(\frac{-f_k \, 2^{t_m}}{f_m} \right)^{1/(m-k)}.$$

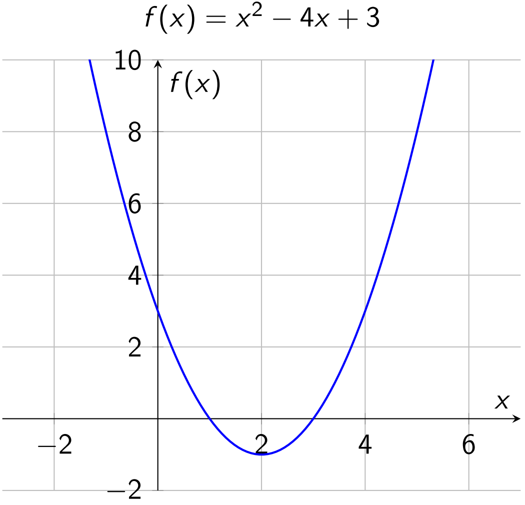
Algorithm Overview (2/2)

1. **Compute bound B**
2. **Normalize:** Compute $Q(x) = f(B \cdot x)$
3. **Bisection algorithm:** Isolate roots on $[0, 1]$

Bisection Algorithm

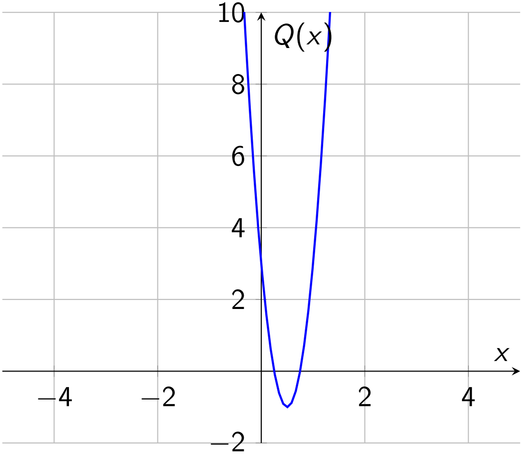
1. Handle root at 0.
2. $Q(x) \to Q_\infty(x) = Q(\frac{1}{x+1})$.
3. Count $V(Q_\infty)$.
4. **Decide:**
   - If $V(Q_\infty) = 0$: No roots for f in $(0, 1)$.
   - If $V(Q_\infty) = 1$: Exactly one root for f in $(0, 1)$. Interval found!
   - If $V(Q_\infty) > 1$: **Bisect and Recurse**.
     - On $(0, 1/2]$: Analyze $Q_{left} = Q(x/2)$.
     - On $(1/2, 1]$: Analyze $Q_{right} = Q((x+1)/2)$.

Example



Example

$$f(4x) = 16x^2 - 16x + 3 = Q(x)$$



Example

$$Q\left(\frac{1}{x+1}\right) = 3x^2 - 10x + 3$$

$$\Rightarrow \quad 2 \textit{ sign changes}$$

$$Q(\tfrac{x}{2}) = 4x^2 - 8x + 3 = Q_{left}(x) \quad \Big| \quad Q(\tfrac{x+1}{2}) = 4x^2 - 1 = Q_{right}(x)$$

Example

$$Q\left(\frac{1}{x+1}\right) = 3x^2 - 10x + 3$$

$\Rightarrow$ *2 sign changes*

$Q(\frac{x}{2}) = 4x^2 - 8x + 3 = Q_{left}(x)$	$Q(\frac{x+1}{2}) = 4x^2 - 1 = Q_{right}(x)$
$Q_{left}(\frac{1}{x+1}) = 3x^2 - 2x - 1$	$Q_{right}(\frac{1}{x+1}) = -x^2 - 2x + 3$
$\Rightarrow$ *1 sign change*	$\Rightarrow$ *1 sign change*
$(0, \frac{1}{2}] \to (0, 2]$	$(\frac{1}{2}, 1] \to (2, 4]$

Key Polynomial Transformations

The algorithm heavily relies on efficient polynomial transformations:

- **Scaling:** $f(x/2)$
- **Shift and Scale:** $f(\frac{x+1}{2})$
- **Reversal and Shift:** $f(\frac{1}{1+x})$

Efficient **Taylor Shift** $f(x+1)$ is critical.

Table of Contents

Taylor Shift

Horner's Method:

- Iterative evaluation.
- $f(x+1) = f_0 + (x+1)(f_1 + \cdots + (x+1)f_n)$, it takes n steps.
- Complexity: $\mathcal{O}(n^2)$ arithmetic operations.

Taylor Shift

Horner's Method:

- Iterative evaluation.
- $f(x+1) = f_0 + (x+1)(f_1 + \cdots + (x+1)f_n)$, it takes n steps.
- Complexity: $\mathcal{O}(n^2)$ arithmetic operations.

Divide and Conquer:

- $f = f^{(0)} + x^{n/2} f^{(1)}$.
- $f(x+1) = f^{(0)}(x+1) + (x+1)^{n/2} f^{(1)}(x+1)$.
- Complexity: $\mathcal{O}(\mathcal{M}(n) \log n)$.
- $\mathcal{M}(n)$: Univariate polynomial multiplication cost.

Taylor Shift: Our Implementation

Idea:

- Unchanged degree during bisection algorithm.
- Same subdivision pattern
- Same powers of $(x+1)$

Solution:

- Precompute subdivision and $(x+1)$ powers once.
- Use an iterative Taylor shift with that data.

Benchmark Methodology

Tested polynomials characteristics :

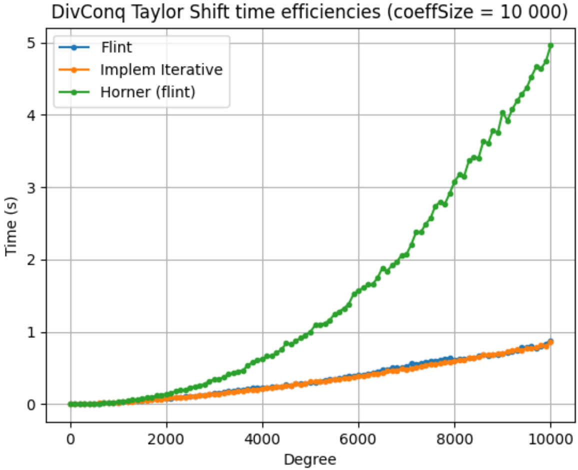
- Dense
- Random coefficients

Polynomial variation :

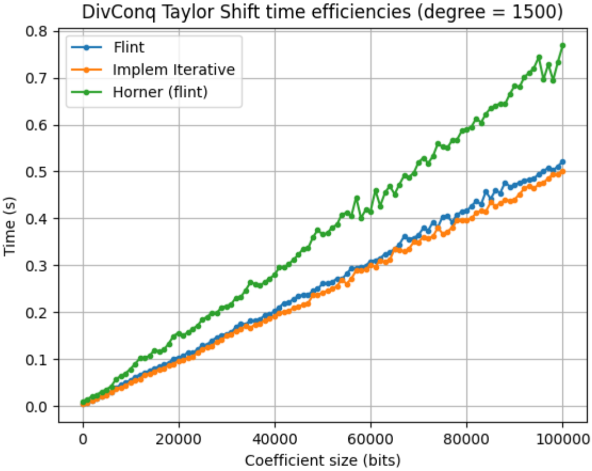
- Growing degree with fixed coefficient bit size
- Growing coefficient bit size with fixed degree

Generate a reusable polynomial base for all benchmarks.

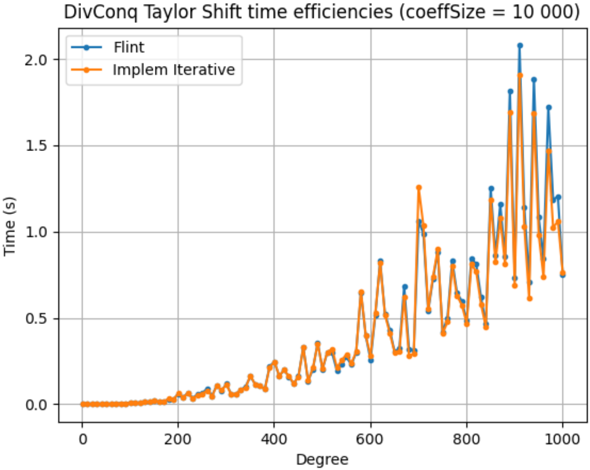
Taylor Shift Benchmarks – Varying Degree



Taylor Shift Benchmarks – Varying Coefficient Size



Taylor Shift in Isolation – Varying Degree



Taylor Shift in Isolation – Varying Coefficient Size

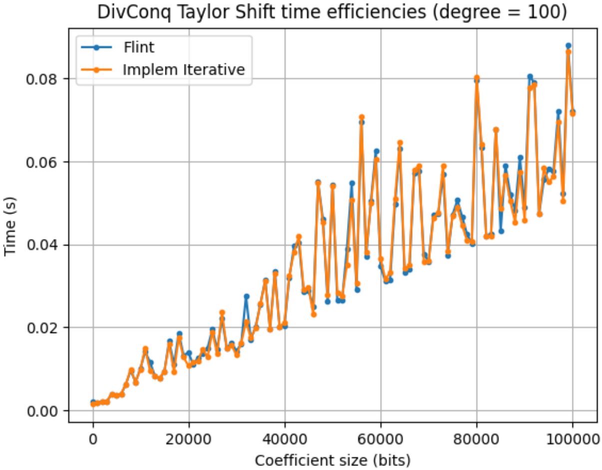


Table of Contents

Coefficient Truncation for Faster Sign Checks

Idea:

- Descartes' rule only needs coefficient *signs*.
- Taylor shift $f(x+1)$ can increase coefficient bit-size by at most the degree of the polynomial.
- Can we use truncated coefficients for $f(\frac{1}{1+x})$?

Taylor Shift bit growth Lemma

Lemma

T(f) the biggest coefficient bit size.
$T(f(x+1)) < T(f) + d$

Lemma

$l \in \mathbb{Z}, f_t(x) = \sum_{i=0}^{d} \frac{f_i}{2^l}$, if

$$|f_t(x+1)_i| > 2^d,$$

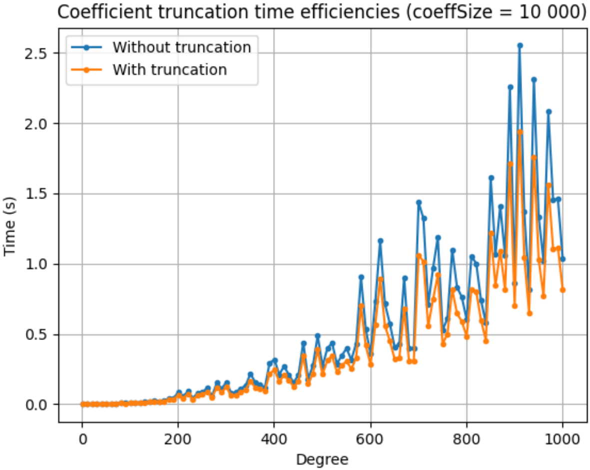
then the sign of the ith coefficient of $f(x+1)$ matches that of $f_t(x+1)_i$.

Coefficient Truncation for Faster Sign Checks

Method:

1. Truncate l bits
2. Compute $f_t(\frac{1}{x+1})$.
3. Count the number of sign changes, taking the truncation into account.
4. If not enough reliable coefficient, start over with full precision

Truncation Benchmarks – Varying Degree



Truncation Benchmarks – Varying Coefficient Size

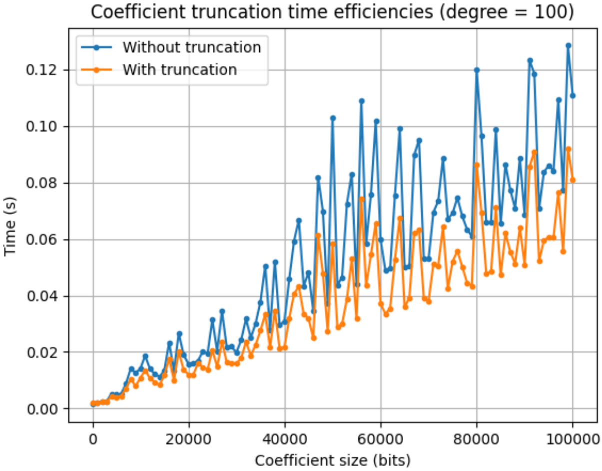


Table of Contents

Conclusion

Achievements:

- Practical implementation of Descartes Bisection algorithm
- Optimized iterative Taylor Shift
- Optimization through coefficient truncation
- Validated complexities and performance through benchmarks.

Next improvement:

- Parallelization

References

Basu, S., Pollack, R., Roy, M.-F. (2006). *Algorithms in Real Algebraic Geometry*. Springer.
Stewart, I. (2004). *Galois Theory* (3rd ed.). Chapman and Hall/CRC.
Rudin, W. (1976). *Principles of Mathematical Analysis* (3rd ed.). McGraw-Hill.
Sagraloff, M., Mehlhorn, K. (2015). *Computing Real Roots of Real Polynomials*. arXiv:1308.4088.
Kobel, A., Rouillier, F., Sagraloff, M. (2016). Computing Real Roots... For Real! *ISSAC '16*, ACM.
Rouillier, G., Zimmermann, P. (2004). Efficient Isolation of Polynomial Real Roots. *CAA Conference*.
Von zur Gathen, J., Gerhard, J. (1997). Fast Algorithms for Taylor Shifts. *ISSAC '97*.

Panagiotis S. Vigklas. *Upper Bounds on the Values of the Positive Roots of Polynomials*. PhD thesis, University of Thessaly, Volos, Greece, 2010. Supervisor: Dr. Alkiviadis Akritas.

Thank You

Questions?