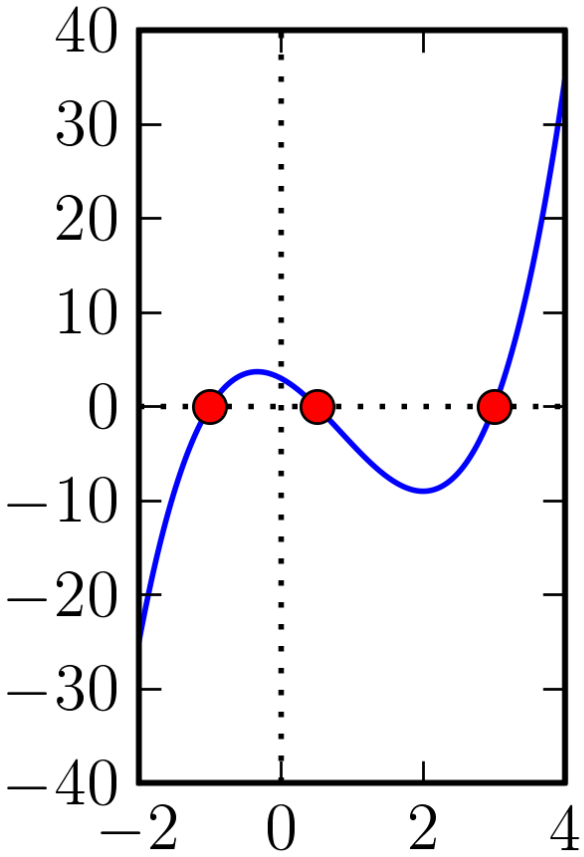
Fast Real Root Isolation

Carl Witty

Sage Days 4

Outline

What is real root isolation?

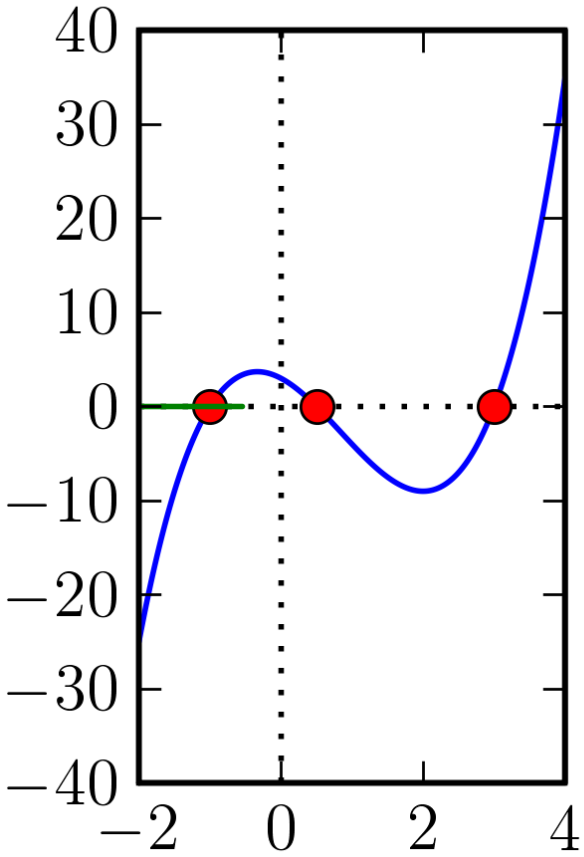


- Given a polynomial, we find an interval with rational endpoints containing each real root (such that the intervals do not overlap).

My algorithm finds:
$[-2, -\frac{35}{64}]$ $[0, 1]$ $[2, 4]$
(The actual roots are -1, $\frac{1}{2}$, and 3.)

- Most of my discussion will focus on squarefree polynomials with integral coefficients.
- The problem is fairly easy if the degree is low and the coefficients are small, but can be very difficult for polynomials of high degree.

What is real root isolation?



- Given a polynomial, we find an interval with rational endpoints containing each real root (such that the intervals do not overlap).

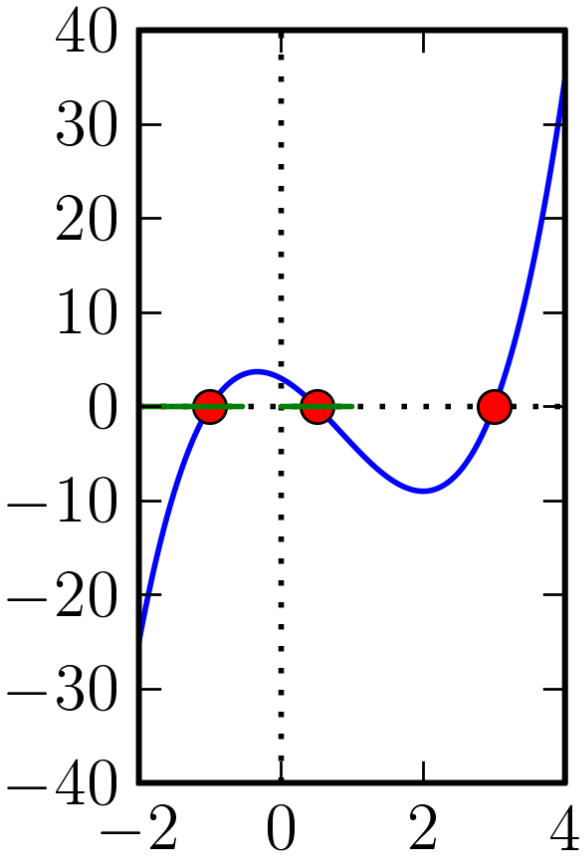
My algorithm finds:

$[-2, -\frac{35}{64}]$ $[0, 1]$ $[2, 4]$

(The actual roots are -1, $\frac{1}{2}$, and 3.)

- Most of my discussion will focus on squarefree polynomials with integral coefficients.
- The problem is fairly easy if the degree is low and the coefficients are small, but can be very difficult for polynomials of high degree.

What is real root isolation?

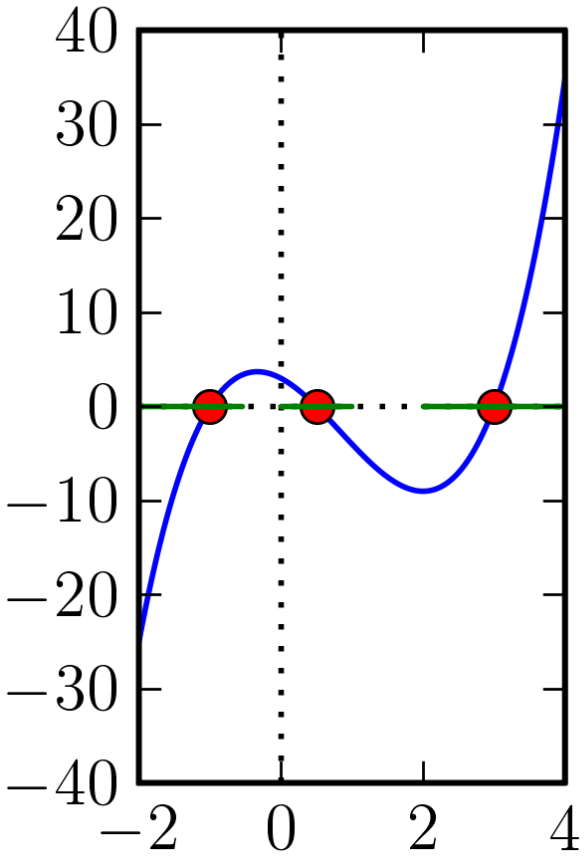


- Given a polynomial, we find an interval with rational endpoints containing each real root (such that the intervals do not overlap).

My algorithm finds:
$[-2, -\frac{35}{64}]$ $[0, 1]$ $[2, 4]$
(The actual roots are -1, $\frac{1}{2}$, and 3.)

- Most of my discussion will focus on squarefree polynomials with integral coefficients.
- The problem is fairly easy if the degree is low and the coefficients are small, but can be very difficult for polynomials of high degree.

What is real root isolation?

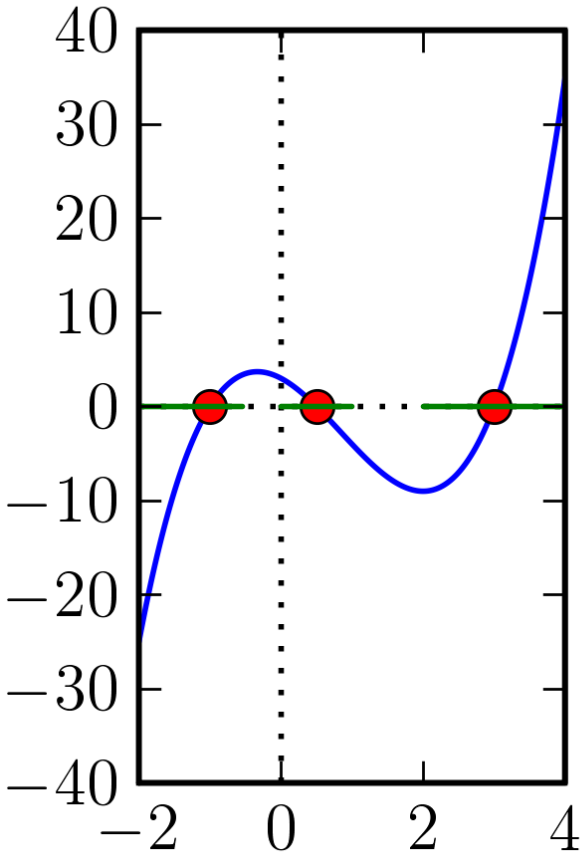


- Given a polynomial, we find an interval with rational endpoints containing each real root (such that the intervals do not overlap).

My algorithm finds:
$[-2, -\frac{35}{64}]$ $[0, 1]$ $[2, 4]$
(The actual roots are -1, $\frac{1}{2}$, and 3.)

- Most of my discussion will focus on squarefree polynomials with integral coefficients.
- The problem is fairly easy if the degree is low and the coefficients are small, but can be very difficult for polynomials of high degree.

What is real root isolation?

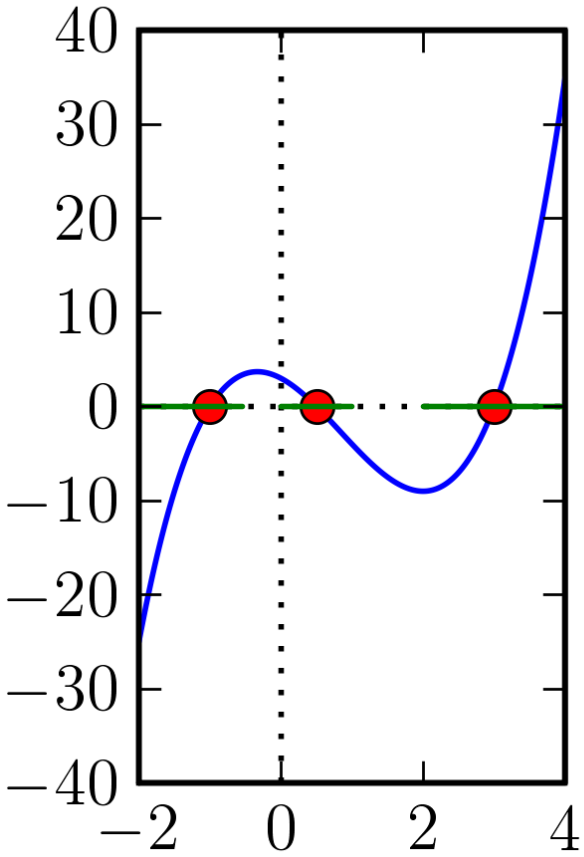


- Given a polynomial, we find an interval with rational endpoints containing each real root (such that the intervals do not overlap).

My algorithm finds:
$[-2, -\frac{35}{64}]$ $[0, 1]$ $[2, 4]$
(The actual roots are -1, $\frac{1}{2}$, and 3.)

- Most of my discussion will focus on squarefree polynomials with integral coefficients.
- The problem is fairly easy if the degree is low and the coefficients are small, but can be very difficult for polynomials of high degree.

What is real root isolation?



- Given a polynomial, we find an interval with rational endpoints containing each real root (such that the intervals do not overlap).

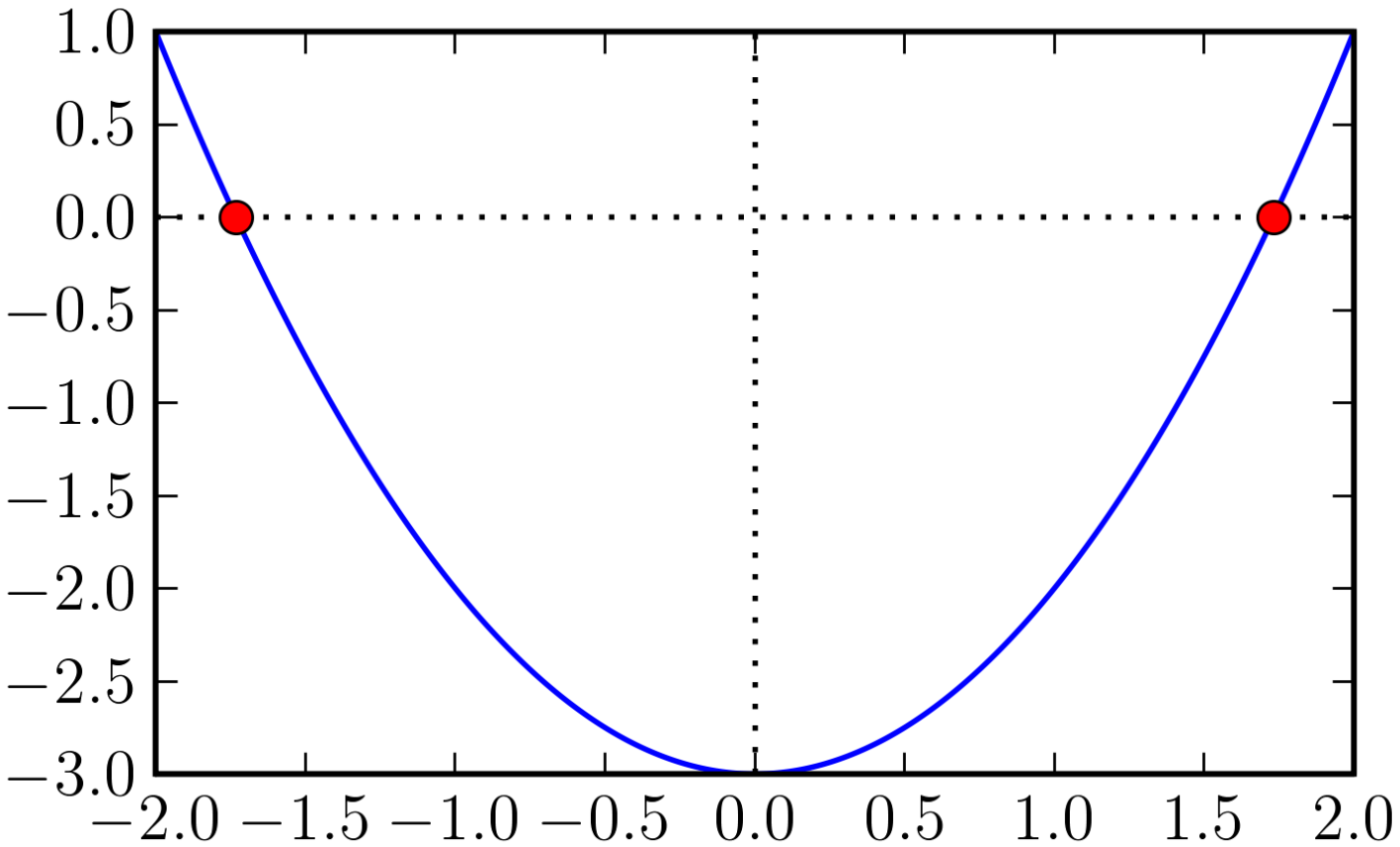
My algorithm finds:
$[-2, -\frac{35}{64}]$ $[0, 1]$ $[2, 4]$
(The actual roots are -1, $\frac{1}{2}$, and 3.)

- Most of my discussion will focus on squarefree polynomials with integral coefficients.
- The problem is fairly easy if the degree is low and the coefficients are small, but can be very difficult for polynomials of high degree.

Constructing a polynomial

Consider the polynomial:

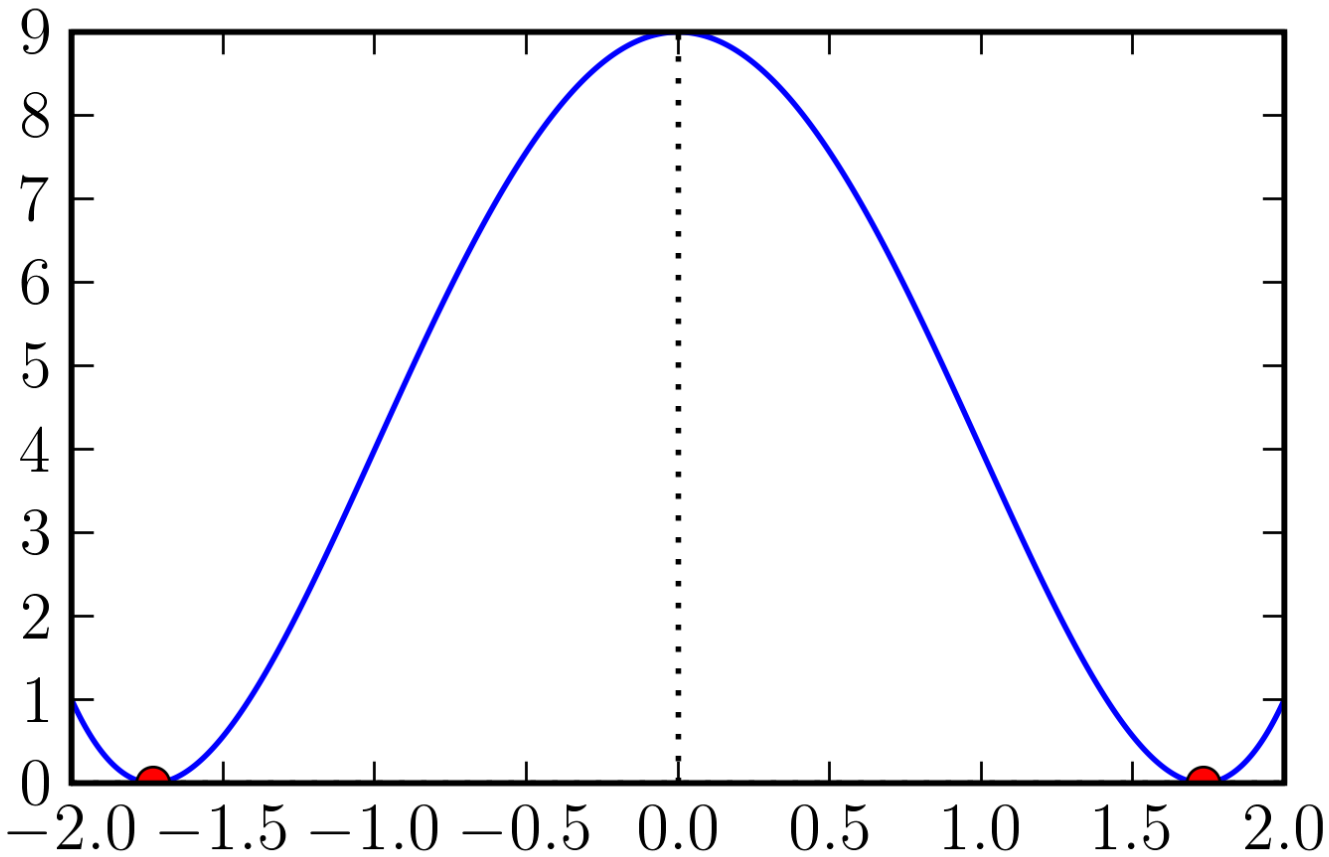
$$x^5(x^2-3)^2-1$$



Constructing a polynomial

Consider the polynomial:

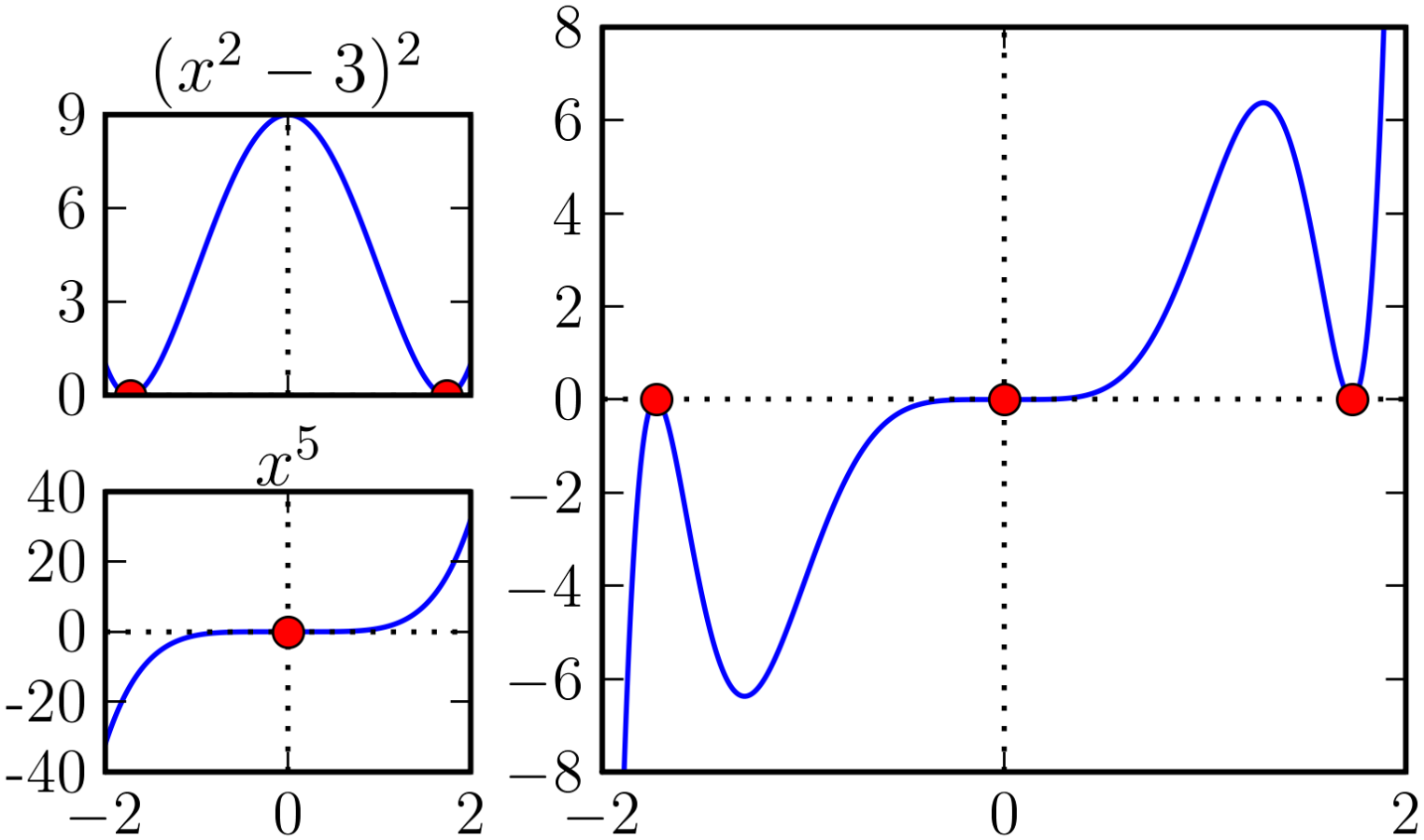
$$x^5(x^2-3)^2-1$$



Constructing a polynomial

Consider the polynomial:

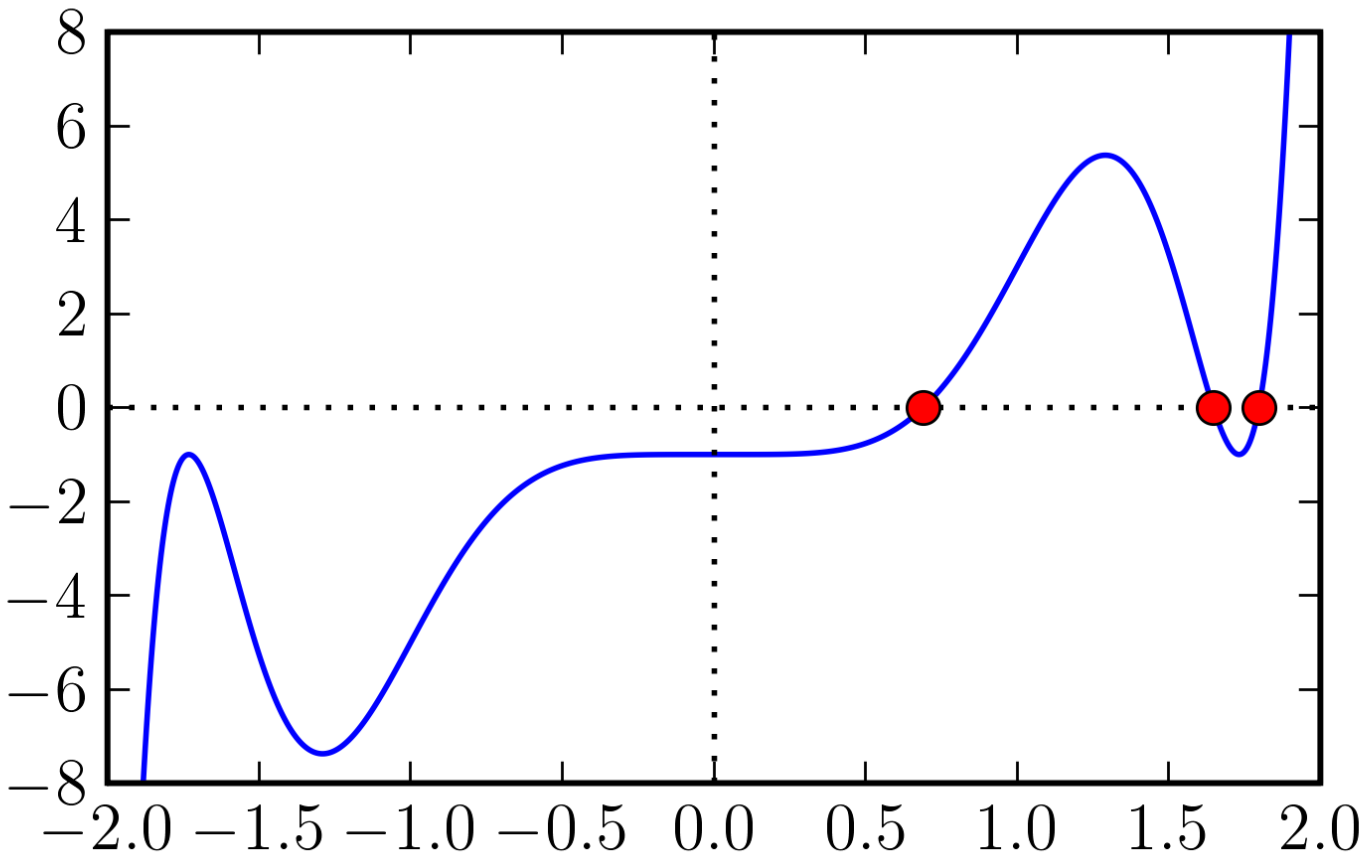
$$x^5(x^2-3)^2-1$$



Constructing a polynomial

Consider the polynomial:

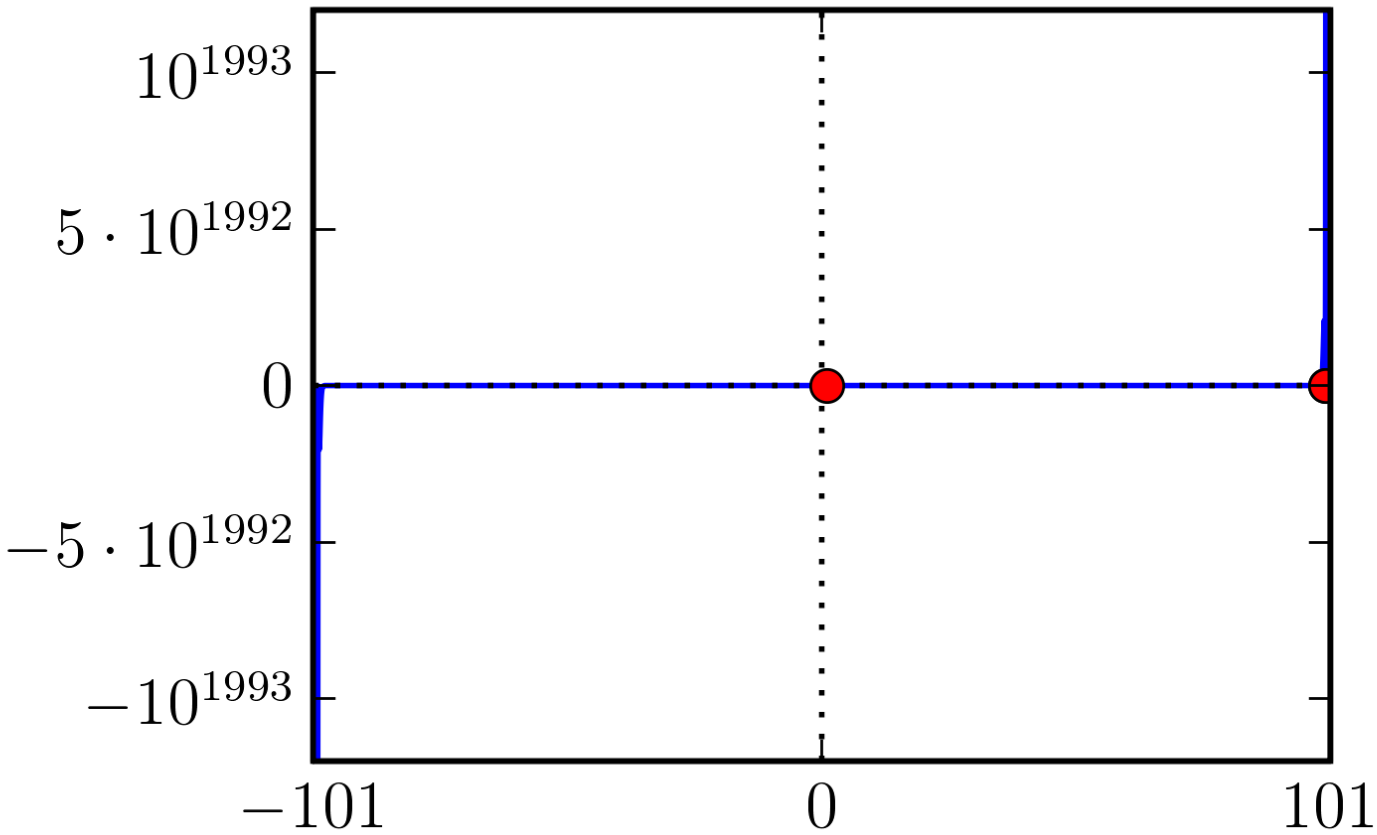
$$x^5(x^2-3)^2-1$$



Constructing a polynomial

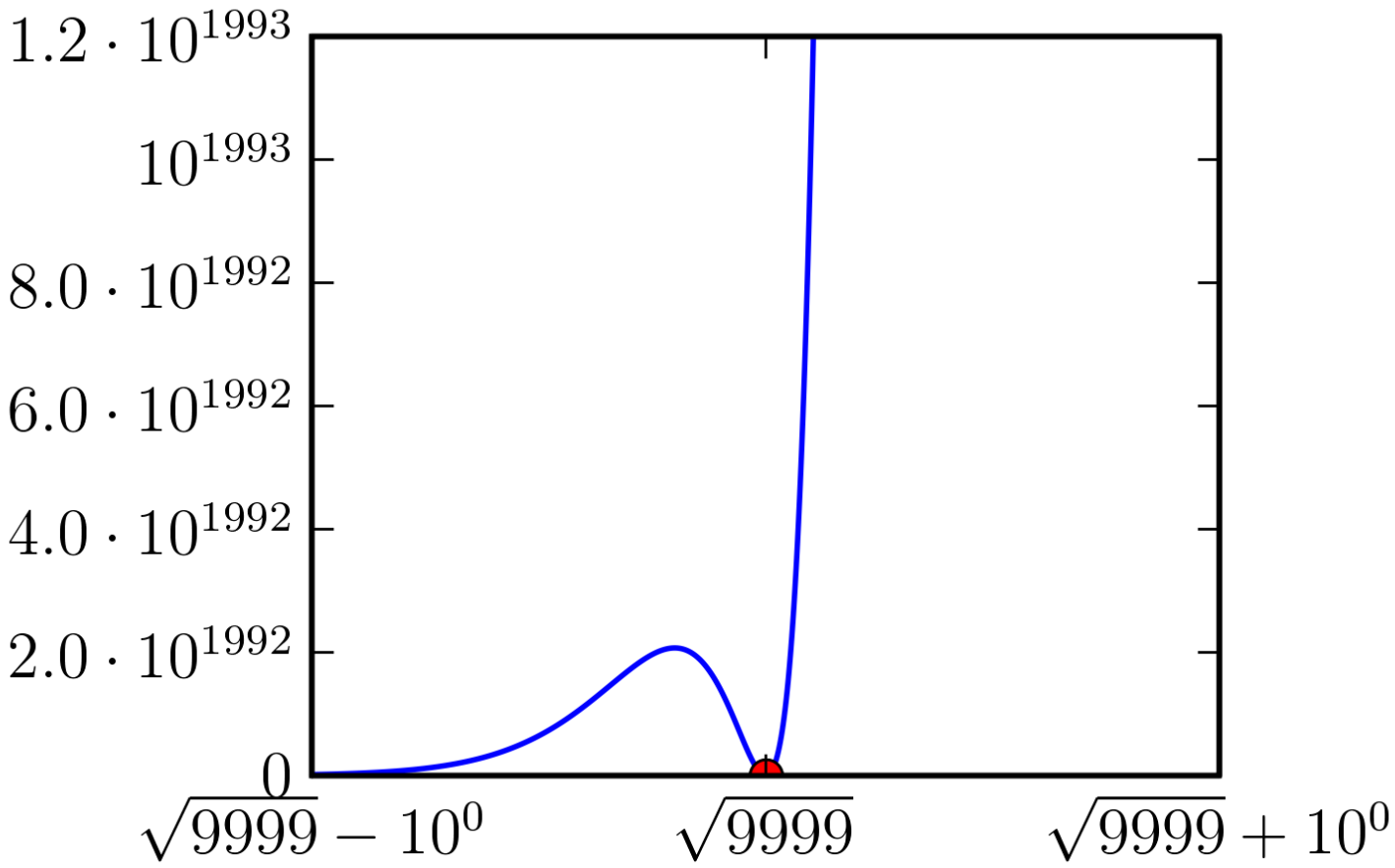
Consider the polynomial:

$$x^{995}(x^2 - 9999)^2 - 1$$



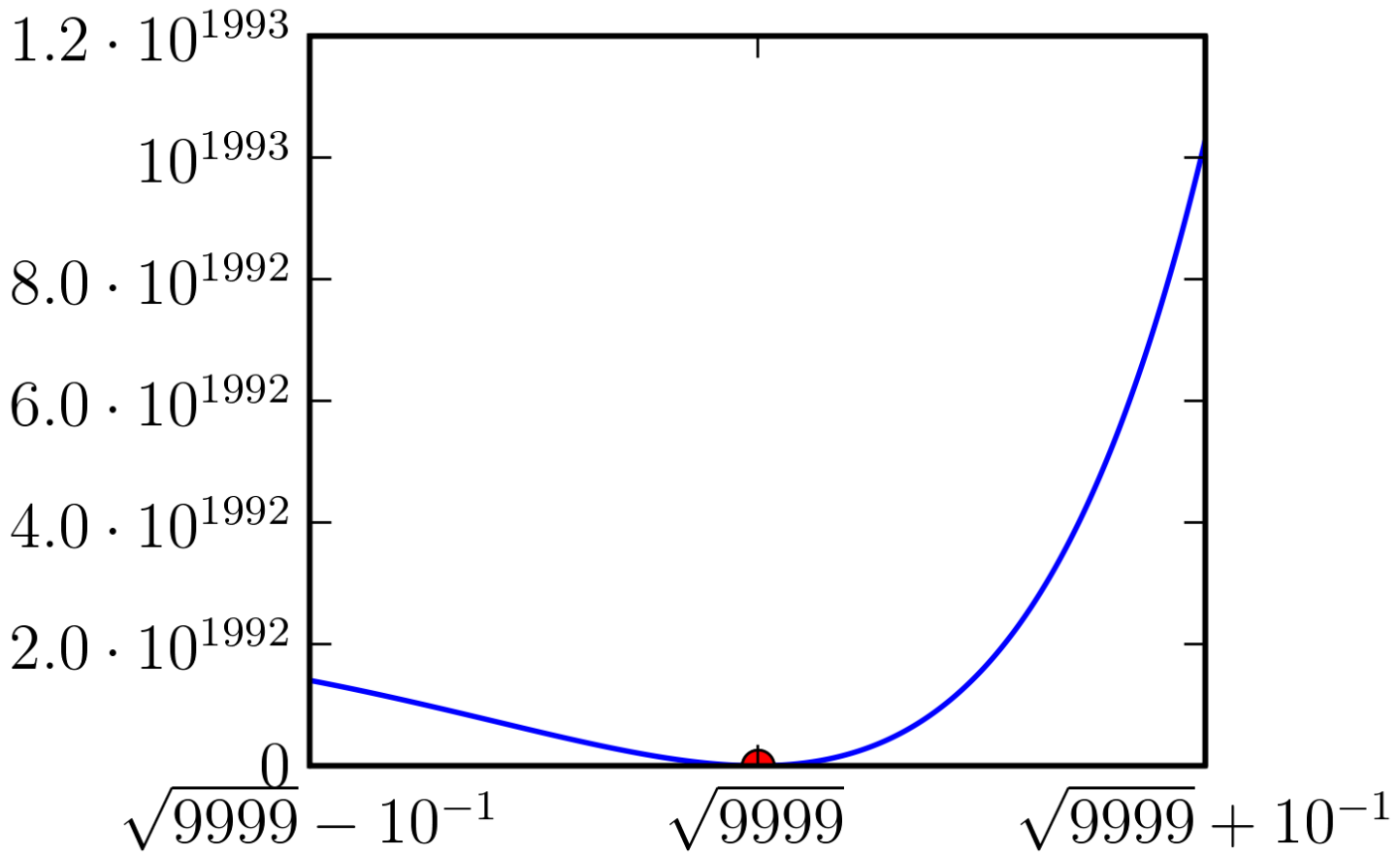
Zooming in on the roots

Let's take a closer look at those roots:



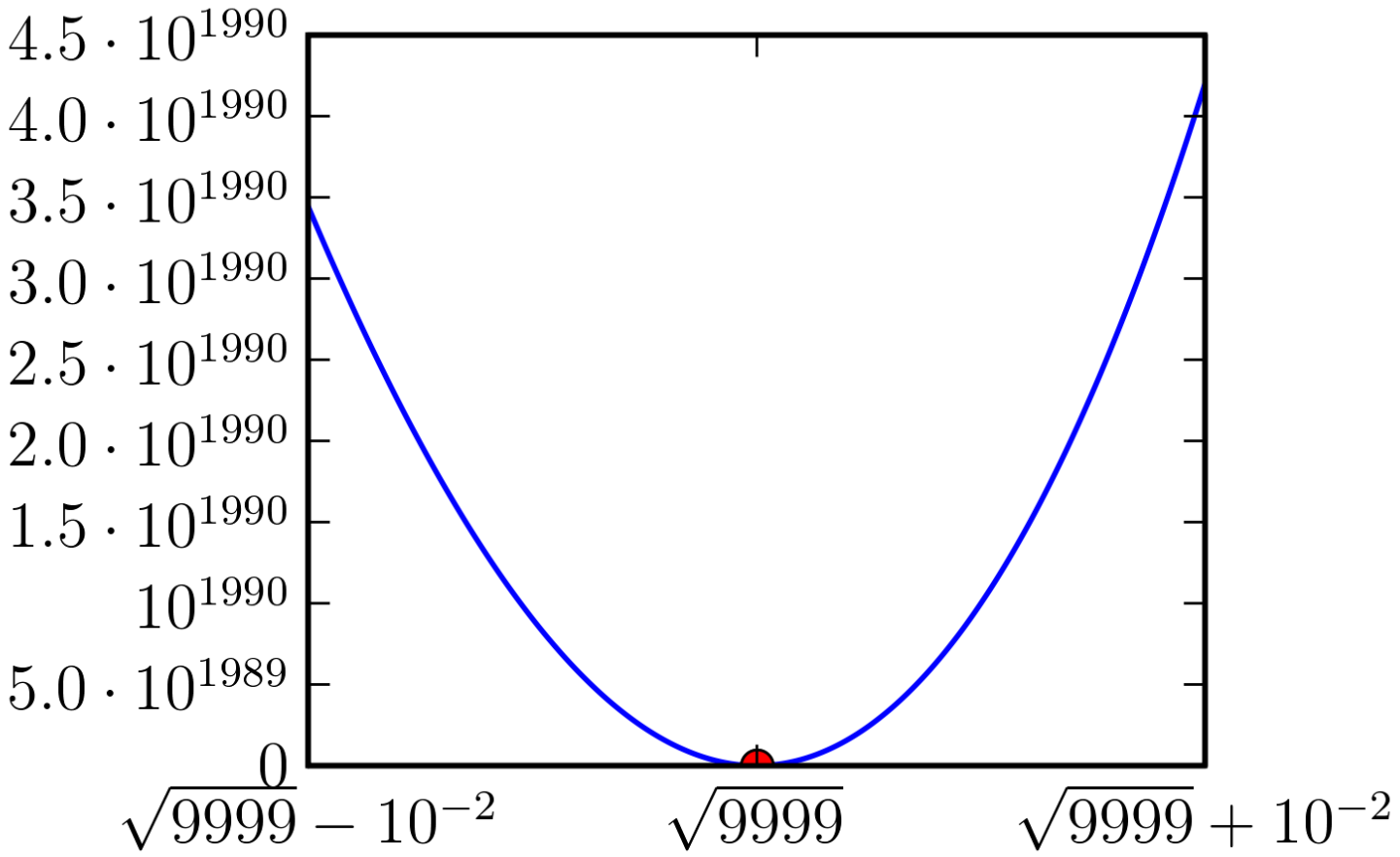
Zooming in on the roots

Let's take a closer look at those roots:



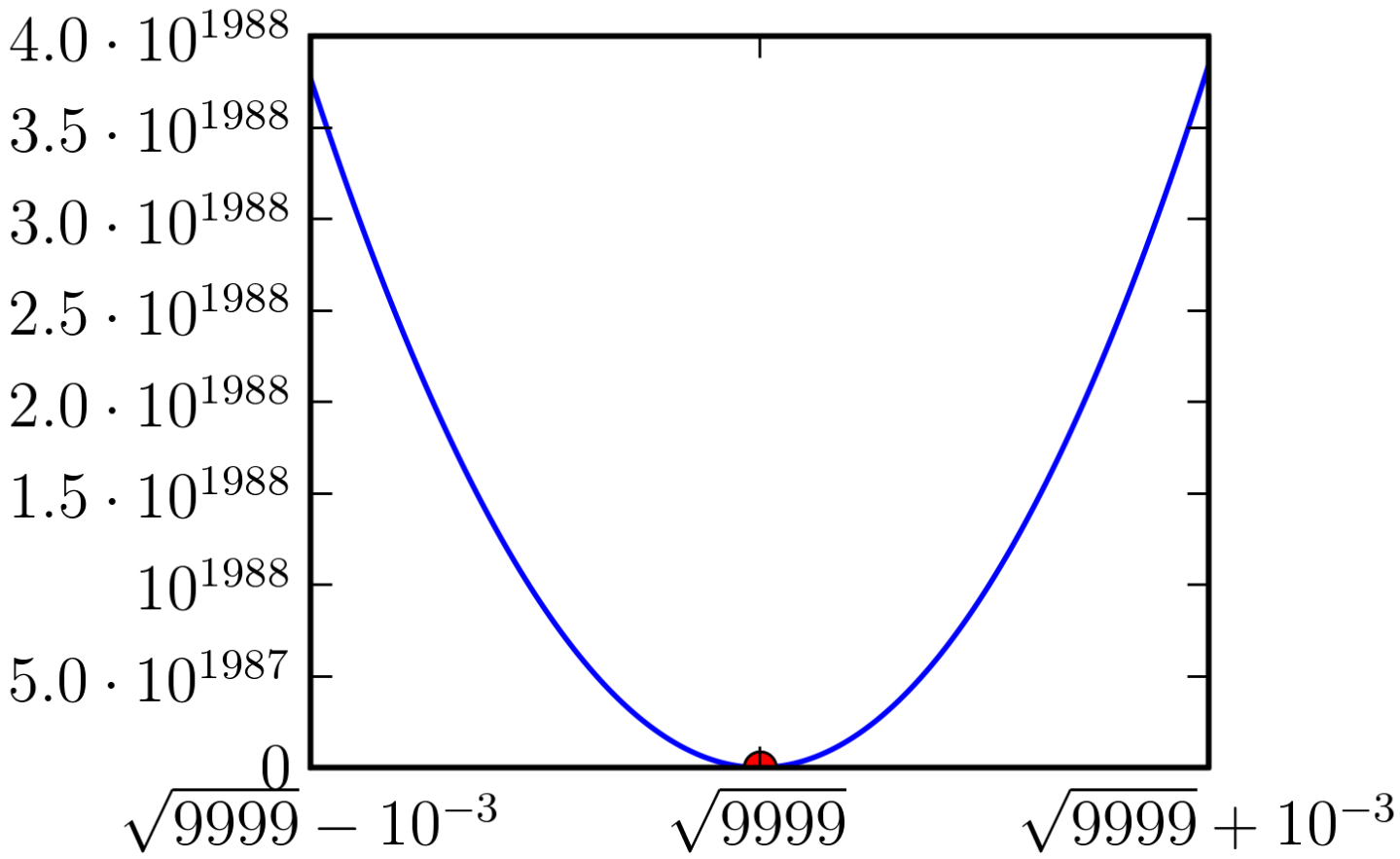
Zooming in on the roots

Let's take a closer look at those roots:



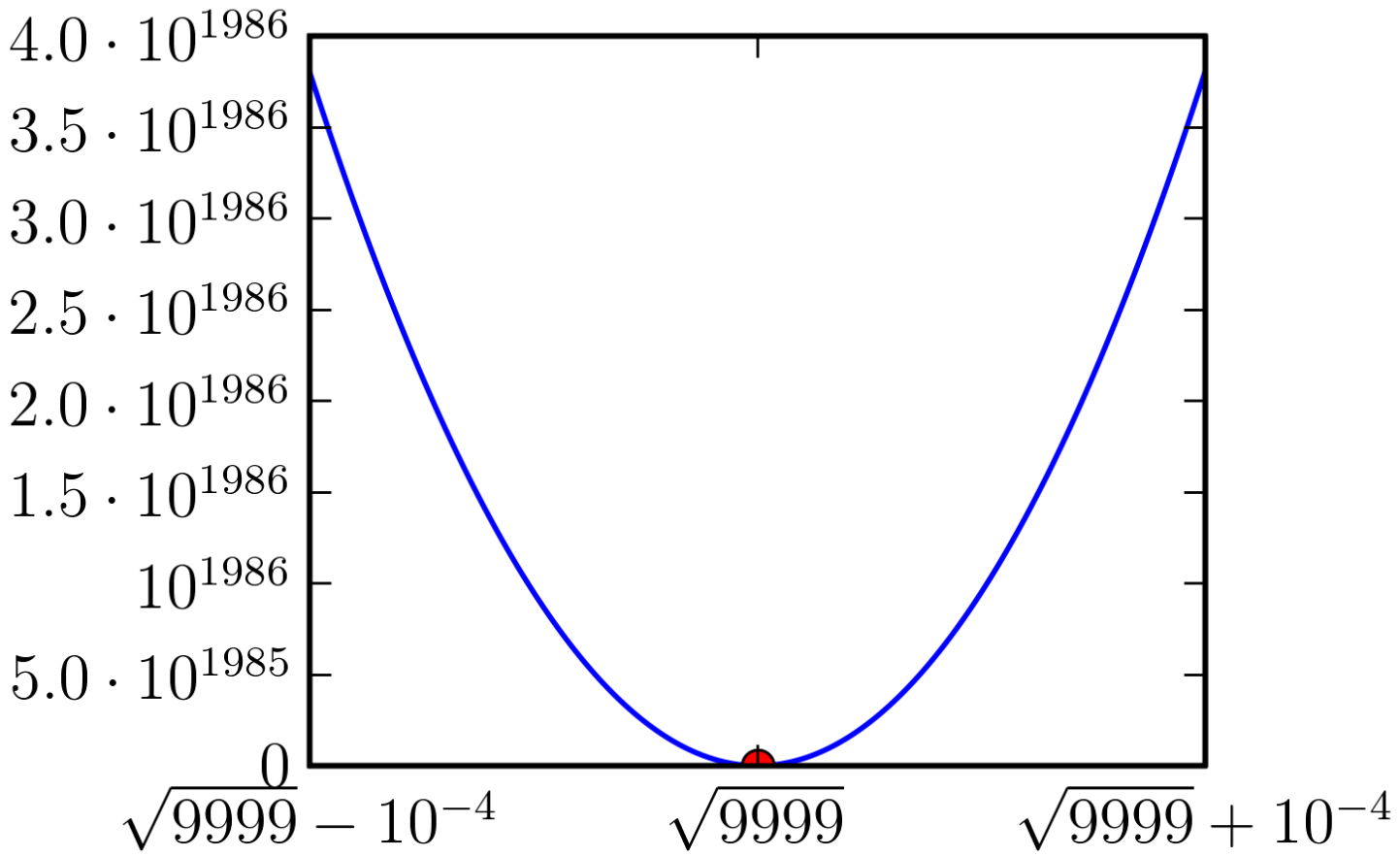
Zooming in on the roots

Let's take a closer look at those roots:



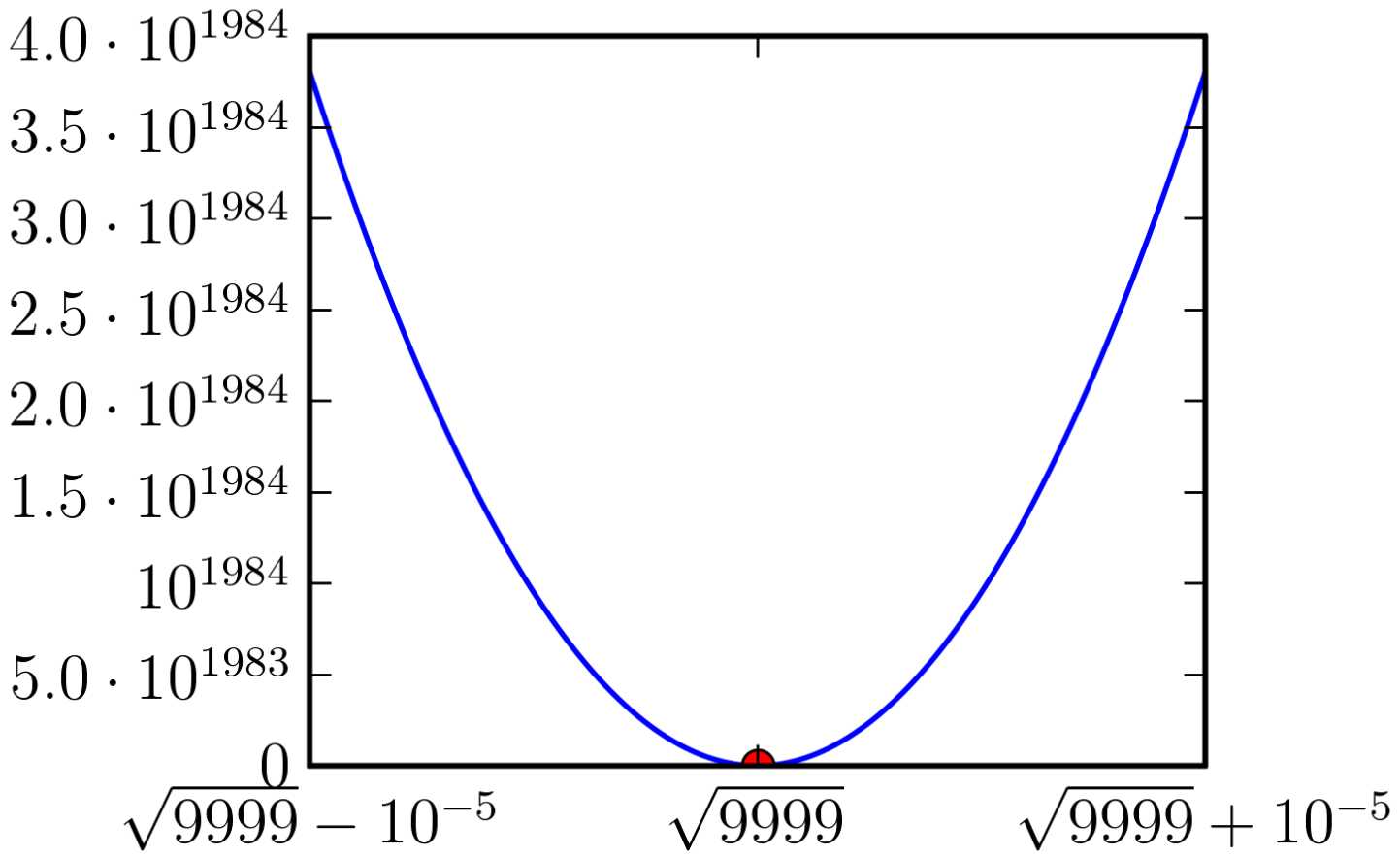
Zooming in on the roots

Let's take a closer look at those roots:



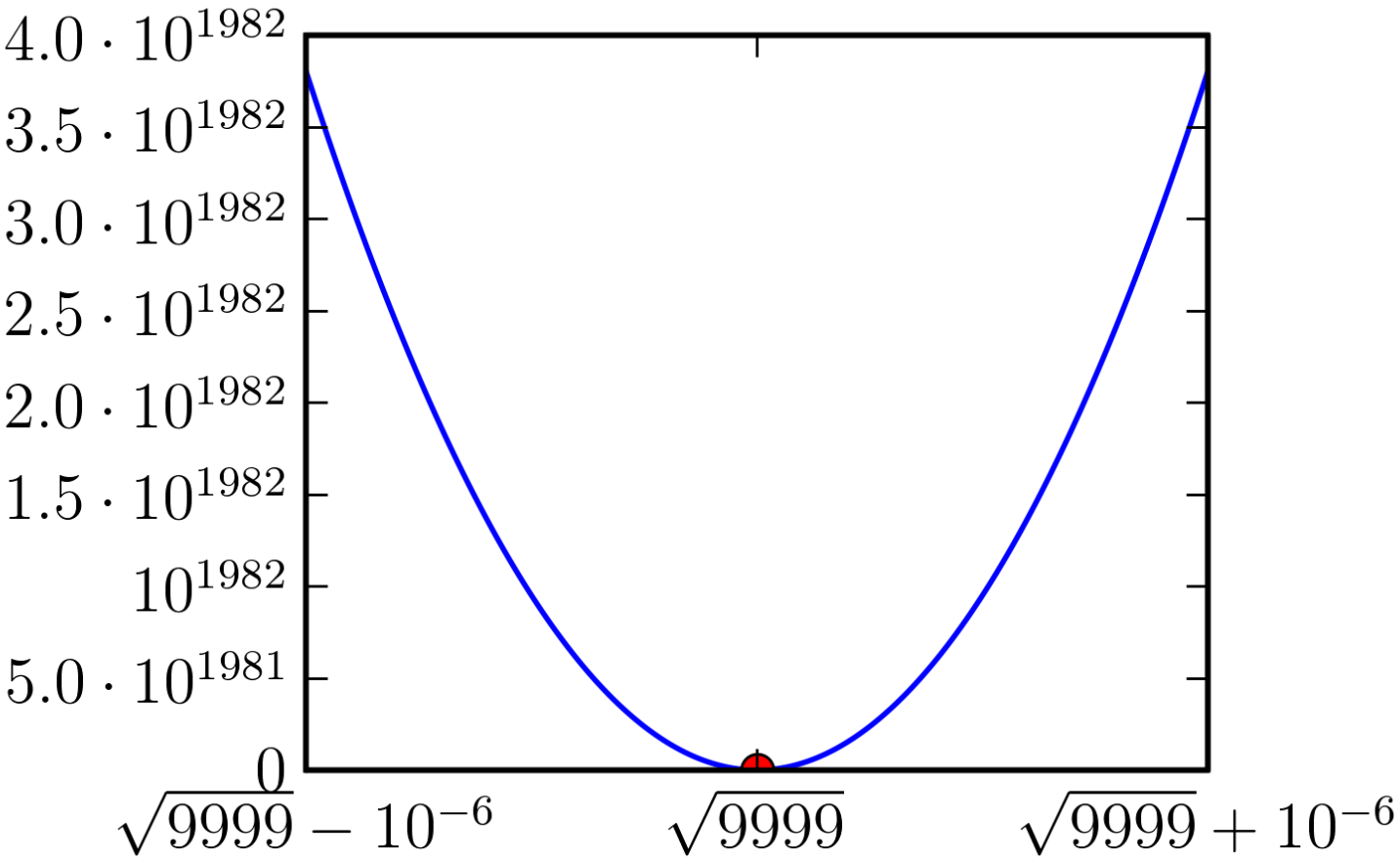
Zooming in on the roots

Let's take a closer look at those roots:



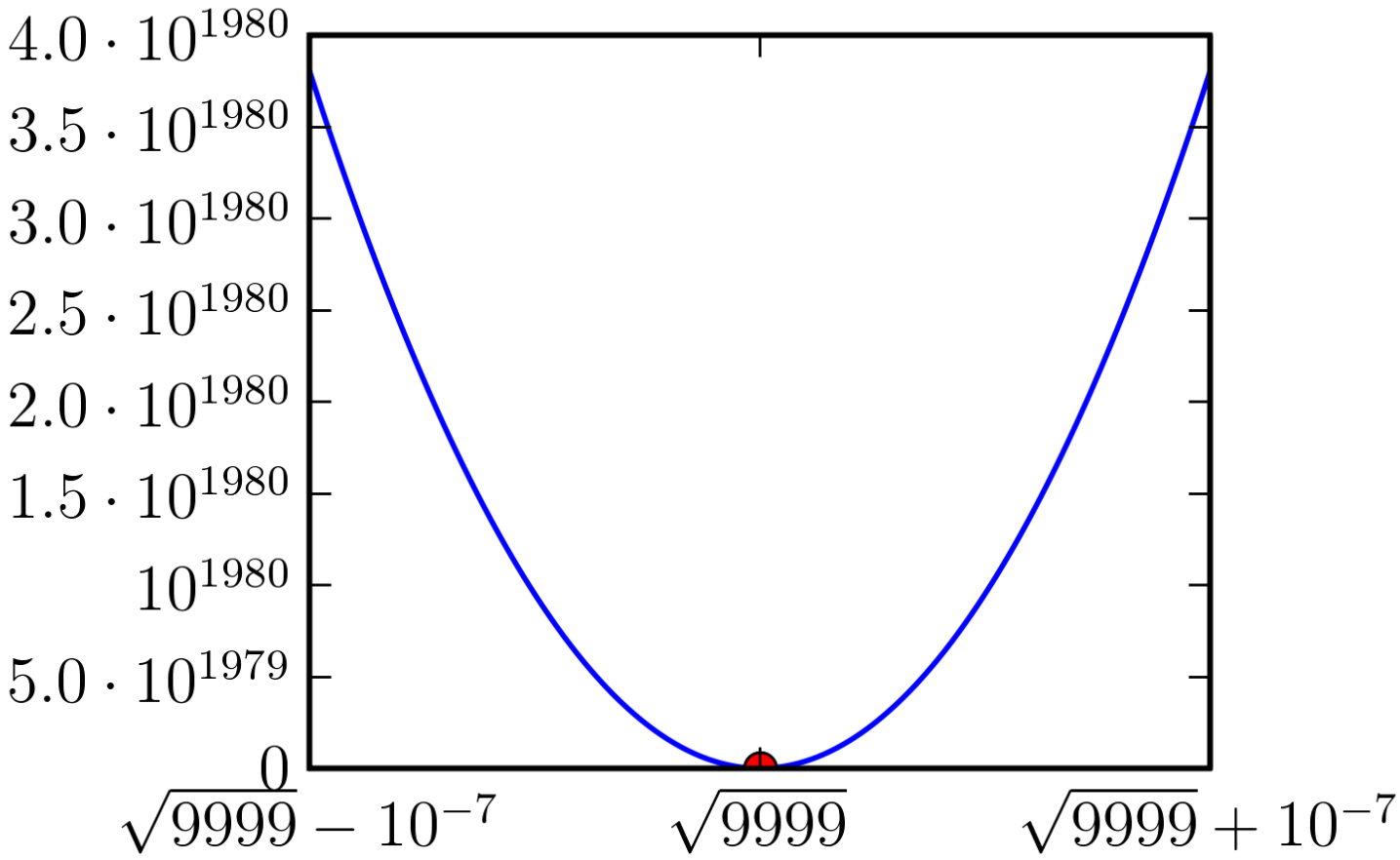
Zooming in on the roots

Let's take a closer look at those roots:



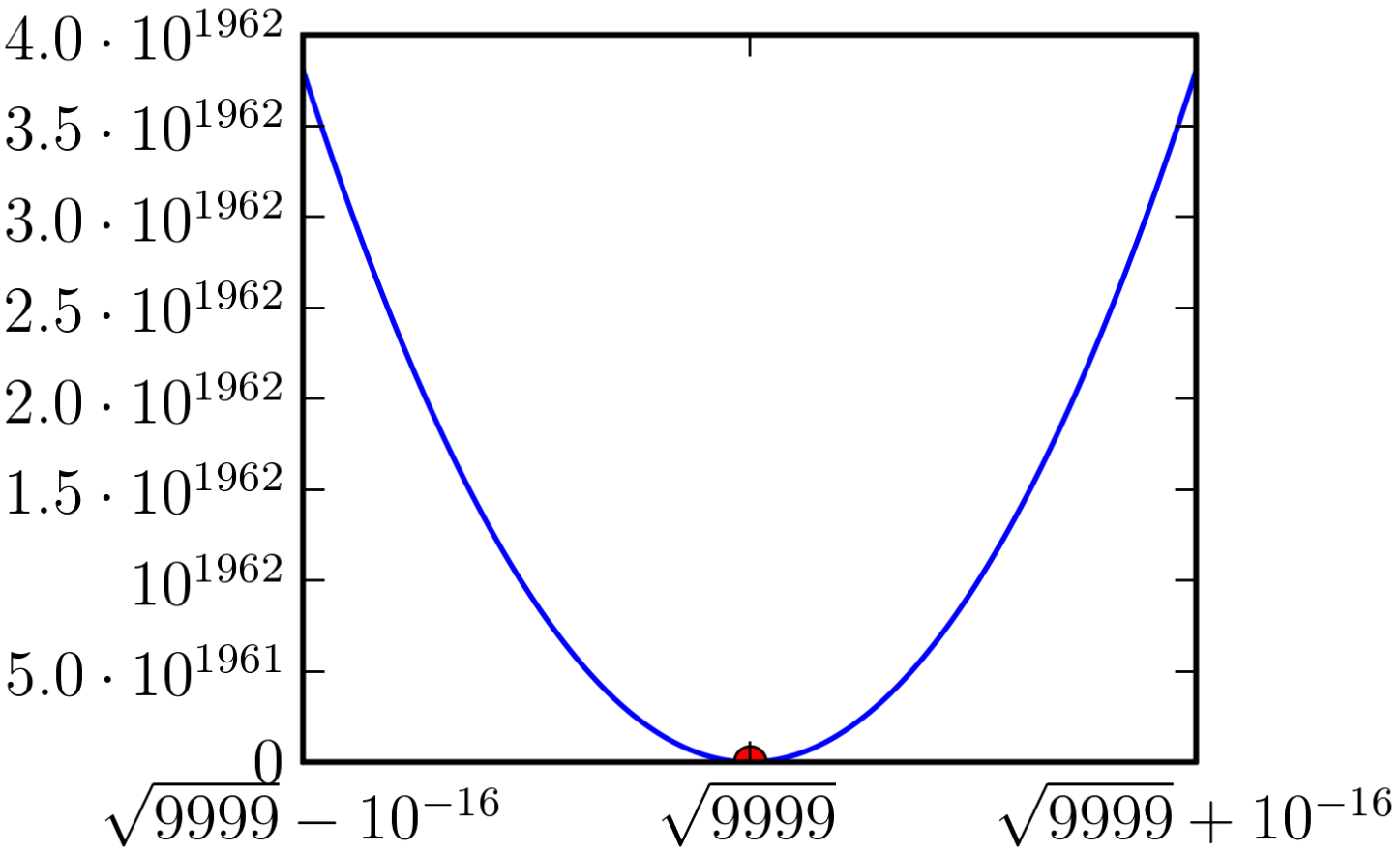
Zooming in on the roots

Let's take a closer look at those roots:



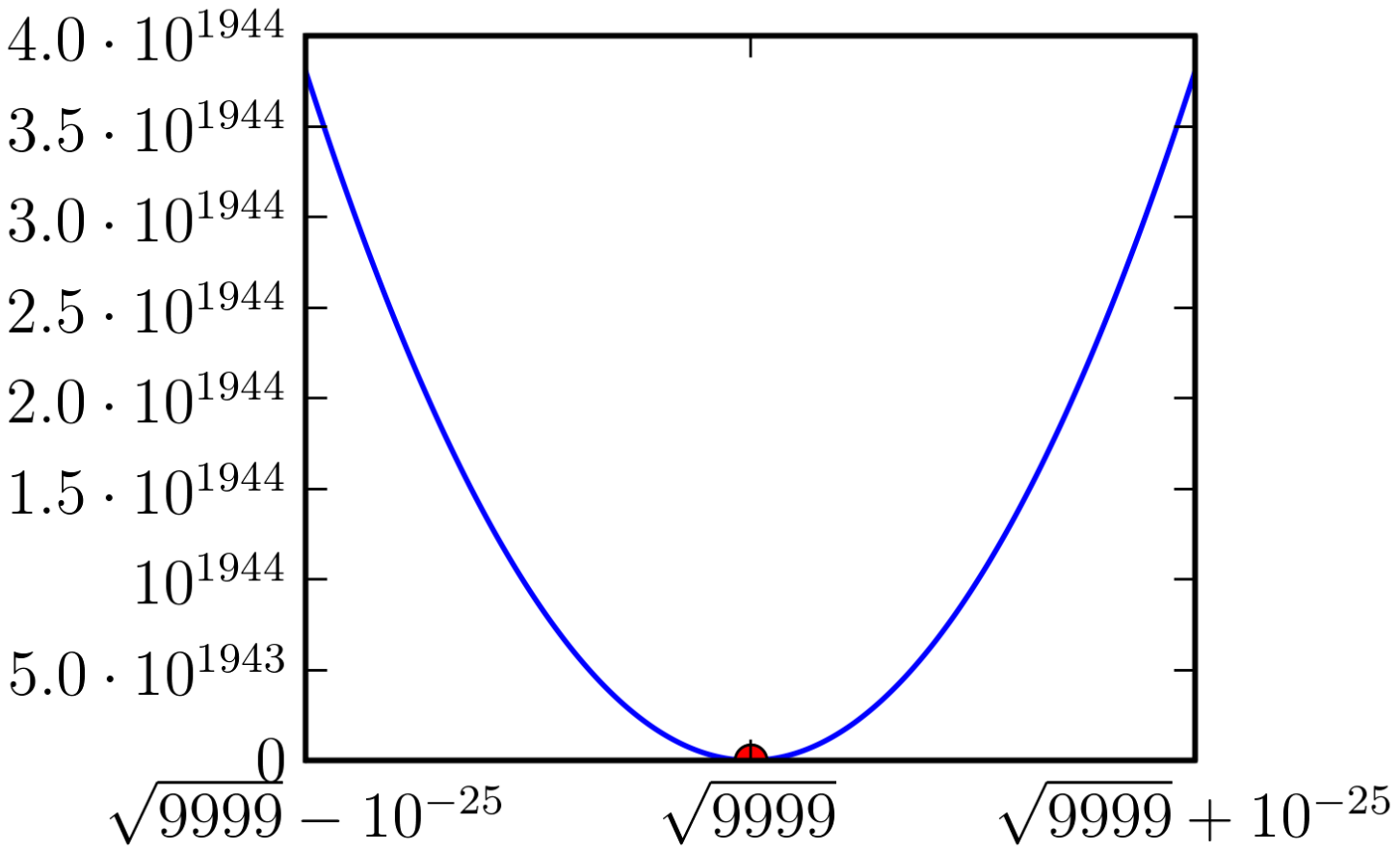
Zooming in on the roots

Let's take a closer look at those roots:



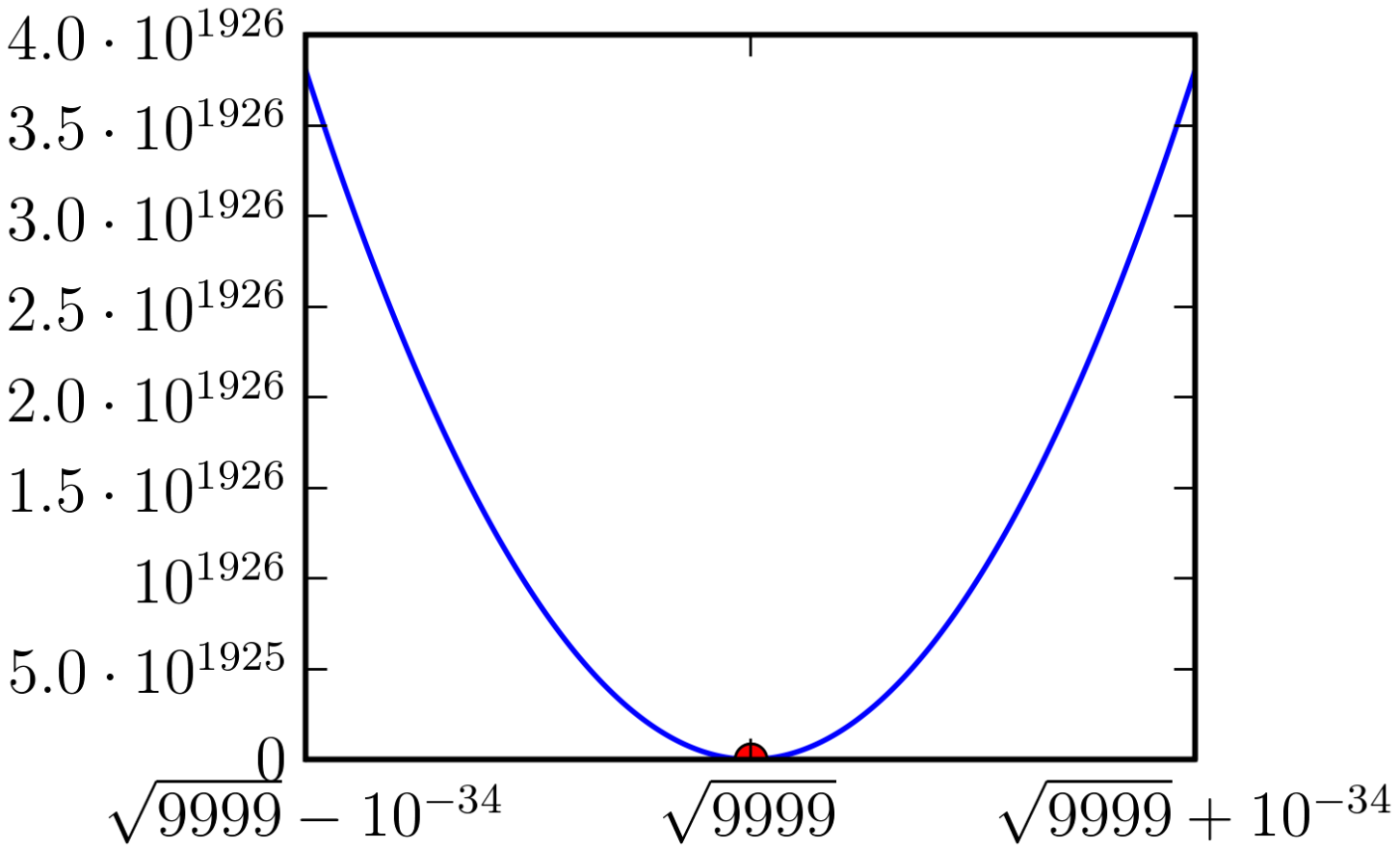
Zooming in on the roots

Let's take a closer look at those roots:



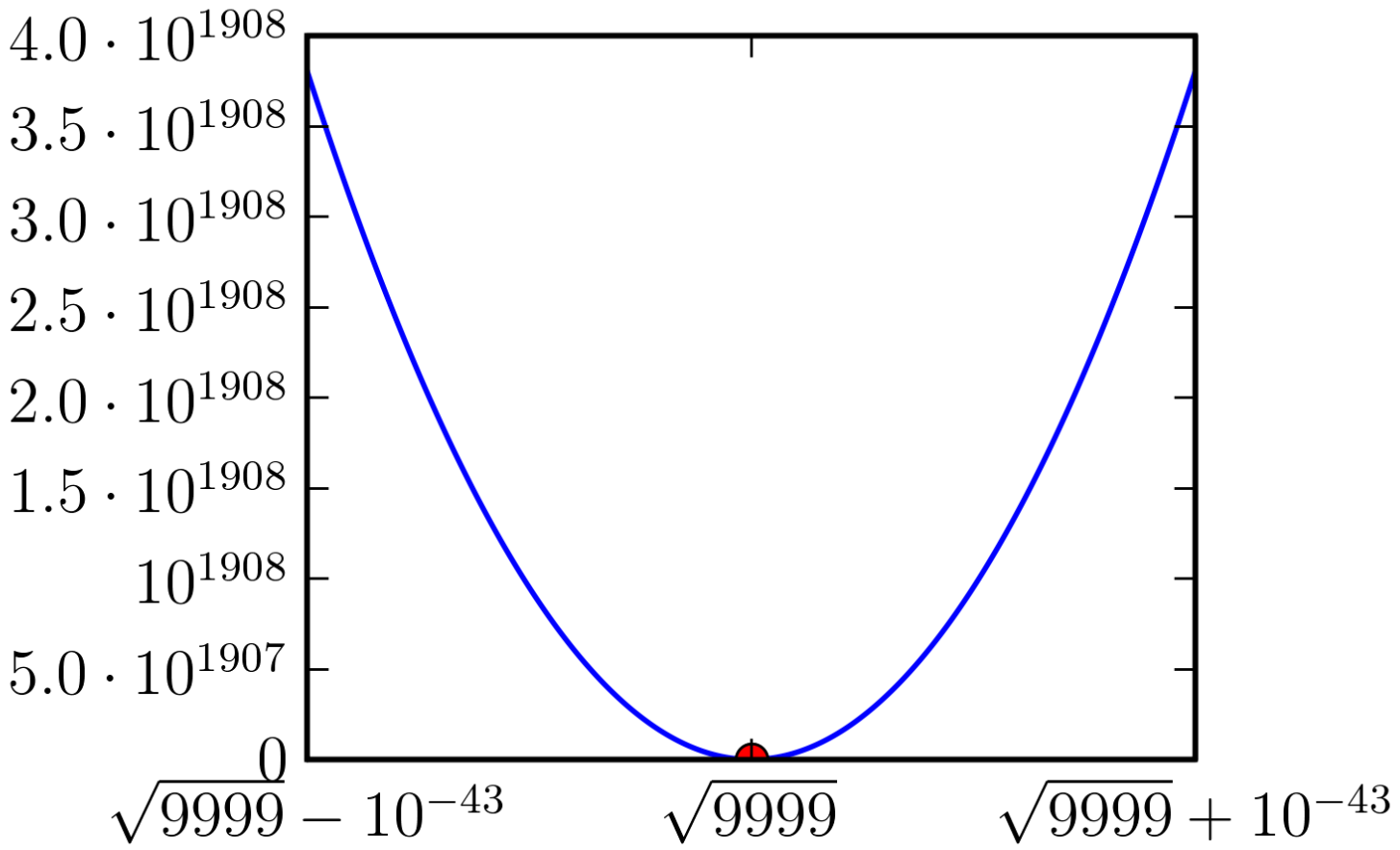
Zooming in on the roots

Let's take a closer look at those roots:



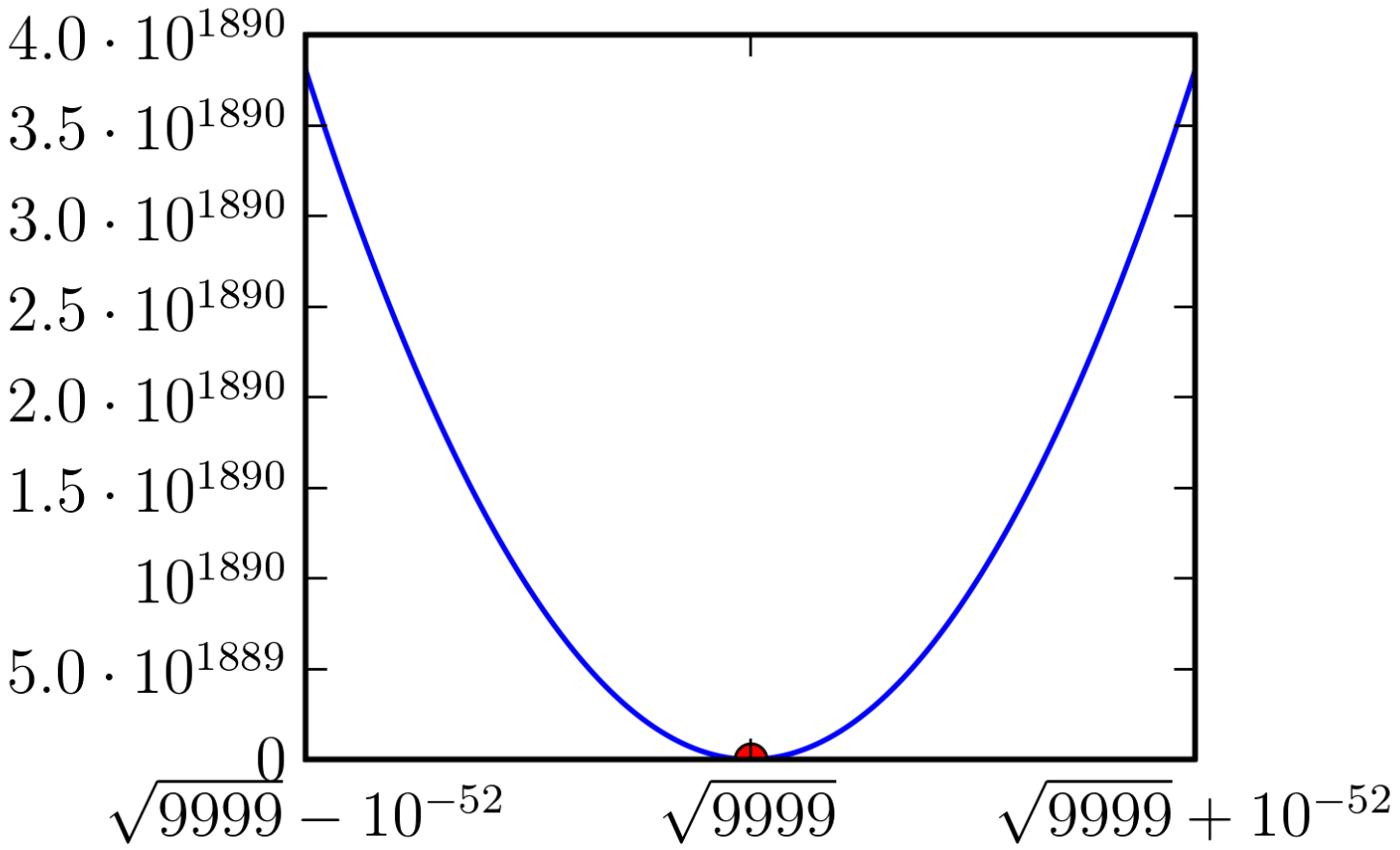
Zooming in on the roots

Let's take a closer look at those roots:



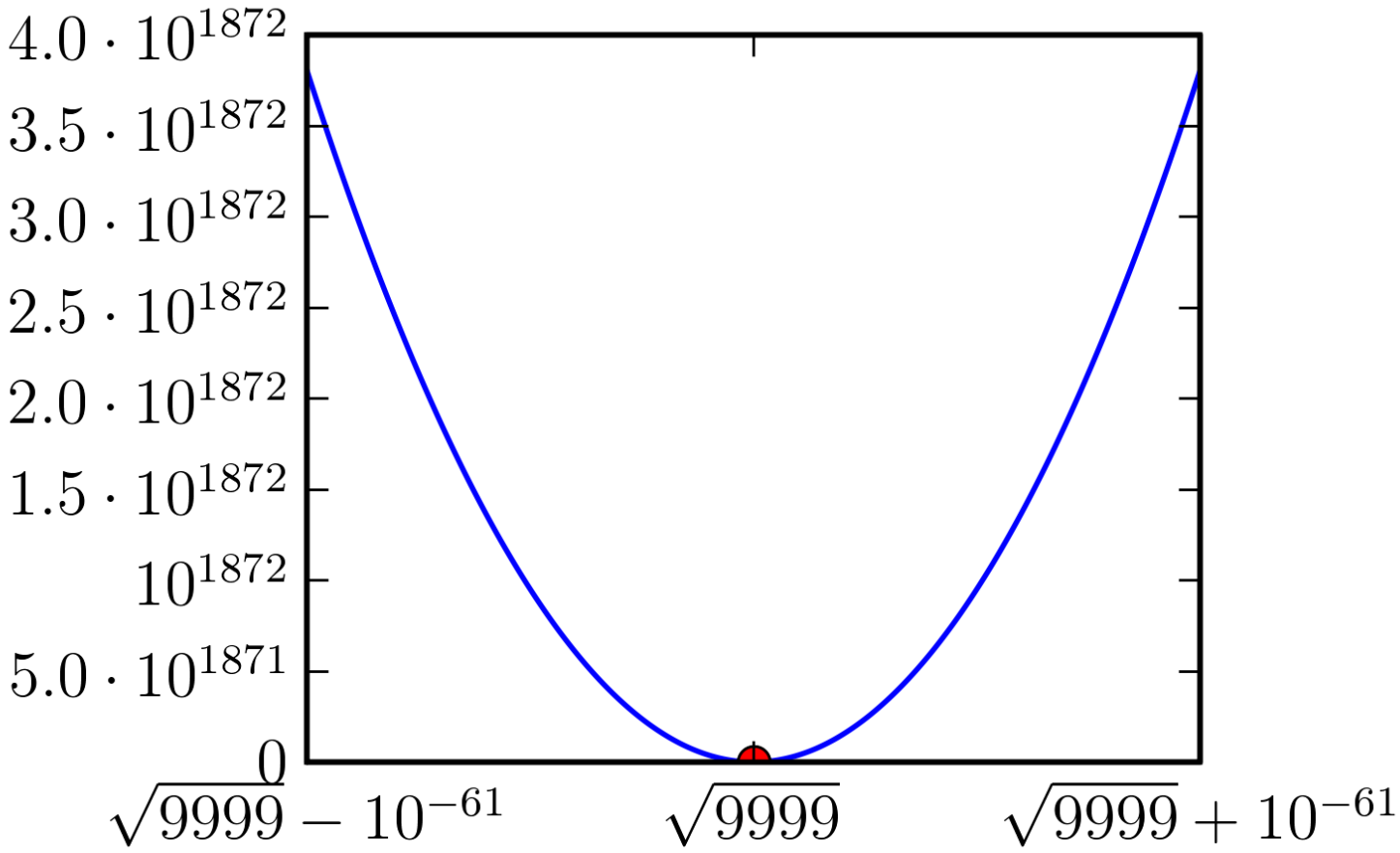
Zooming in on the roots

Let's take a closer look at those roots:



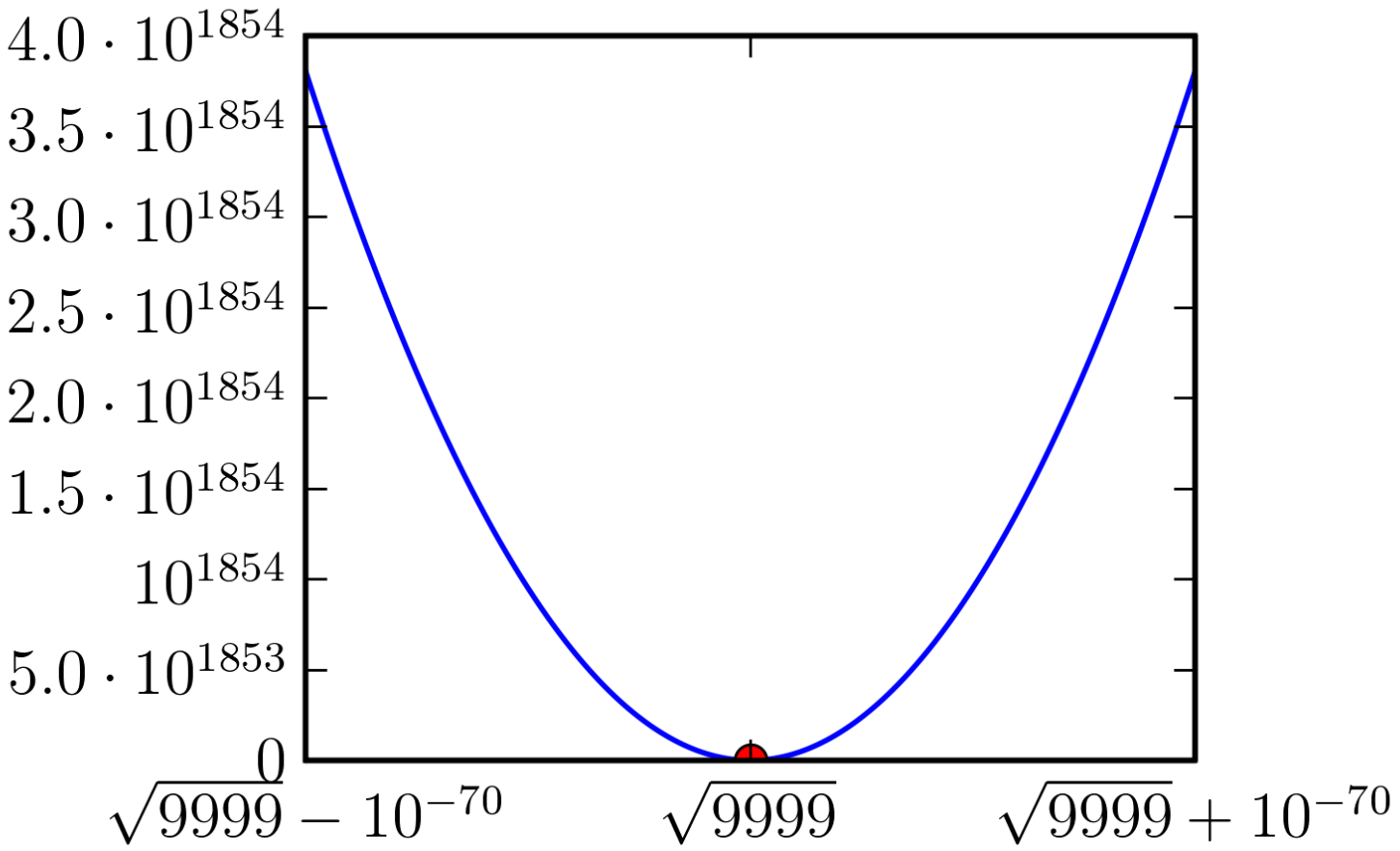
Zooming in on the roots

Let's take a closer look at those roots:



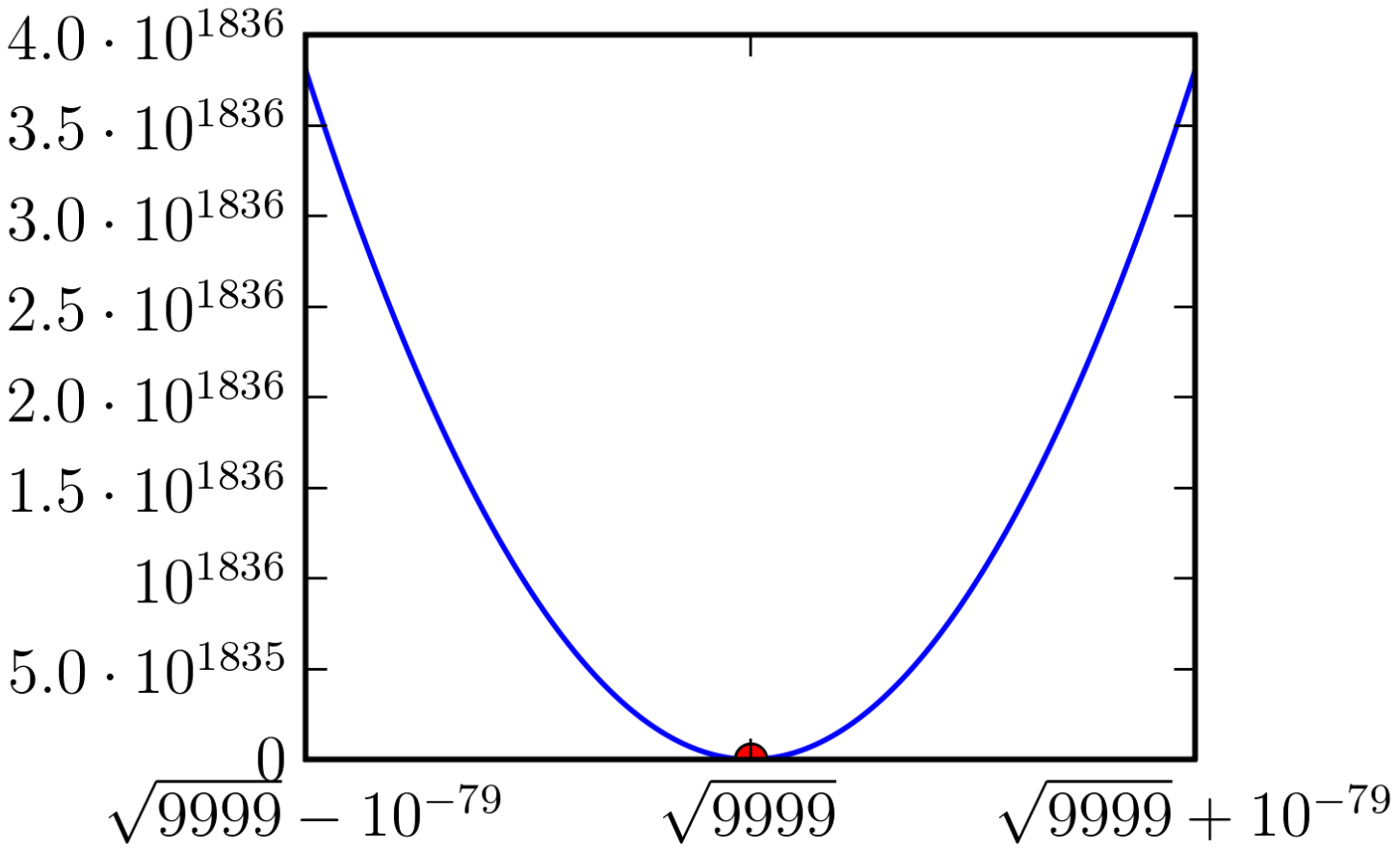
Zooming in on the roots

Let's take a closer look at those roots:



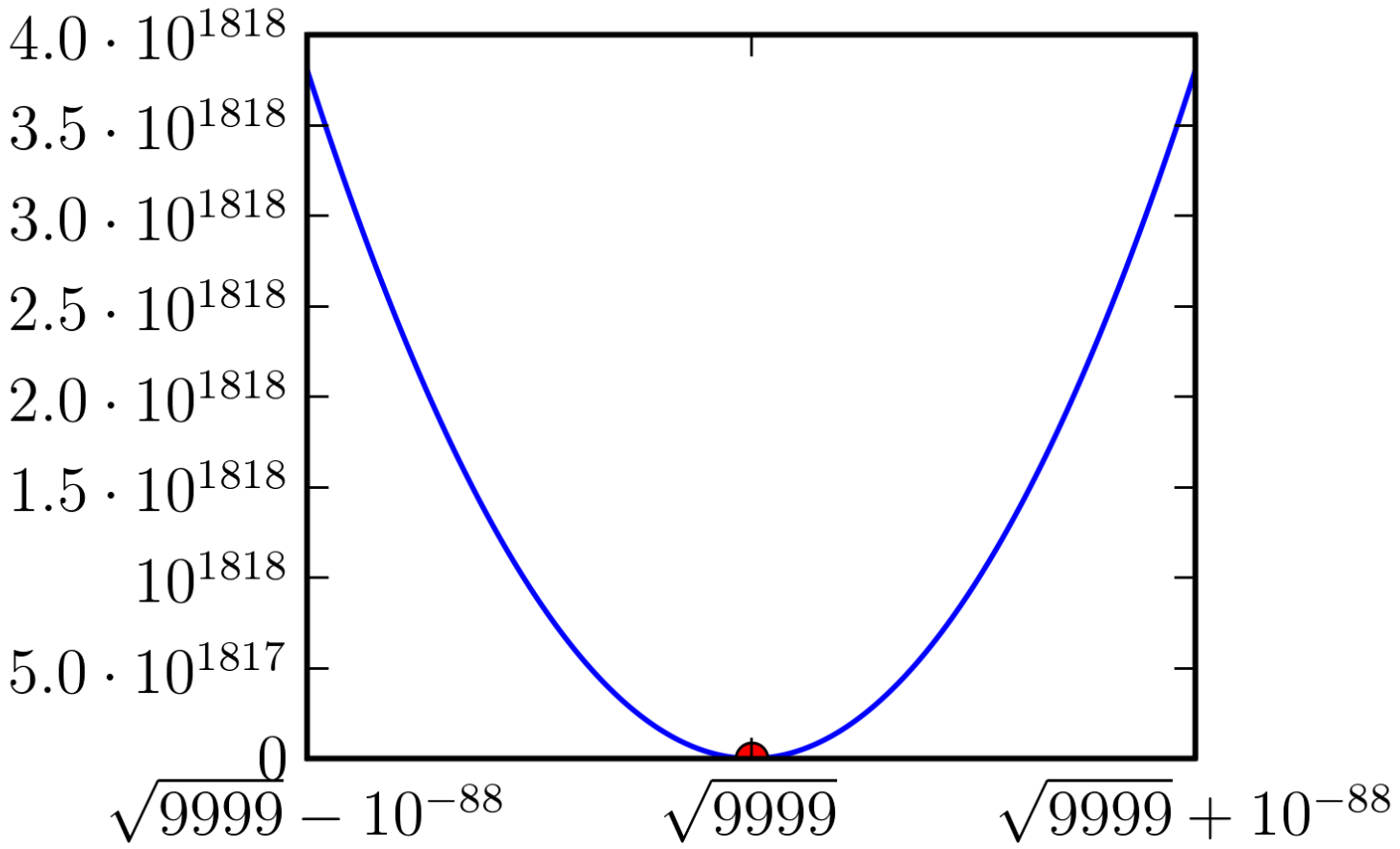
Zooming in on the roots

Let's take a closer look at those roots:



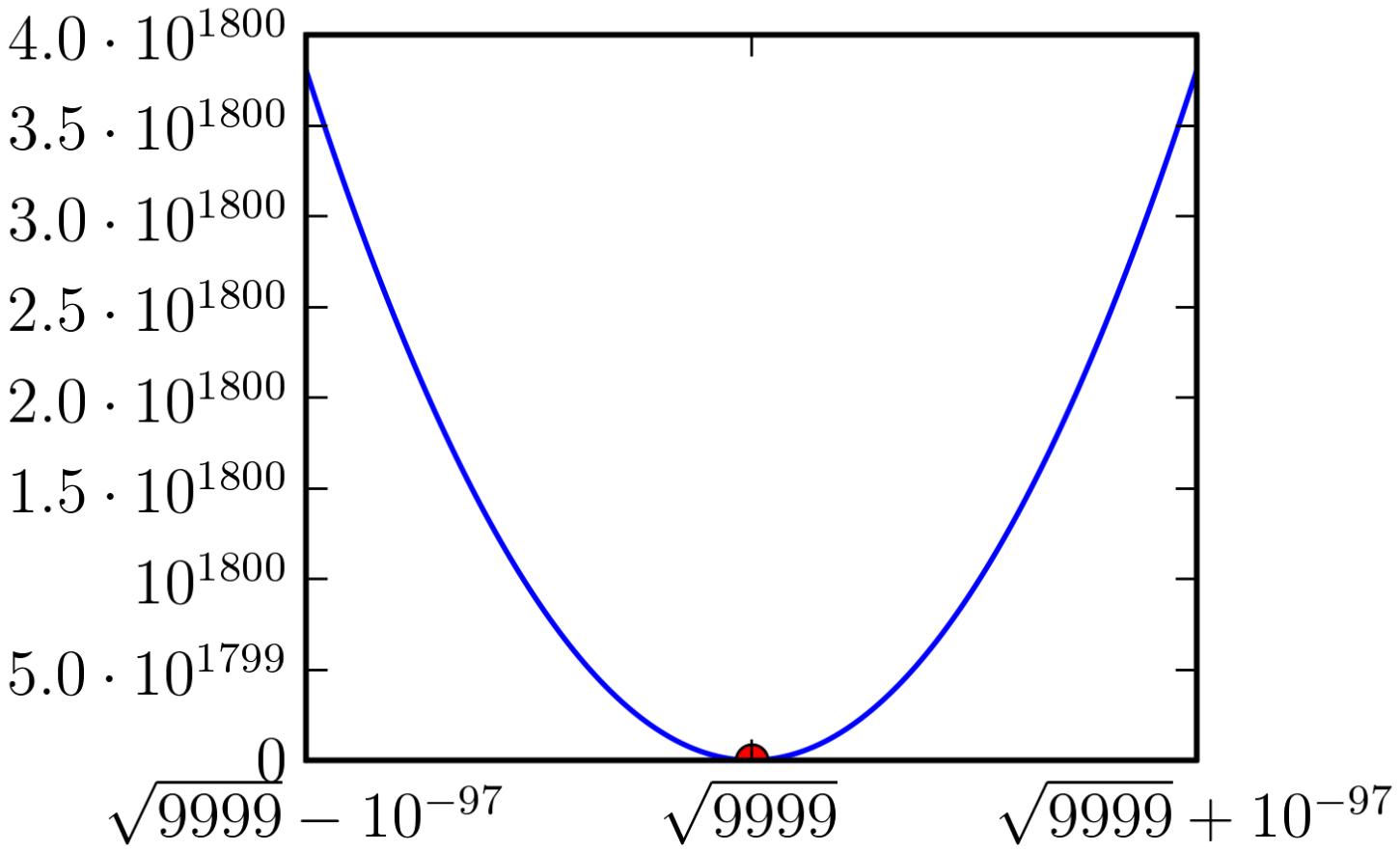
Zooming in on the roots

Let's take a closer look at those roots:



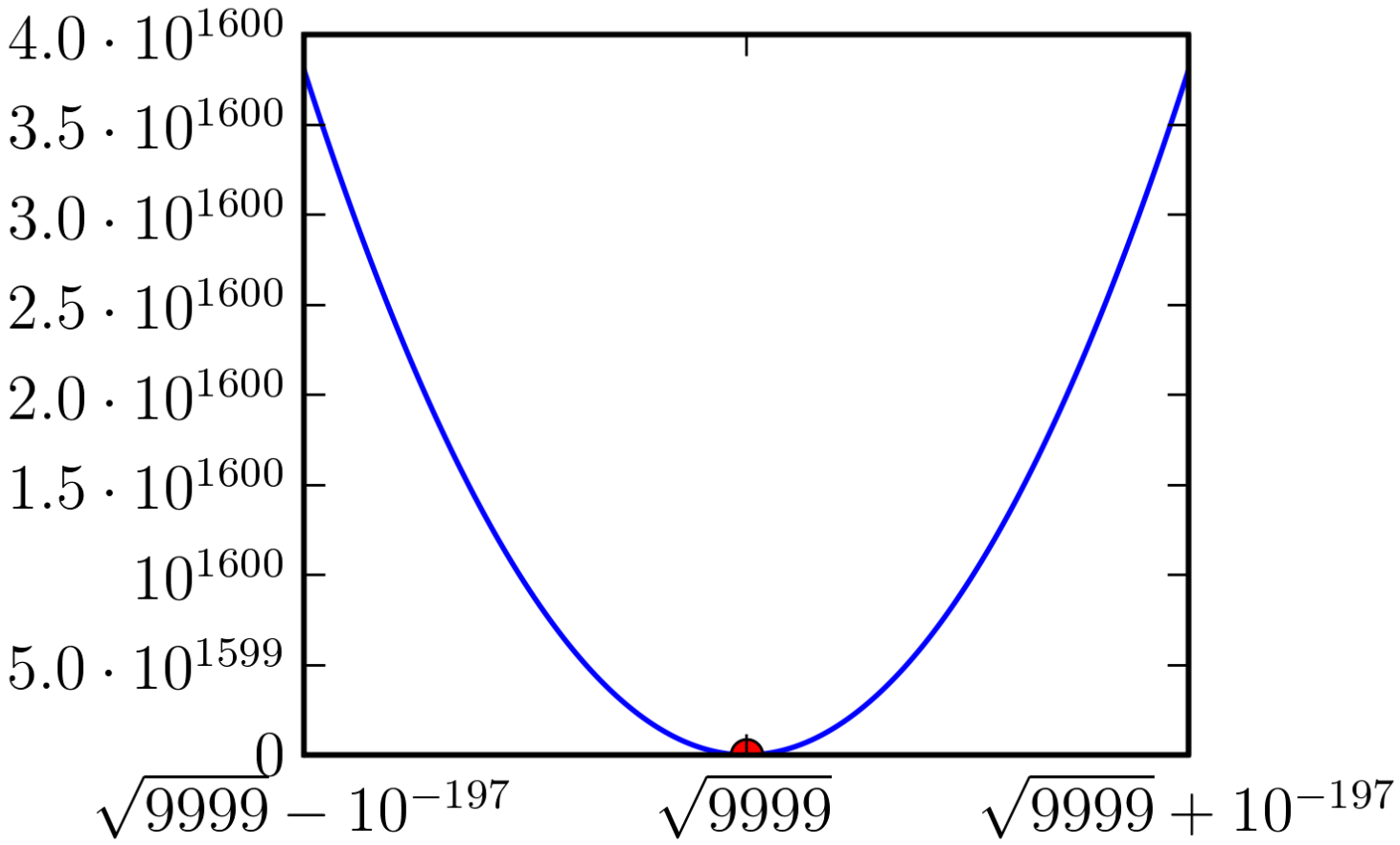
Zooming in on the roots

Let's take a closer look at those roots:



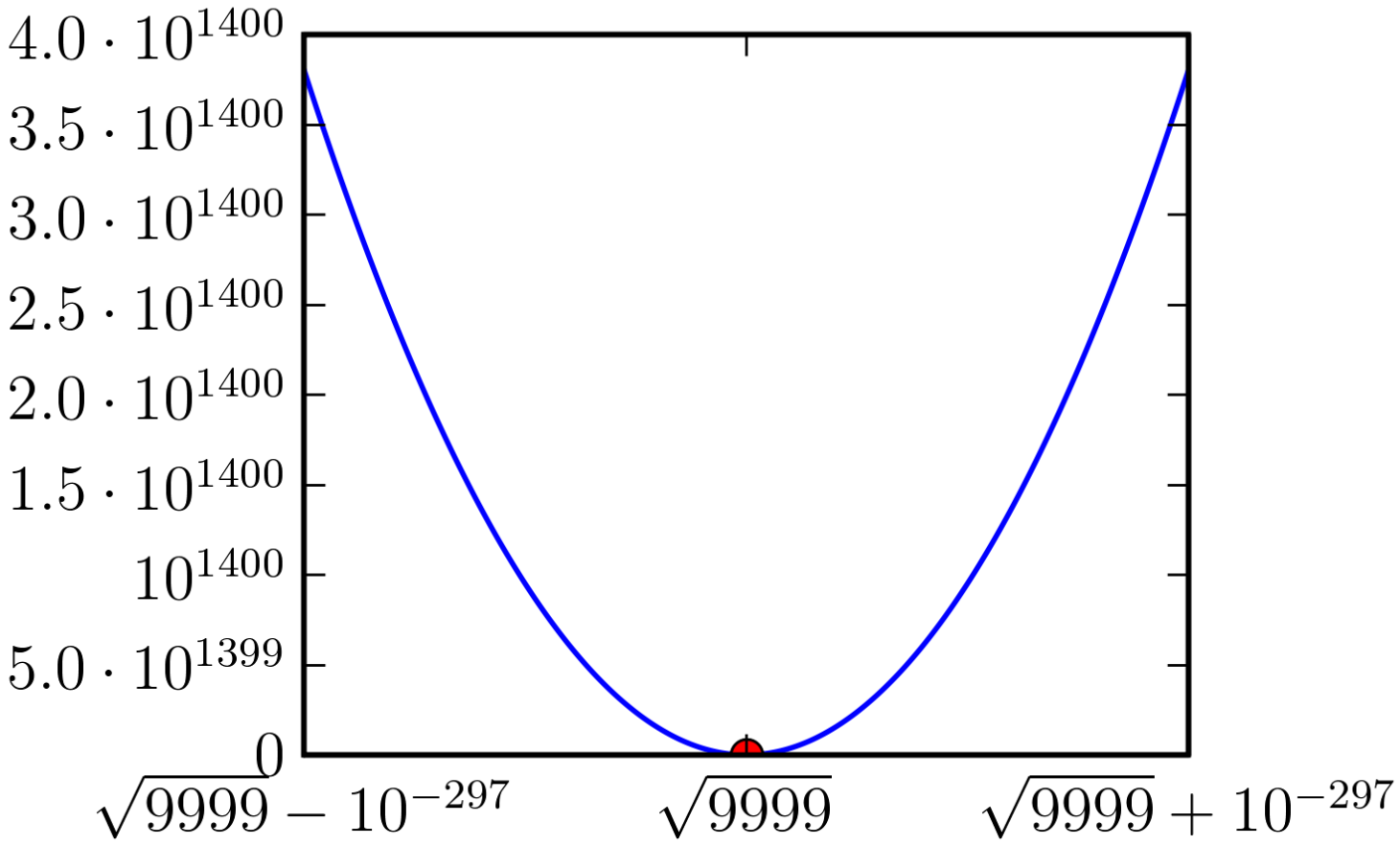
Zooming in on the roots

Let's take a closer look at those roots:



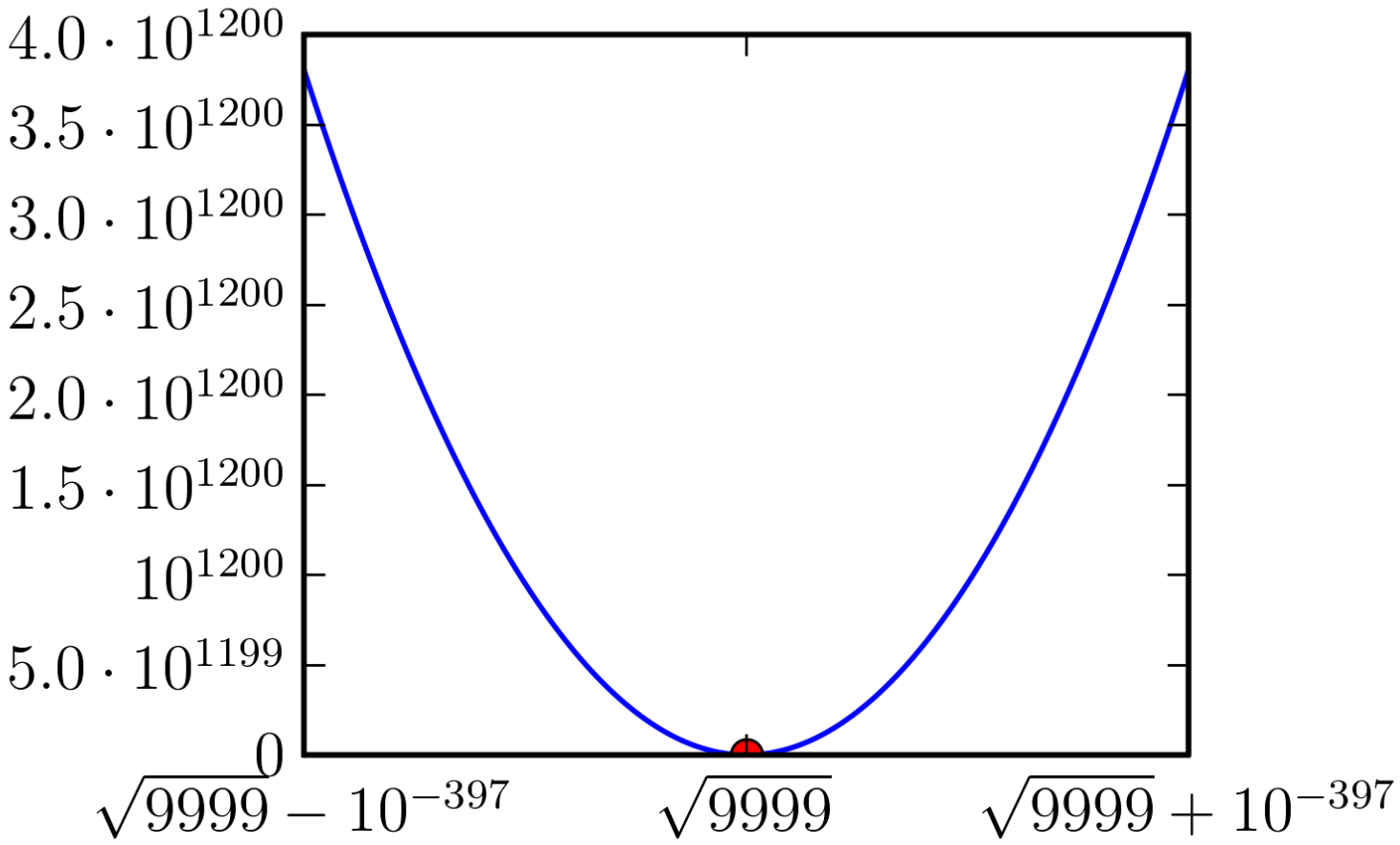
Zooming in on the roots

Let's take a closer look at those roots:



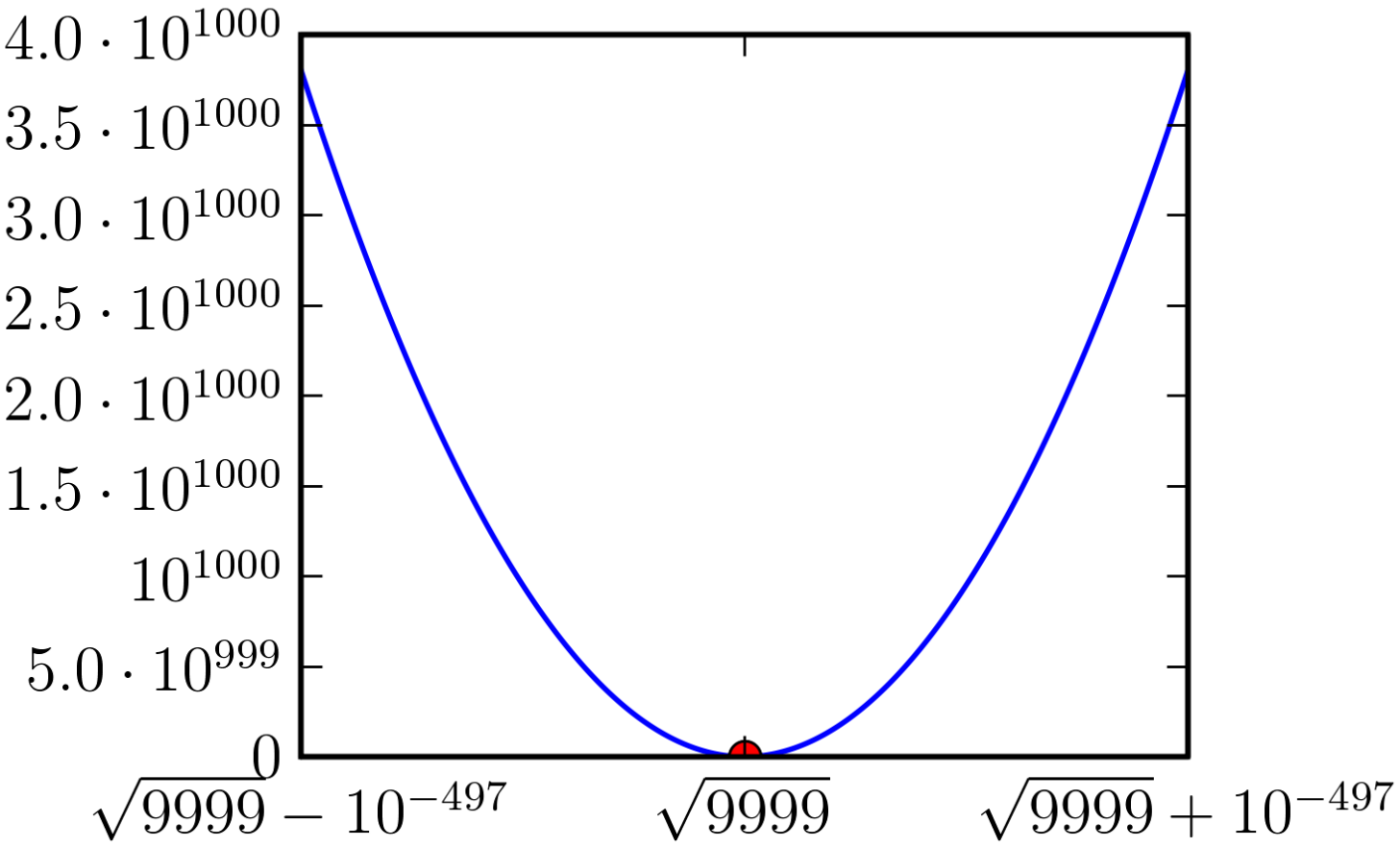
Zooming in on the roots

Let's take a closer look at those roots:



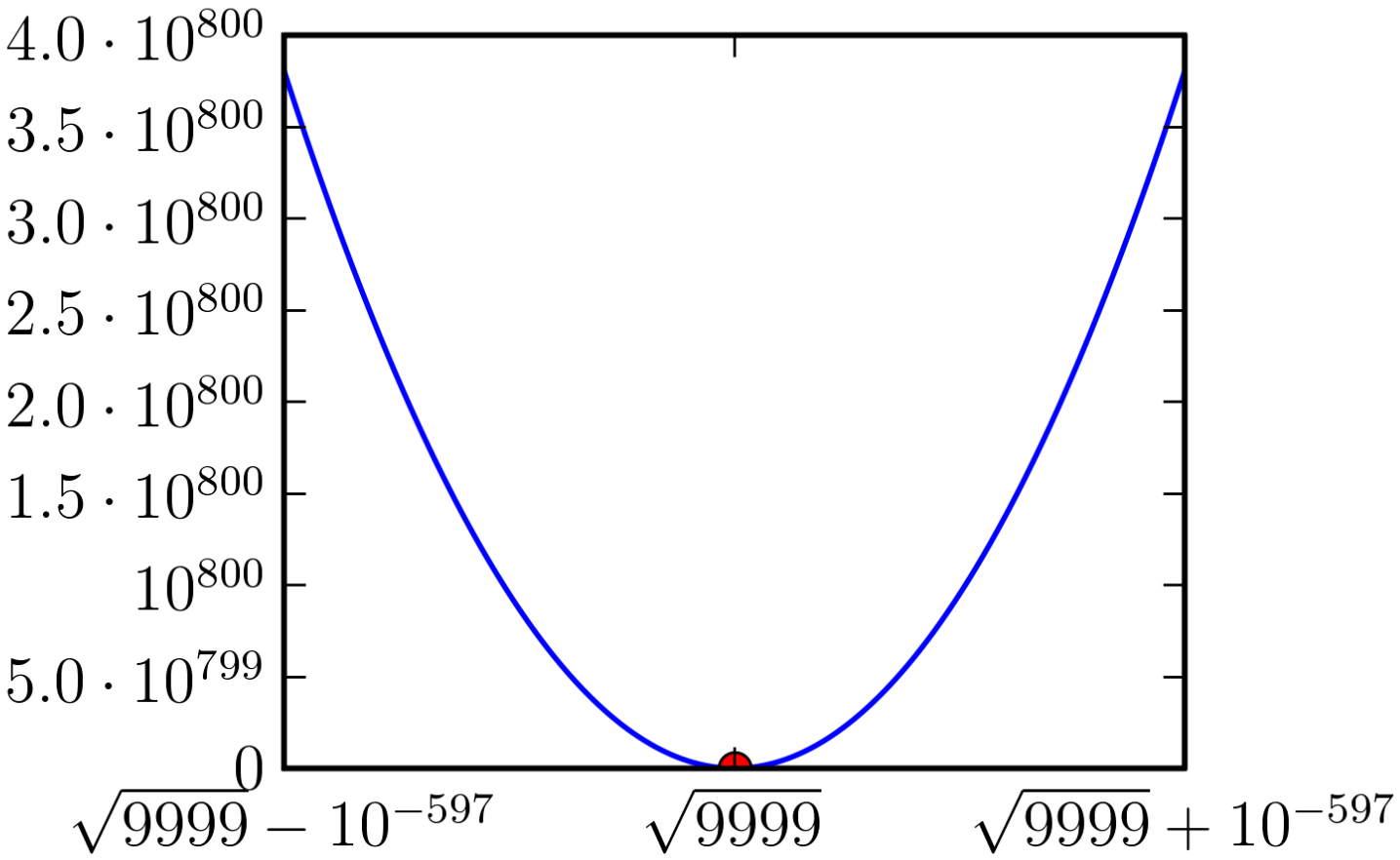
Zooming in on the roots

Let's take a closer look at those roots:



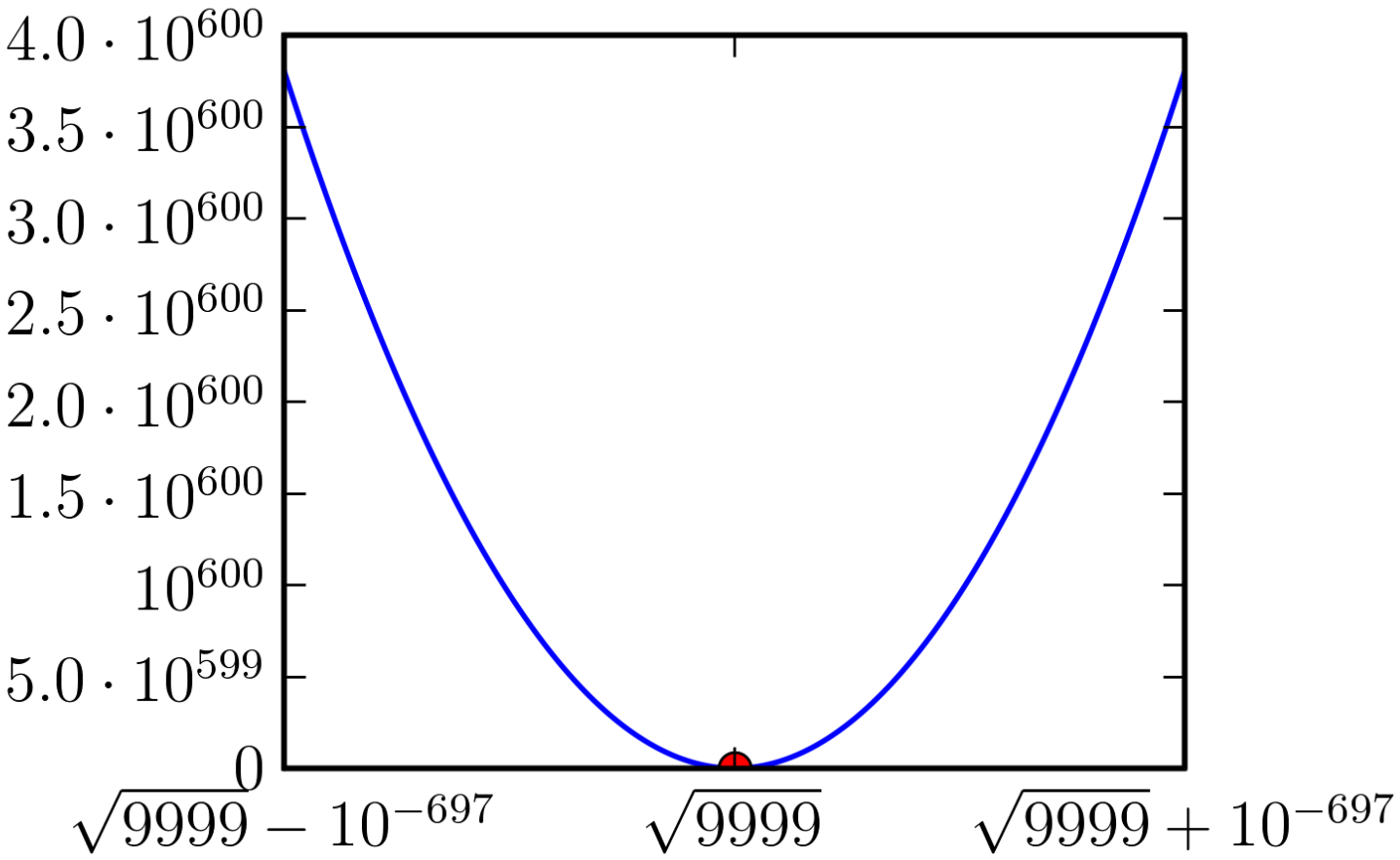
Zooming in on the roots

Let's take a closer look at those roots:



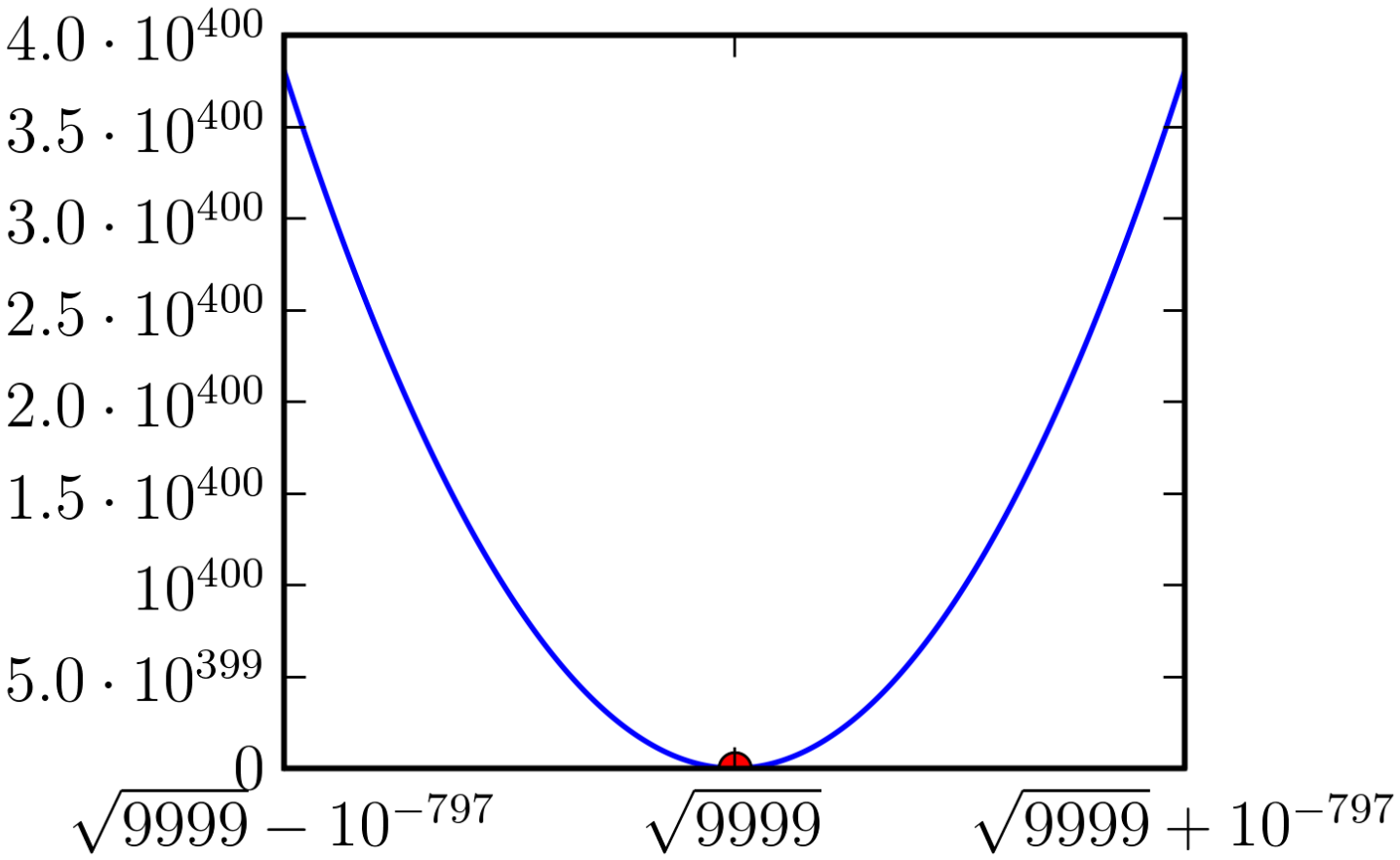
Zooming in on the roots

Let's take a closer look at those roots:



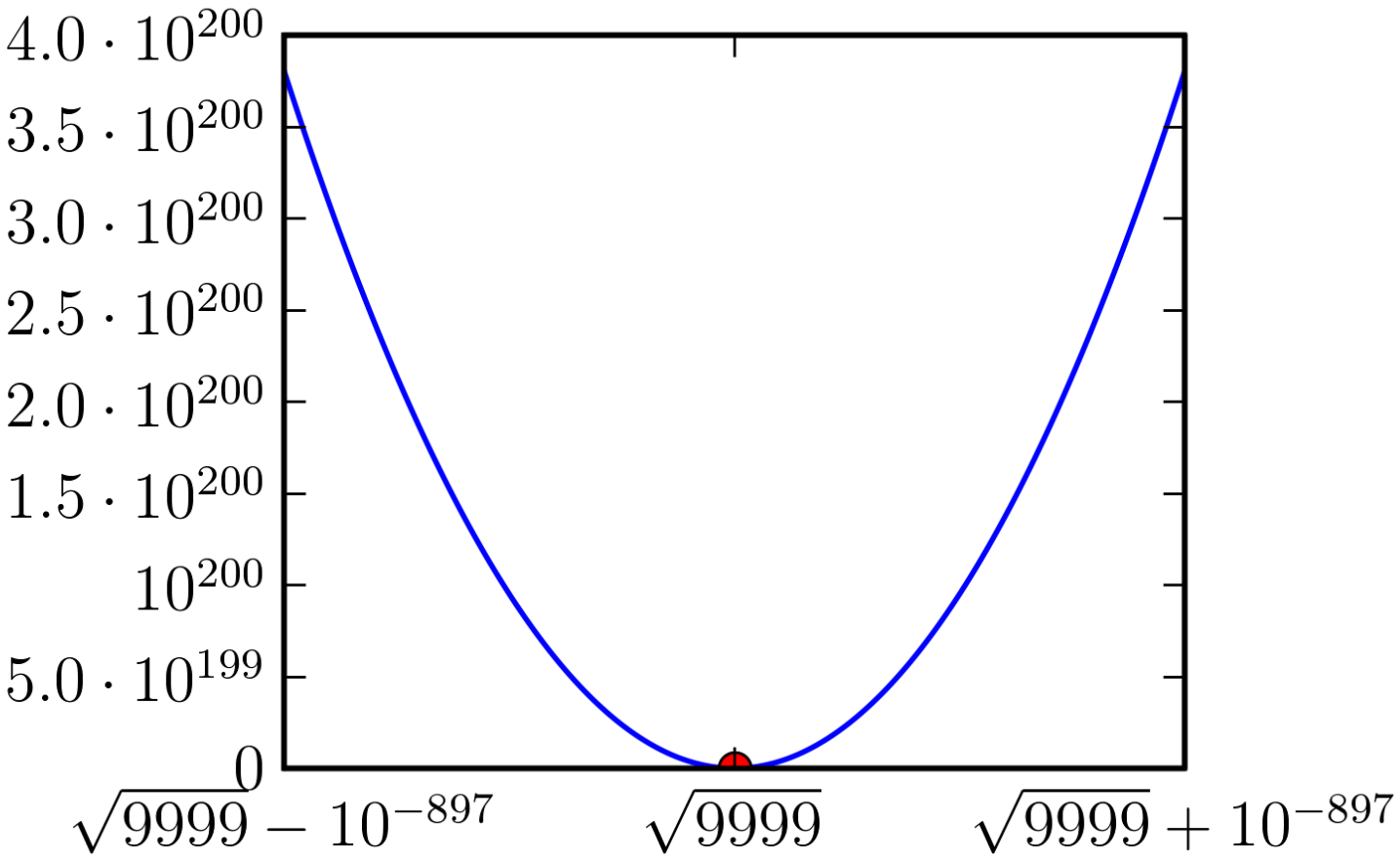
Zooming in on the roots

Let's take a closer look at those roots:



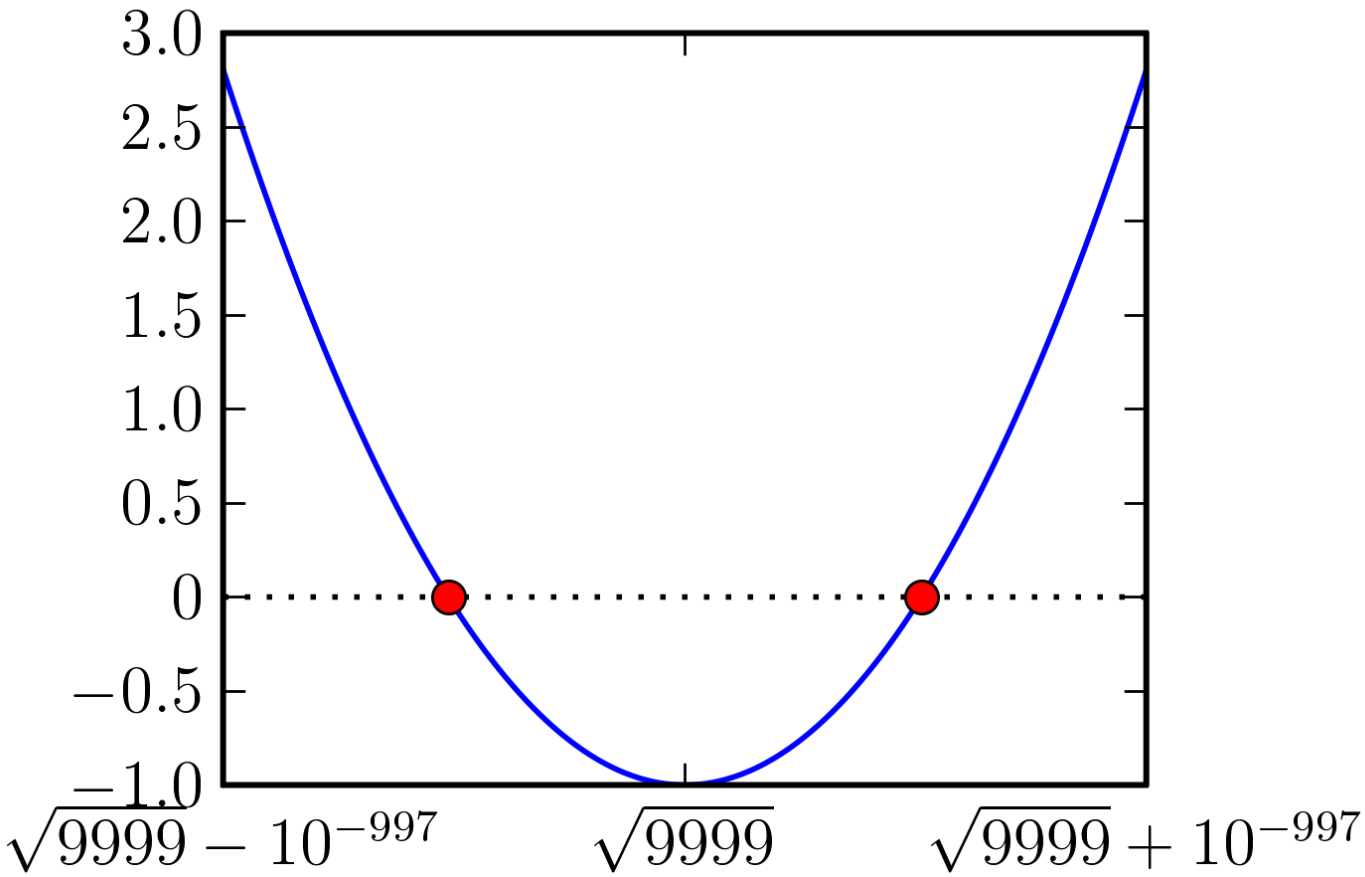
Zooming in on the roots

Let's take a closer look at those roots:



Zooming in on the roots

Let's take a closer look at those roots:



Yes, it's hard

- Our goal is to find a rational between these two roots. The simplest rational between the roots has a 502-digit numerator. My algorithm will actually find a rational whose denominator is a power of 2; the simplest such rational has a 1000-digit numerator. (I don't try to find the simplest such rational; my algorithm finds a rational with a 2018-digit numerator.)
- Most of the algorithms in the literature will compute the numerator of the exact value of the polynomial at the separating rational they find. This is a number with more than $996,000$ digits.

Yes, it's hard

- Our goal is to find a rational between these two roots. The simplest rational between the roots has a 502-digit numerator. My algorithm will actually find a rational whose denominator is a power of 2; the simplest such rational has a 1000-digit numerator. (I don't try to find the simplest such rational; my algorithm finds a rational with a 2018-digit numerator.)
- Most of the algorithms in the literature will compute the numerator of the exact value of the polynomial at the separating rational they find. This is a number with more than $996,000$ digits.

Subdivision algorithms

- Most algorithms operate as follows:
  - Find a segment of the real numbers guaranteed to contain all the real roots.
  - Try to verify that the segment contains either zero or one real roots.
  - If this fails—if the segment might contain more than one real root—then divide the segment in two pieces and recursively find the real roots in each piece.
- This leads to a conceptual binary tree structure. With the algorithms I know about, the difficult polynomials are the ones with a deep tree.
- A polynomial with many roots will have a wide tree—one with many leaves. Such a tree cannot be extremely shallow; a binary tree with 1024 leaves must have average depth at least 10.
- On this challenge polynomial, an algorithm that does strict bisection will have a tree depth greater than 3300.

Subdivision algorithms

- Most algorithms operate as follows:
  - Find a segment of the real numbers guaranteed to contain all the real roots.
  - Try to verify that the segment contains either zero or one real roots.
  - If this fails—if the segment might contain more than one real root—then divide the segment in two pieces and recursively find the real roots in each piece.
- This leads to a conceptual binary tree structure. With the algorithms I know about, the difficult polynomials are the ones with a deep tree.
- A polynomial with many roots will have a wide tree—one with many leaves. Such a tree cannot be extremely shallow; a binary tree with 1024 leaves must have average depth at least 10.
- On this challenge polynomial, an algorithm that does strict bisection will have a tree depth greater than 3300.

Subdivision algorithms

- Most algorithms operate as follows:
  - Find a segment of the real numbers guaranteed to contain all the real roots.
  - Try to verify that the segment contains either zero or one real roots.
  - If this fails—if the segment might contain more than one real root—then divide the segment in two pieces and recursively find the real roots in each piece.
- This leads to a conceptual binary tree structure. With the algorithms I know about, the difficult polynomials are the ones with a deep tree.
- A polynomial with many roots will have a wide tree—one with many leaves. Such a tree cannot be extremely shallow; a binary tree with 1024 leaves must have average depth at least 10.
- On this challenge polynomial, an algorithm that does strict bisection will have a tree depth greater than 3300.

Subdivision algorithms

- Most algorithms operate as follows:
  - Find a segment of the real numbers guaranteed to contain all the real roots.
  - Try to verify that the segment contains either zero or one real roots.
  - If this fails—if the segment might contain more than one real root—then divide the segment in two pieces and recursively find the real roots in each piece.
- This leads to a conceptual binary tree structure. With the algorithms I know about, the difficult polynomials are the ones with a deep tree.
- A polynomial with many roots will have a wide tree—one with many leaves. Such a tree cannot be extremely shallow; a binary tree with 1024 leaves must have average depth at least 10.
- On this challenge polynomial, an algorithm that does strict bisection will have a tree depth greater than 3300.

Outline

Outline

Properties of Möbius transformations

Definition

A Möbius transformation is a function of the form

$$x \mapsto \frac{ax+b}{cx+d}$$

where $ad \neq bc$.

- Given two segments of the real number line, there is a Möbius transformation that will map one segment onto the other. (For instance, there is a Möbius transformation that will map $(1/7, 2/3)$ onto $(0, \infty)$.)
- Given a Möbius transformation f and a polynomial with roots $r_1, r_2, \ldots, r_k$, you can construct a new polynomial with roots $f(r_1), f(r_2), \ldots, f(r_k)$ (just based on the coefficients of the polynomial, and without knowing the roots).

Properties of Möbius transformations

Definition

A Möbius transformation is a function of the form

$$x \mapsto \frac{ax+b}{cx+d}$$

where $ad \neq bc$.

- Given two segments of the real number line, there is a Möbius transformation that will map one segment onto the other. (For instance, there is a Möbius transformation that will map $(1/7, 2/3)$ onto $(0, \infty)$.)
- Given a Möbius transformation f and a polynomial with roots $r_1, r_2, \ldots, r_k$, you can construct a new polynomial with roots $f(r_1), f(r_2), \ldots, f(r_k)$ (just based on the coefficients of the polynomial, and without knowing the roots).

Properties of Möbius transformations

Definition

A Möbius transformation is a function of the form

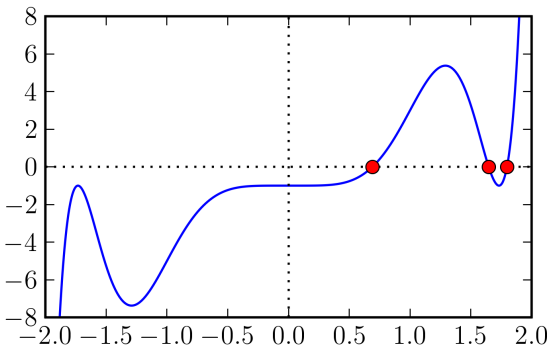
$$x \mapsto \frac{ax+b}{cx+d}$$

where $ad \neq bc$.

- Given two segments of the real number line, there is a Möbius transformation that will map one segment onto the other. (For instance, there is a Möbius transformation that will map $(1/7, 2/3)$ onto $(0, \infty)$.)
- Given a Möbius transformation f and a polynomial with roots $r_1, r_2, \ldots, r_k$, you can construct a new polynomial with roots $f(r_1), f(r_2), \ldots, f(r_k)$ (just based on the coefficients of the polynomial, and without knowing the roots).

Outline

Estimating the number of positive real roots



- Recall our polynomial $x^5(x^2-3)^2-1$. Expanded out, this is:

$$+x^9-6x^7+9x^5-1$$

- Sign variations: 3. Positive real roots: 3.
- We can map $x \mapsto -x$ to get:

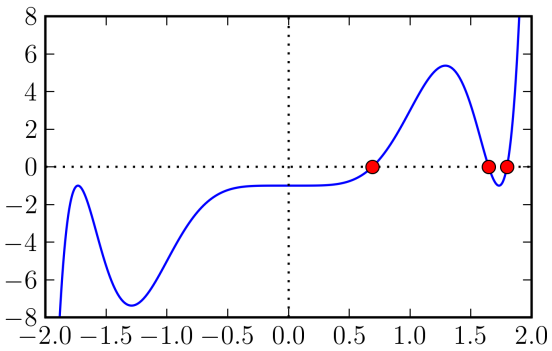
$$-x^9+6x^7-9x^5-1$$

- Sign variations: 2. Negative real roots: 0.

Theorem

Descartes' Rule of Signs: The number of positive real roots of a polynomial is $\leq$ the number of sign transitions. The difference is even.

Estimating the number of positive real roots



- Recall our polynomial $x^5(x^2-3)^2-1$. Expanded out, this is:

$$+x^9-6x^7+9x^5-1$$

- Sign variations: 3. Positive real roots: 3.
- We can map $x \mapsto -x$ to get:

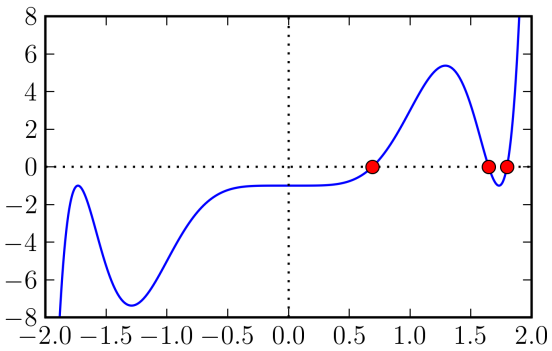
$$-x^9+6x^7-9x^5-1$$

- Sign variations: 2. Negative real roots: 0.

Theorem

Descartes' Rule of Signs: The number of positive real roots of a polynomial is $\leq$ the number of sign transitions. The difference is even.

Estimating the number of positive real roots



- Recall our polynomial $x^5(x^2-3)^2-1$. Expanded out, this is:

$$+x^9-6x^7+9x^5-1$$

- Sign variations: 3. Positive real roots: 3.
- We can map $x \mapsto -x$ to get:

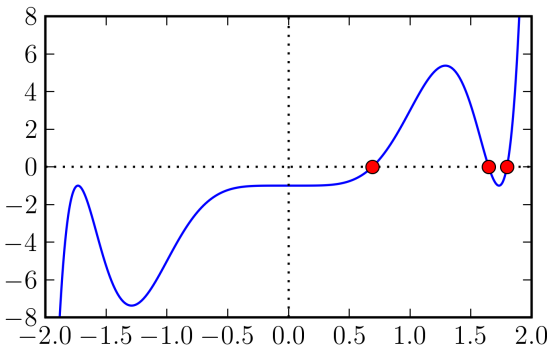
$$-x^9+6x^7-9x^5-1$$

- Sign variations: 2. Negative real roots: 0.

Theorem

Descartes' Rule of Signs: The number of positive real roots of a polynomial is $\leq$ the number of sign transitions. The difference is even.

Estimating the number of positive real roots



- Recall our polynomial $x^5(x^2-3)^2-1$. Expanded out, this is:

$$+x^9-6x^7+9x^5-1$$

- Sign variations: 3. Positive real roots: 3.
- We can map $x \mapsto -x$ to get:

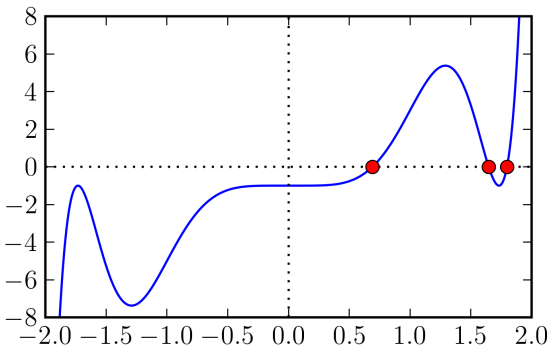
$$-x^9+6x^7-9x^5-1$$

- Sign variations: 2. Negative real roots: 0.

Theorem

Descartes' Rule of Signs: The number of positive real roots of a polynomial is $\leq$ the number of sign transitions. The difference is even.

Estimating the number of positive real roots



- Recall our polynomial $x^5(x^2-3)^2-1$. Expanded out, this is:

$$+x^9-6x^7+9x^5-1$$

- Sign variations: 3. Positive real roots: 3.
- We can map $x \mapsto -x$ to get:

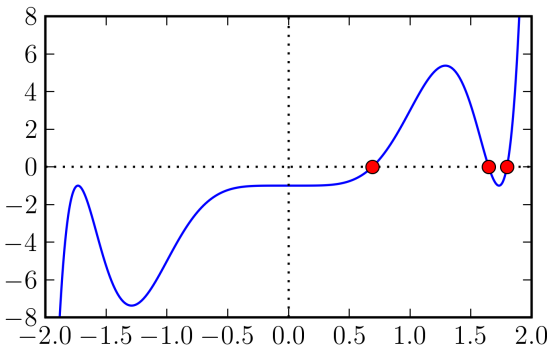
$$-x^9+6x^7-9x^5-1$$

- Sign variations: 2. Negative real roots: 0.

Theorem

Descartes' Rule of Signs: The number of positive real roots of a polynomial is $\leq$ the number of sign transitions. The difference is even.

Estimating the number of positive real roots



- Recall our polynomial $x^5(x^2-3)^2-1$. Expanded out, this is:

$$+x^9-6x^7+9x^5-1$$

- Sign variations: 3. Positive real roots: 3.
- We can map $x \mapsto -x$ to get:

$$-x^9+6x^7-9x^5-1$$

- Sign variations: 2. Negative real roots: 0.

Theorem

Descartes' Rule of Signs: The number of positive real roots of a polynomial is $\leq$ the number of sign transitions. The difference is even.

Using Descartes' rule of signs

Theorem

Descartes' Rule of Signs: The number of positive real roots of a polynomial is $\leq$ the number of sign transitions. The difference is even.

- Simple consequences: if the number of sign variations of a polynomial is 0, it has no positive real roots. If the number is one, it has exactly one positive real root. (If the number is greater than one, we do not know how many positive real roots there are.)
- The combination of Möbius transformations and Descartes' rule of signs is all you need for real root isolation: to count the number of real roots in an interval, use a Möbius transformation to map that interval onto $(0, \infty)$.

Using Descartes' rule of signs

Theorem

Descartes' Rule of Signs: The number of positive real roots of a polynomial is $\leq$ the number of sign transitions. The difference is even.

- Simple consequences: if the number of sign variations of a polynomial is 0, it has no positive real roots. If the number is one, it has exactly one positive real root. (If the number is greater than one, we do not know how many positive real roots there are.)
- The combination of Möbius transformations and Descartes' rule of signs is all you need for real root isolation: to count the number of real roots in an interval, use a Möbius transformation to map that interval onto $(0, \infty)$.

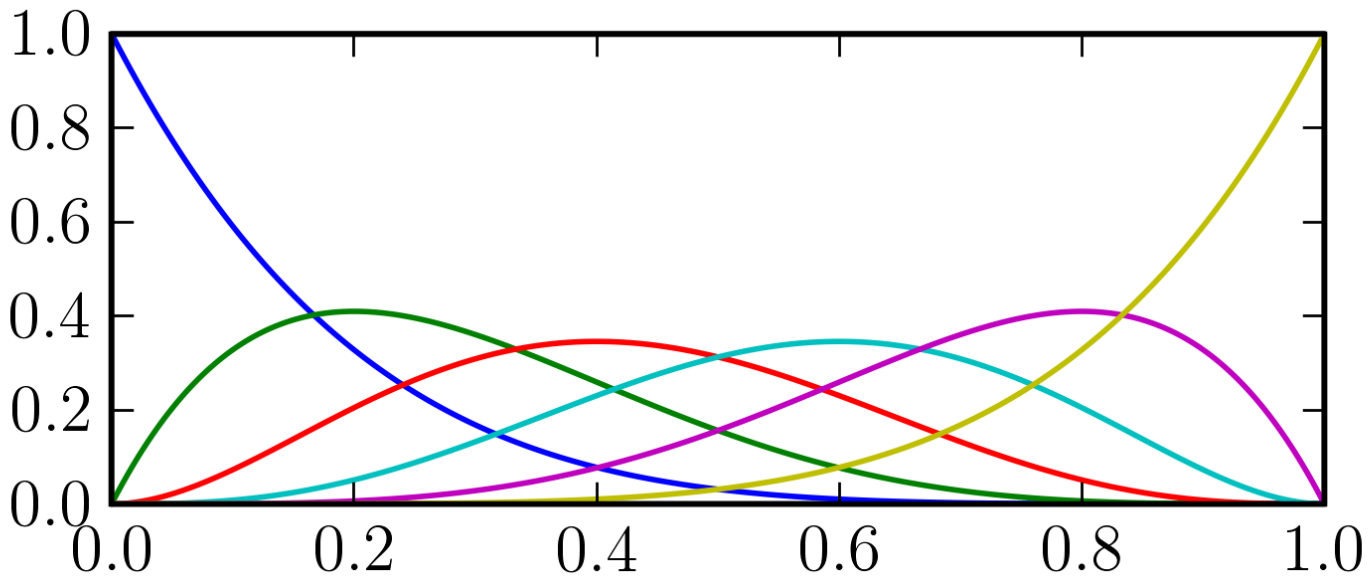
Outline

Definition of Bernstein polynomials

Definition

The Bernstein basis polynomials of degree n are

$$B_{k,n}(x) = \binom{n}{k} x^k (1-x)^{n-k}$$



Definition of Bernstein polynomials

Definition

The Bernstein basis polynomials of degree n are

$$B_{k,n}(x) = \binom{n}{k} x^k (1-x)^{n-k}$$

- Any polynomial of degree n or less can be expressed as a linear combination of the degree n Bernstein basis polynomials.
- I will write

$$\sum_{k=0}^{n} \beta_k B_{k,n}(x)$$

as $[\beta_k, \beta_{k-1}, \ldots, \beta_0]$ (note backwards order).

Definition of Bernstein polynomials

Definition

The Bernstein basis polynomials of degree n are

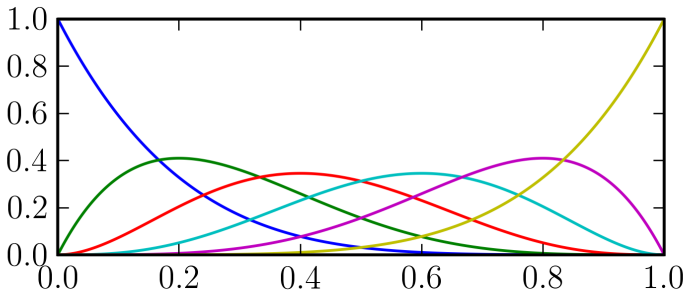
$$B_{k,n}(x) = \binom{n}{k} x^k (1-x)^{n-k}$$

- Any polynomial of degree n or less can be expressed as a linear combination of the degree n Bernstein basis polynomials.
- I will write

$$\sum_{k=0}^{n} \beta_k B_{k,n}(x)$$

as $[\beta_k, \beta_{k-1}, \ldots, \beta_0]$ (note backwards order).

Properties of the Bernstein basis

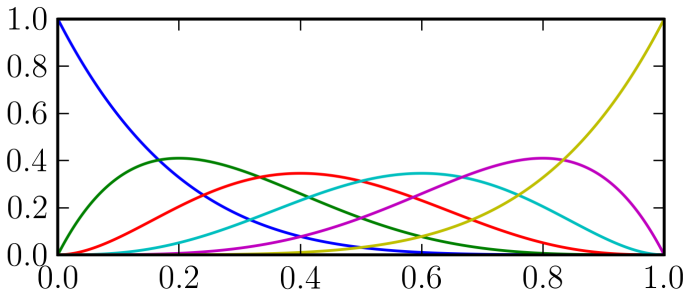


- If $0 \leq x \leq 1$ then $0 \leq B_{k,n}(x) \leq 1$.
- The degree-n Bernstein polynomials sum to 1:

$$\sum_{k=0}^{n} B_{k,n}(x) = 1$$

- If each Bernstein coefficient of P_1 is $\leq$ the corresponding Bernstein coefficient of P_2, then $P_1(x) \leq P_2(x)$ for $0 \leq x \leq 1$.
- If each Bernstein coefficient of P is between a and b, then $a \leq P(x) \leq b$ for $0 \leq x \leq 1$.

Properties of the Bernstein basis

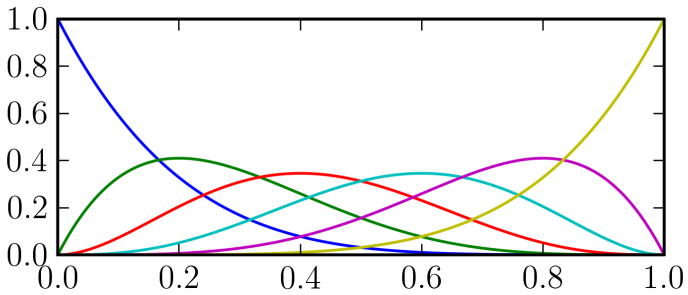


- If $0 \leq x \leq 1$ then $0 \leq B_{k,n}(x) \leq 1$.
- The degree-n Bernstein polynomials sum to 1:

$$\sum_{k=0}^{n} B_{k,n}(x) = 1$$

- If each Bernstein coefficient of P_1 is $\leq$ the corresponding Bernstein coefficient of P_2, then $P_1(x) \leq P_2(x)$ for $0 \leq x \leq 1$.
- If each Bernstein coefficient of P is between a and b, then $a \leq P(x) \leq b$ for $0 \leq x \leq 1$.

Properties of the Bernstein basis

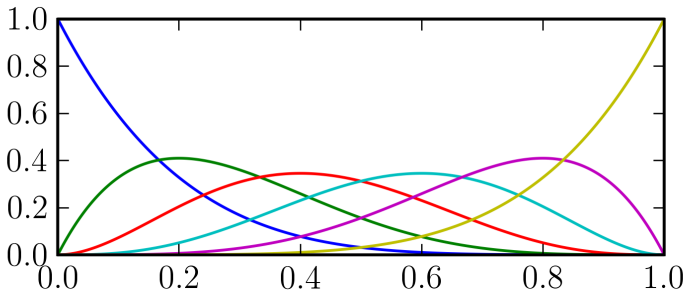


- If $0 \leq x \leq 1$ then $0 \leq B_{k,n}(x) \leq 1$.
- The degree-n Bernstein polynomials sum to 1:

$$\sum_{k=0}^{n} B_{k,n}(x) = 1$$

- If each Bernstein coefficient of P_1 is $\leq$ the corresponding Bernstein coefficient of P_2, then $P_1(x) \leq P_2(x)$ for $0 \leq x \leq 1$.
- If each Bernstein coefficient of P is between a and b, then $a \leq P(x) \leq b$ for $0 \leq x \leq 1$.

Properties of the Bernstein basis

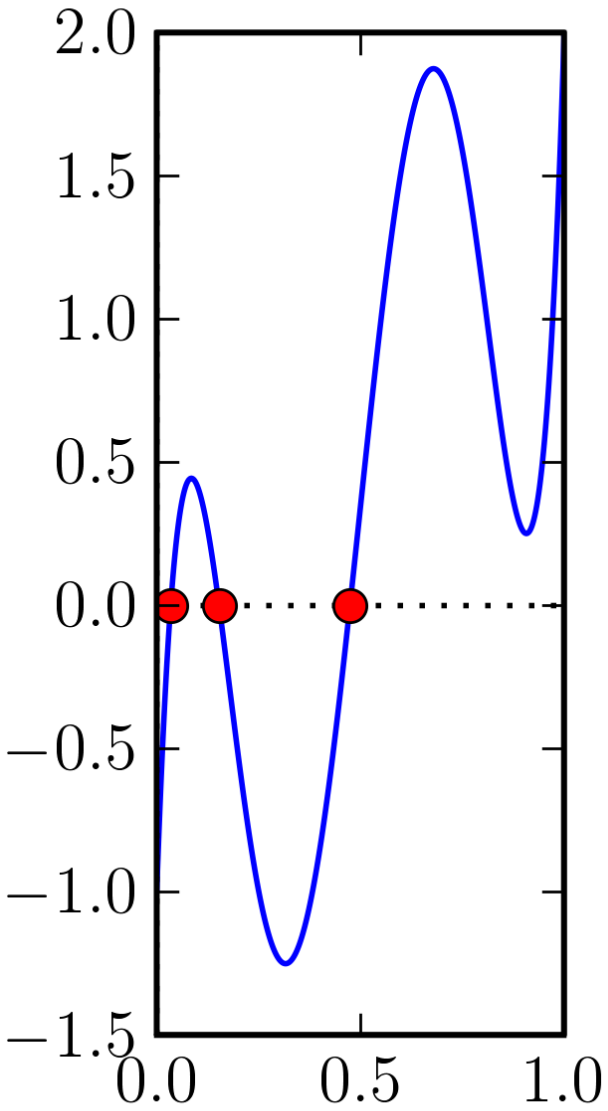


- If $0 \leq x \leq 1$ then $0 \leq B_{k,n}(x) \leq 1$.
- The degree-n Bernstein polynomials sum to 1:

$$\sum_{k=0}^{n} B_{k,n}(x) = 1$$

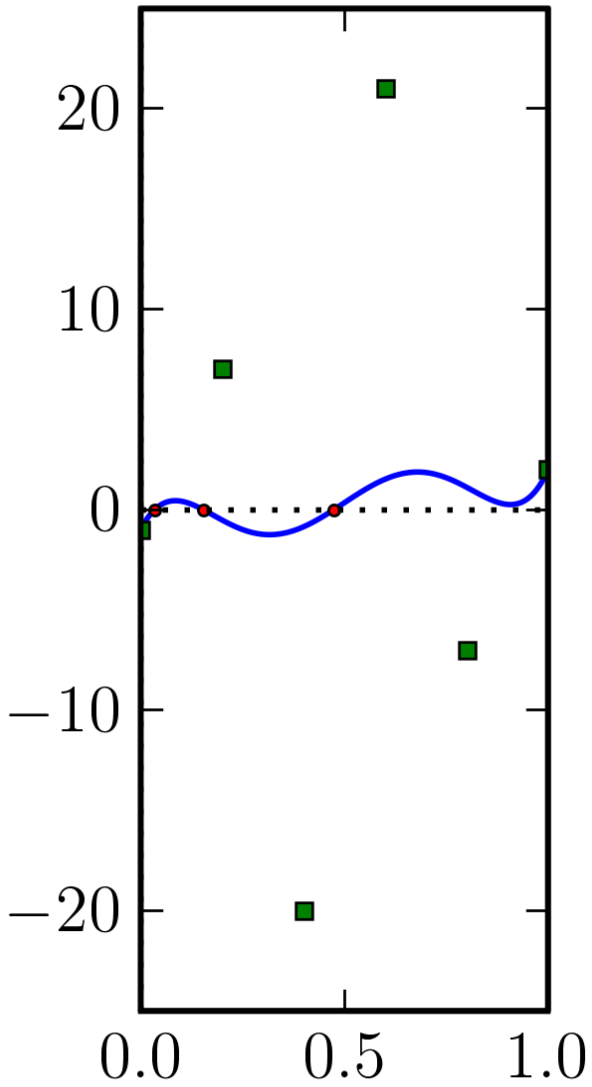
- If each Bernstein coefficient of P_1 is $\leq$ the corresponding Bernstein coefficient of P_2, then $P_1(x) \leq P_2(x)$ for $0 \leq x \leq 1$.
- If each Bernstein coefficient of P is between a and b, then $a \leq P(x) \leq b$ for $0 \leq x \leq 1$.

The Bernstein control polygon



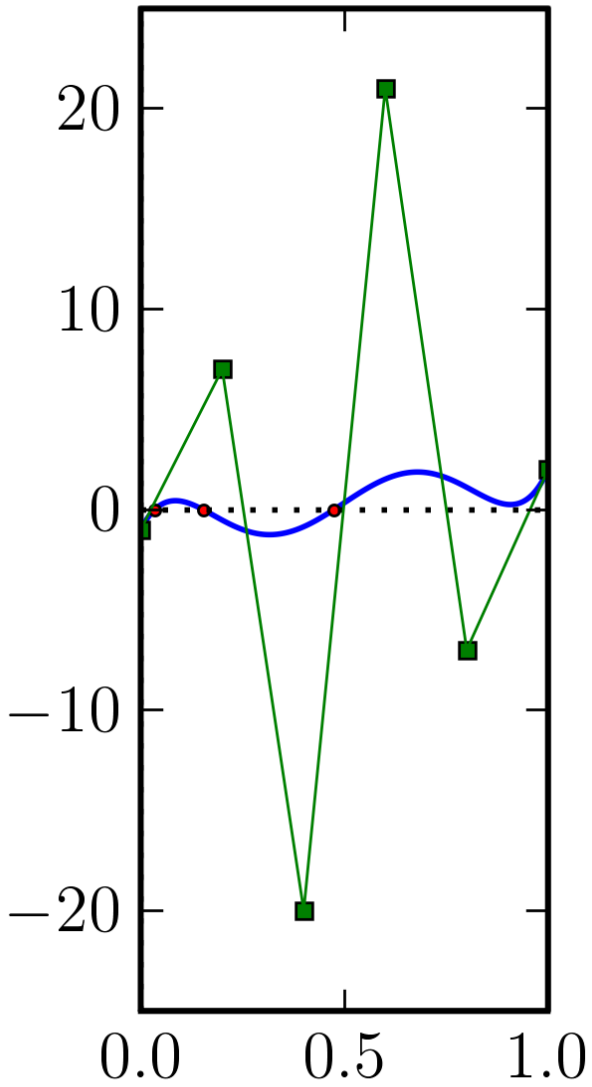
- Consider the polynomial $[-1, 7, -20, 21, -7, 2]$ (this is $483x^5 - 1200x^4 + 1030x^3 - 350x^2 + 40x - 1$).
- The first and last Bernstein coefficient match the values at 0 and 1.
- The slopes of the Bernstein coefficients match the slopes of the polynomial at 0 and 1.
- The polynomial between $x = 0$ and $x = 1$ is contained in the convex hull of the Bernstein coefficients.
- The number of real roots in the interval $(0, 1)$ is $\leq$ the number of sign variations in the Bernstein coefficients; the difference is even.

The Bernstein control polygon



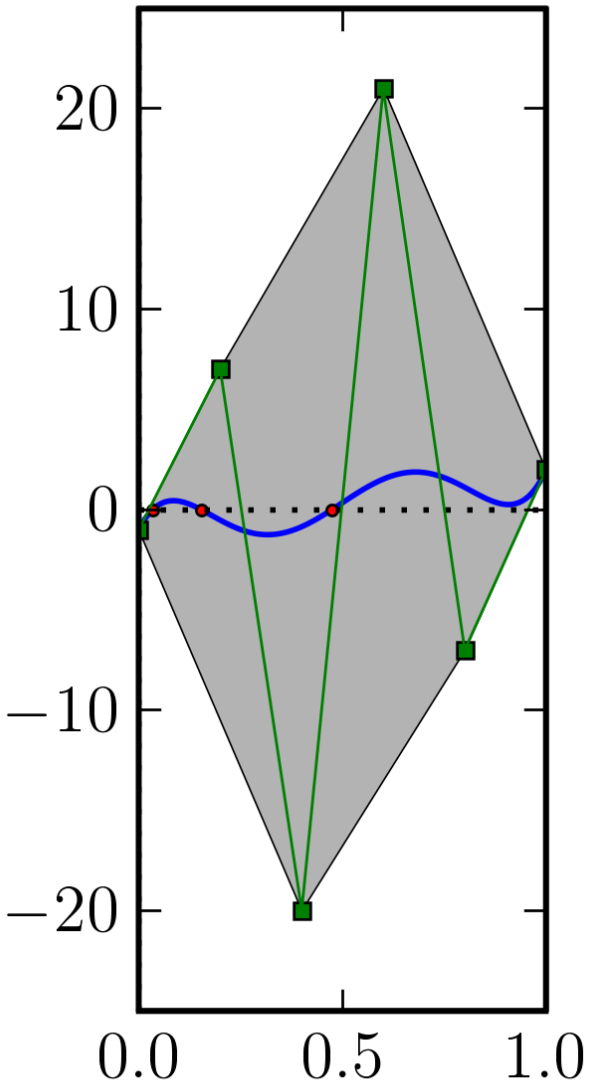
- Consider the polynomial $[-1, 7, -20, 21, -7, 2]$ (this is $483x^5 - 1200x^4 + 1030x^3 - 350x^2 + 40x - 1$).
- The first and last Bernstein coefficient match the values at 0 and 1.
- The slopes of the Bernstein coefficients match the slopes of the polynomial at 0 and 1.
- The polynomial between $x = 0$ and $x = 1$ is contained in the convex hull of the Bernstein coefficients.
- The number of real roots in the interval $(0, 1)$ is $\leq$ the number of sign variations in the Bernstein coefficients; the difference is even.

The Bernstein control polygon



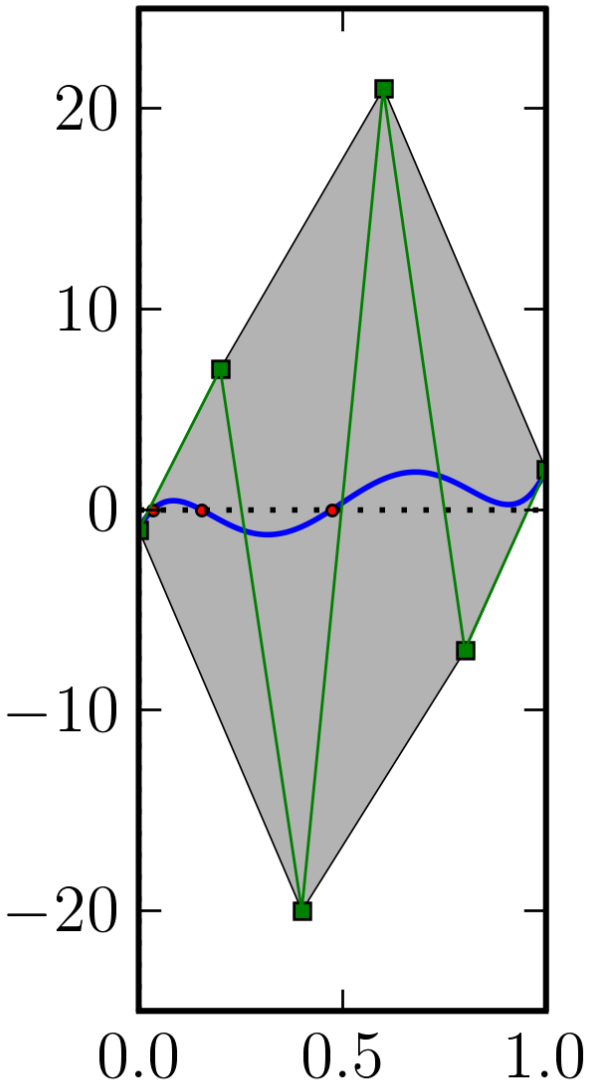
- Consider the polynomial $[-1, 7, -20, 21, -7, 2]$ (this is $483x^5 - 1200x^4 + 1030x^3 - 350x^2 + 40x - 1$).
- The first and last Bernstein coefficient match the values at 0 and 1.
- The slopes of the Bernstein coefficients match the slopes of the polynomial at 0 and 1.
- The polynomial between $x = 0$ and $x = 1$ is contained in the convex hull of the Bernstein coefficients.
- The number of real roots in the interval $(0, 1)$ is $\leq$ the number of sign variations in the Bernstein coefficients; the difference is even.

The Bernstein control polygon



- Consider the polynomial $[-1, 7, -20, 21, -7, 2]$ (this is $483x^5 - 1200x^4 + 1030x^3 - 350x^2 + 40x - 1$).
- The first and last Bernstein coefficient match the values at 0 and 1.
- The slopes of the Bernstein coefficients match the slopes of the polynomial at 0 and 1.
- The polynomial between $x = 0$ and $x = 1$ is contained in the convex hull of the Bernstein coefficients.
- The number of real roots in the interval $(0, 1)$ is $\leq$ the number of sign variations in the Bernstein coefficients; the difference is even.

The Bernstein control polygon



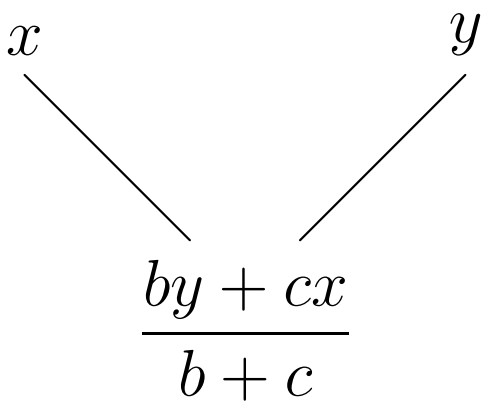
- Consider the polynomial $[-1, 7, -20, 21, -7, 2]$ (this is $483x^5 - 1200x^4 + 1030x^3 - 350x^2 + 40x - 1$).
- The first and last Bernstein coefficient match the values at 0 and 1.
- The slopes of the Bernstein coefficients match the slopes of the polynomial at 0 and 1.
- The polynomial between $x = 0$ and $x = 1$ is contained in the convex hull of the Bernstein coefficients.
- The number of real roots in the interval $(0, 1)$ is $\leq$ the number of sign variations in the Bernstein coefficients; the difference is even.

Outline

de Casteljau's algorithm—preliminaries

- Given a polynomial P in Bernstein basis form and a split point a, de Casteljau's algorithm will create two polynomials P_1 and P_2.
  - P_1 is P stretched so that $0 \mapsto 0$ and $a \mapsto 1$.
  - P_2 is P stretched and shifted so that $a \mapsto 0$ and $1 \mapsto 1$.

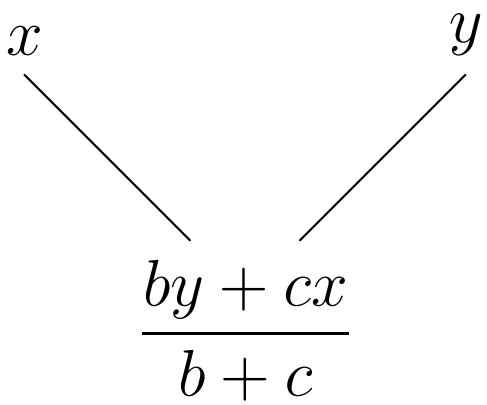
  We restrict ourselves to rational a, written in the form $\frac{b}{b+c}$.
- The basic operation in de Casteljau's algorithm is the weighted average:



de Casteljau's algorithm—preliminaries

- Given a polynomial P in Bernstein basis form and a split point a, de Casteljau's algorithm will create two polynomials P_1 and P_2.
  - P_1 is P stretched so that $0 \mapsto 0$ and $a \mapsto 1$.
  - P_2 is P stretched and shifted so that $a \mapsto 0$ and $1 \mapsto 1$.

  We restrict ourselves to rational a, written in the form $\frac{b}{b+c}$.
- The basic operation in de Casteljau's algorithm is the weighted average:



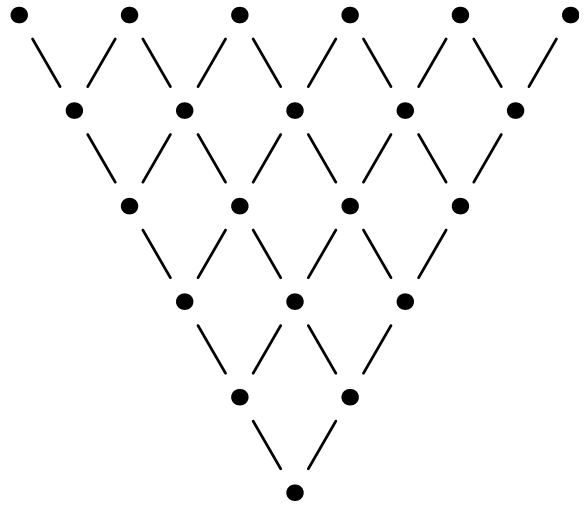
de Casteljau's algorithm

- Begin by writing down the Bernstein coefficients in a line. (Here, each dot represents one Bernstein coefficient.)

- Then, construct a triangle, where each number in the triangle is the weighted average of the two numbers above it.
- Now P_1 is found along the left side of the triangle and P_2 is found along the right side of the triangle. (The bottommost number becomes both the last coefficient of P_1 and the first coefficient of P_2.)

de Casteljau's algorithm

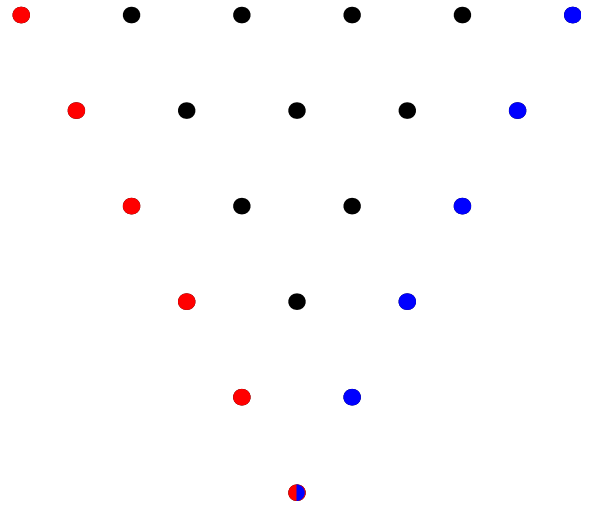
- Begin by writing down the Bernstein coefficients in a line. (Here, each dot represents one Bernstein coefficient.)



- Then, construct a triangle, where each number in the triangle is the weighted average of the two numbers above it.
- Now P_1 is found along the left side of the triangle and P_2 is found along the right side of the triangle. (The bottommost number becomes both the last coefficient of P_1 and the first coefficient of P_2.)

de Casteljau's algorithm

- Begin by writing down the Bernstein coefficients in a line. (Here, each dot represents one Bernstein coefficient.)



- Then, construct a triangle, where each number in the triangle is the weighted average of the two numbers above it.
- Now P_1 is found along the left side of the triangle and P_2 is found along the right side of the triangle. (The bottommost number becomes both the last coefficient of P_1 and the first coefficient of P_2.)

Consequences of de Casteljau's algorithm

- If your original degree-d polynomial has integral coefficients, then the bottommost number is a rational with denominator $\leq (b+c)^d$.
- To work with integral coefficients (typically much faster than rationals), you can multiply P_1 and P_2 by $(b+c)^d$. (This preserves the locations of the roots.)
- We see that the coefficients of P_1 and P_2 may have $d \log_2(b+c)$ more bits than the coefficients of P. If we split by bisection, this is just d. For our challenge polynomial, where we need more than 3300 splits and d is 999, we end up with numbers with 3.3 million bits.

Consequences of de Casteljau's algorithm

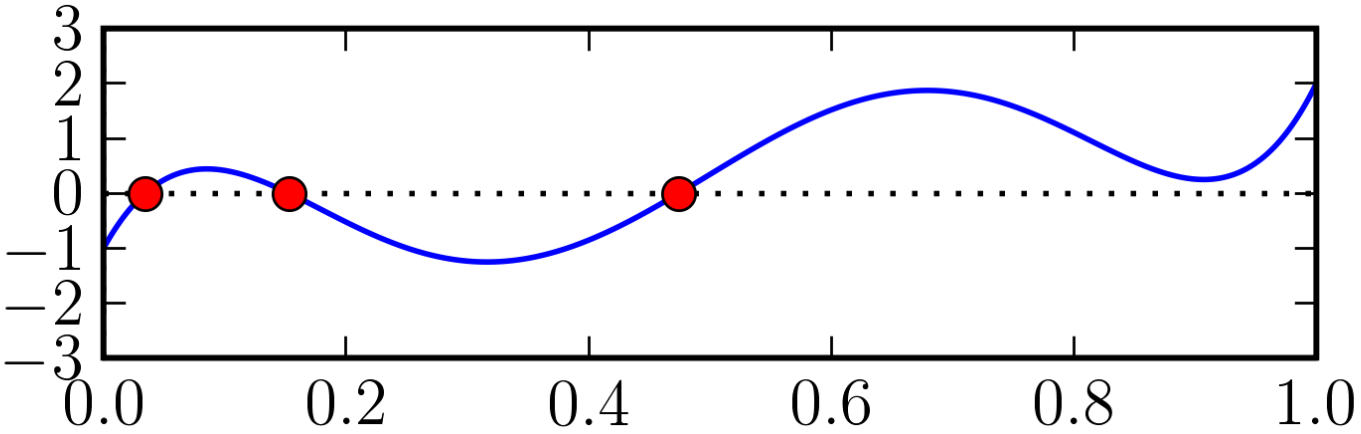
- If your original degree-d polynomial has integral coefficients, then the bottommost number is a rational with denominator $\leq (b+c)^d$.
- To work with integral coefficients (typically much faster than rationals), you can multiply P_1 and P_2 by $(b+c)^d$. (This preserves the locations of the roots.)
- We see that the coefficients of P_1 and P_2 may have $d \log_2(b+c)$ more bits than the coefficients of P. If we split by bisection, this is just d. For our challenge polynomial, where we need more than 3300 splits and d is 999, we end up with numbers with 3.3 million bits.

Consequences of de Casteljau's algorithm

- If your original degree-d polynomial has integral coefficients, then the bottommost number is a rational with denominator $\leq (b+c)^d$.
- To work with integral coefficients (typically much faster than rationals), you can multiply P_1 and P_2 by $(b+c)^d$. (This preserves the locations of the roots.)
- We see that the coefficients of P_1 and P_2 may have $d \log_2(b+c)$ more bits than the coefficients of P. If we split by bisection, this is just d. For our challenge polynomial, where we need more than 3300 splits and d is 999, we end up with numbers with 3.3 million bits.

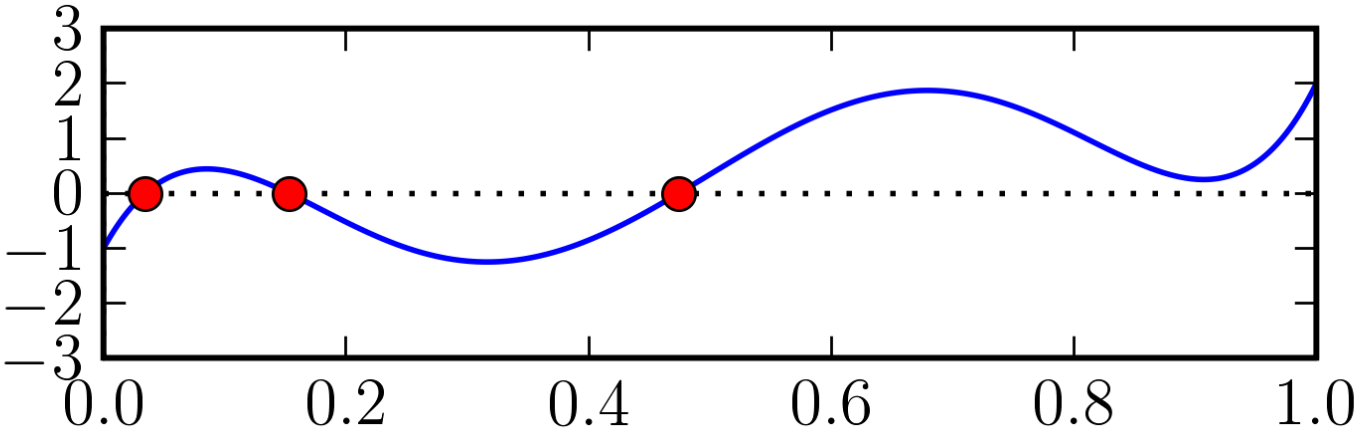
Outline

Approximating a Bernstein polynomial with fewer bits



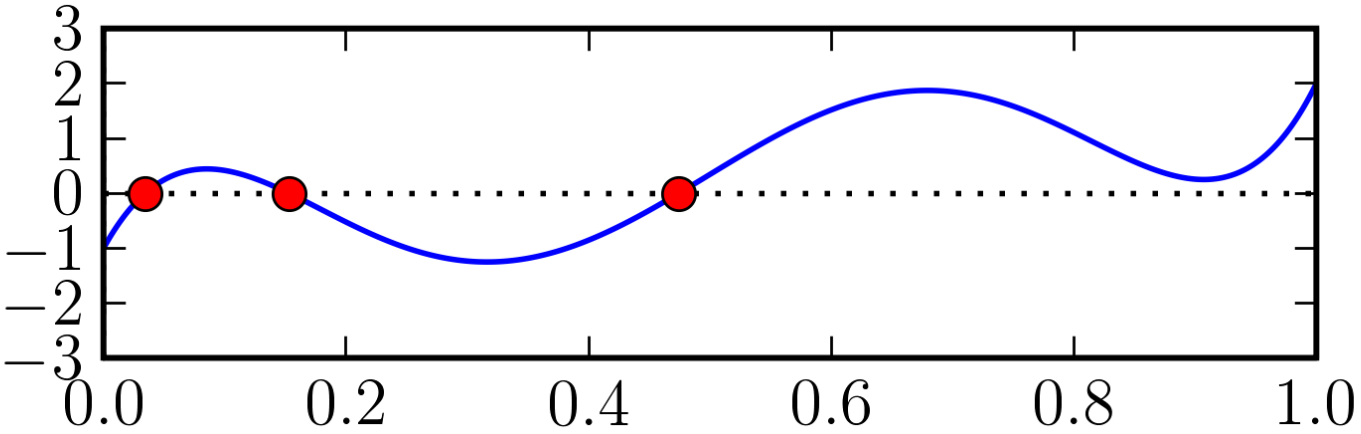
- We revisit $P = [-1, 7, -20, 21, -7, 2]$. P uses up to 5 bits per coefficient.
- If we only want to use 4 bits, we can use an interval polynomial: $Q = [-2, 6, -20, 20, -8, 2] + [0..1]$. This represents "the set of all polynomials where the first coefficient is between -2 and -1, ..., and the last coefficient is between 2 and 3". (So $P \in Q$.)
- We define $\underline{Q}$ as the lower bounds of the coefficients and $\overline{Q}$ as the upper bounds of the coefficients: $\underline{Q} = [-2, 6, -20, 20, -8, 2]$ and $\overline{Q} = [-1, 7, -19, 21, -7, 3]$.
- If $P \in Q$, then $\underline{Q}(x) \leq P(x) \leq \overline{Q}(x)$ when $0 \leq x \leq 1$.

Approximating a Bernstein polynomial with fewer bits



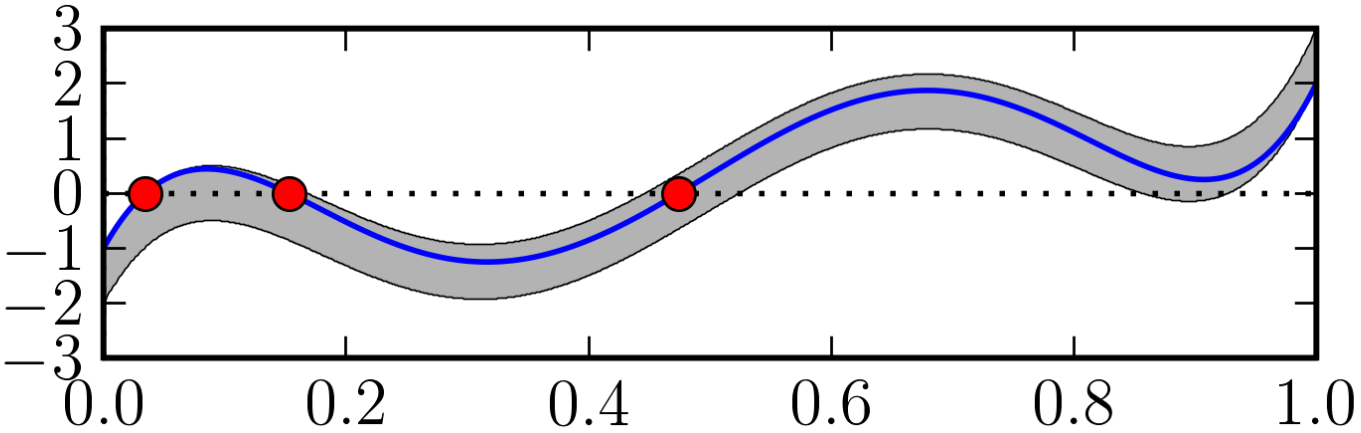
- We revisit $P = [-1, 7, -20, 21, -7, 2]$. P uses up to 5 bits per coefficient.
- If we only want to use 4 bits, we can use an interval polynomial: $Q = [-2, 6, -20, 20, -8, 2] + [0..1]$. This represents "the set of all polynomials where the first coefficient is between -2 and -1, ..., and the last coefficient is between 2 and 3". (So $P \in Q$.)
- We define $\underline{Q}$ as the lower bounds of the coefficients and $\overline{Q}$ as the upper bounds of the coefficients: $\underline{Q} = [-2, 6, -20, 20, -8, 2]$ and $\overline{Q} = [-1, 7, -19, 21, -7, 3]$.
- If $P \in Q$, then $\underline{Q}(x) \leq P(x) \leq \overline{Q}(x)$ when $0 \leq x \leq 1$.

Approximating a Bernstein polynomial with fewer bits



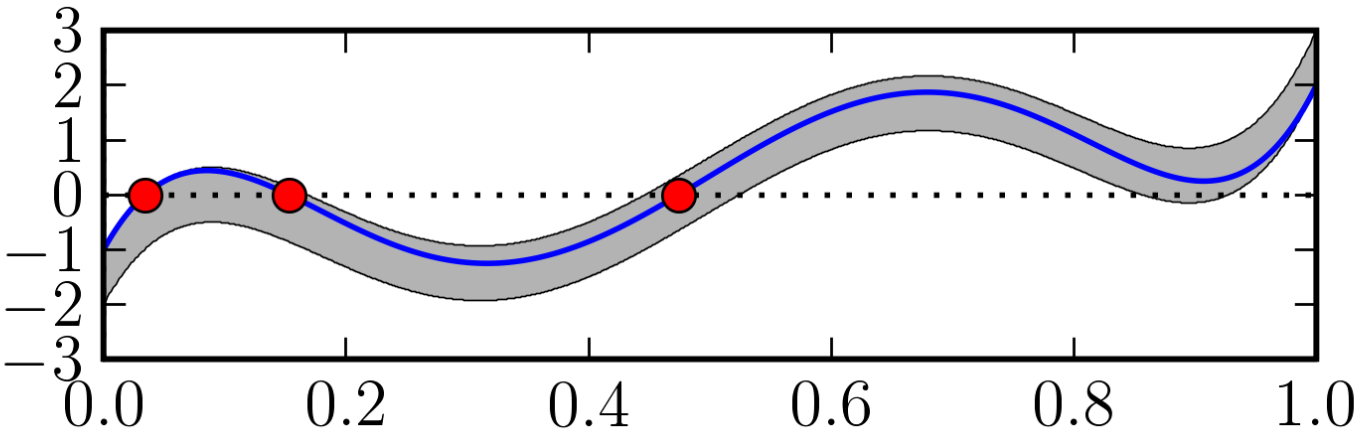
- We revisit $P = [-1, 7, -20, 21, -7, 2]$. P uses up to 5 bits per coefficient.
- If we only want to use 4 bits, we can use an interval polynomial: $Q = [-2, 6, -20, 20, -8, 2] + [0..1]$. This represents "the set of all polynomials where the first coefficient is between -2 and -1, ..., and the last coefficient is between 2 and 3". (So $P \in Q$.)
- We define $\underline{Q}$ as the lower bounds of the coefficients and $\overline{Q}$ as the upper bounds of the coefficients: $\underline{Q} = [-2, 6, -20, 20, -8, 2]$ and $\overline{Q} = [-1, 7, -19, 21, -7, 3]$.
- If $P \in Q$, then $\underline{Q}(x) \leq P(x) \leq \overline{Q}(x)$ when $0 \leq x \leq 1$.

Approximating a Bernstein polynomial with fewer bits



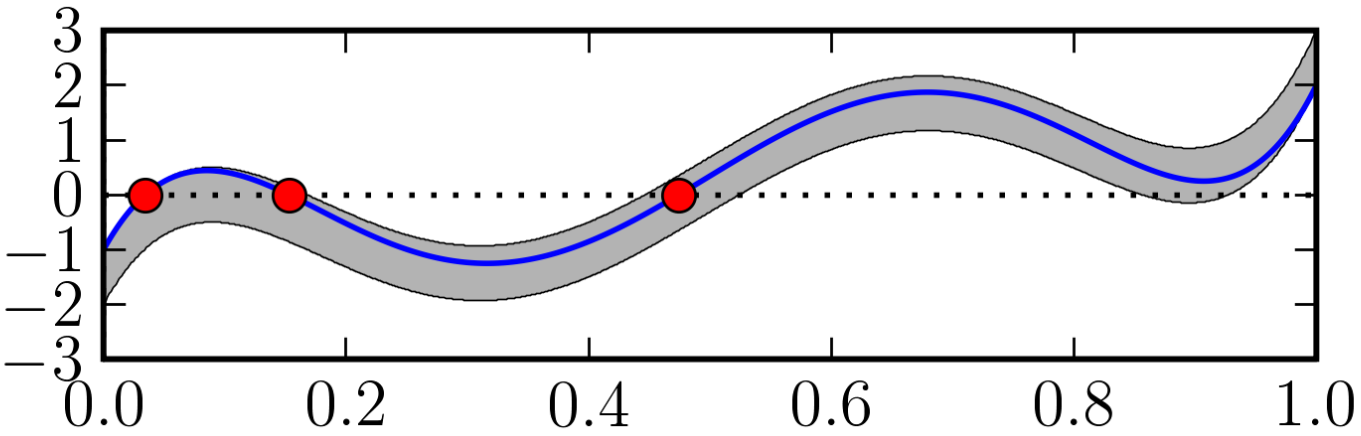
- We revisit $P = [-1, 7, -20, 21, -7, 2]$. P uses up to 5 bits per coefficient.
- If we only want to use 4 bits, we can use an interval polynomial: $Q = [-2, 6, -20, 20, -8, 2] + [0..1]$. This represents "the set of all polynomials where the first coefficient is between -2 and -1, ..., and the last coefficient is between 2 and 3". (So $P \in Q$.)
- We define $\underline{Q}$ as the lower bounds of the coefficients and $\overline{Q}$ as the upper bounds of the coefficients: $\underline{Q} = [-2, 6, -20, 20, -8, 2]$ and $\overline{Q} = [-1, 7, -19, 21, -7, 3]$.
- If $P \in Q$, then $\underline{Q}(x) \le P(x) \le \overline{Q}(x)$ when $0 \le x \le 1$.

Using interval polynomials



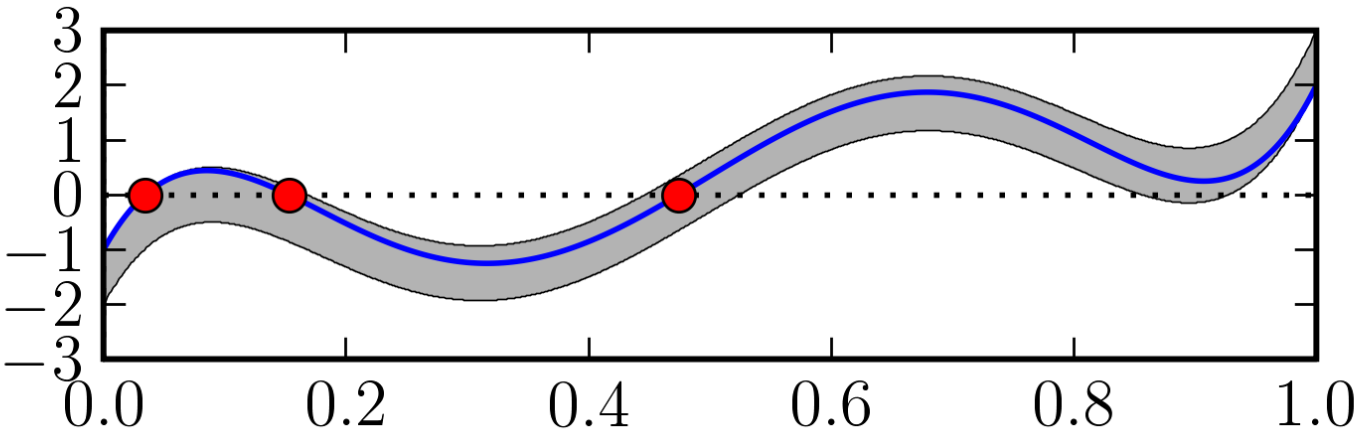
- We've lost a lot of information by dropping that last bit: just looking at Q, we may have either 1, 3, or 5 real roots.
- When splitting an interval polynomial, we reject split points where we cannot assign a definite sign.
- We will want to find as much information as we can with a given precision interval polynomial; if we cannot totally isolate the roots, we will try again with higher precision.

Using interval polynomials



- We've lost a lot of information by dropping that last bit: just looking at Q, we may have either 1, 3, or 5 real roots.
- When splitting an interval polynomial, we reject split points where we cannot assign a definite sign.
- We will want to find as much information as we can with a given precision interval polynomial; if we cannot totally isolate the roots, we will try again with higher precision.

Using interval polynomials



- We've lost a lot of information by dropping that last bit: just looking at Q, we may have either 1, 3, or 5 real roots.
- When splitting an interval polynomial, we reject split points where we cannot assign a definite sign.
- We will want to find as much information as we can with a given precision interval polynomial; if we cannot totally isolate the roots, we will try again with higher precision.

de Casteljau's algorithm for intervals

- The interval version of de Casteljau's algorithm increases the error bound by d on a degree-d polynomial.
- On our challenge polynomial, with about 3300 splittings and degree about 1000, this means we will end up with an error bound of about 3.3 million. Without this error, we could isolate the roots by starting with an initial interval polynomial with 7140-bit coefficients; with the error, we need 7162-bit coefficients.

de Casteljau's algorithm for intervals

- The interval version of de Casteljau's algorithm increases the error bound by d on a degree-d polynomial.
- On our challenge polynomial, with about 3300 splittings and degree about 1000, this means we will end up with an error bound of about 3.3 million. Without this error, we could isolate the roots by starting with an initial interval polynomial with 7140-bit coefficients; with the error, we need 7162-bit coefficients.

Bernstein polynomials in floating point

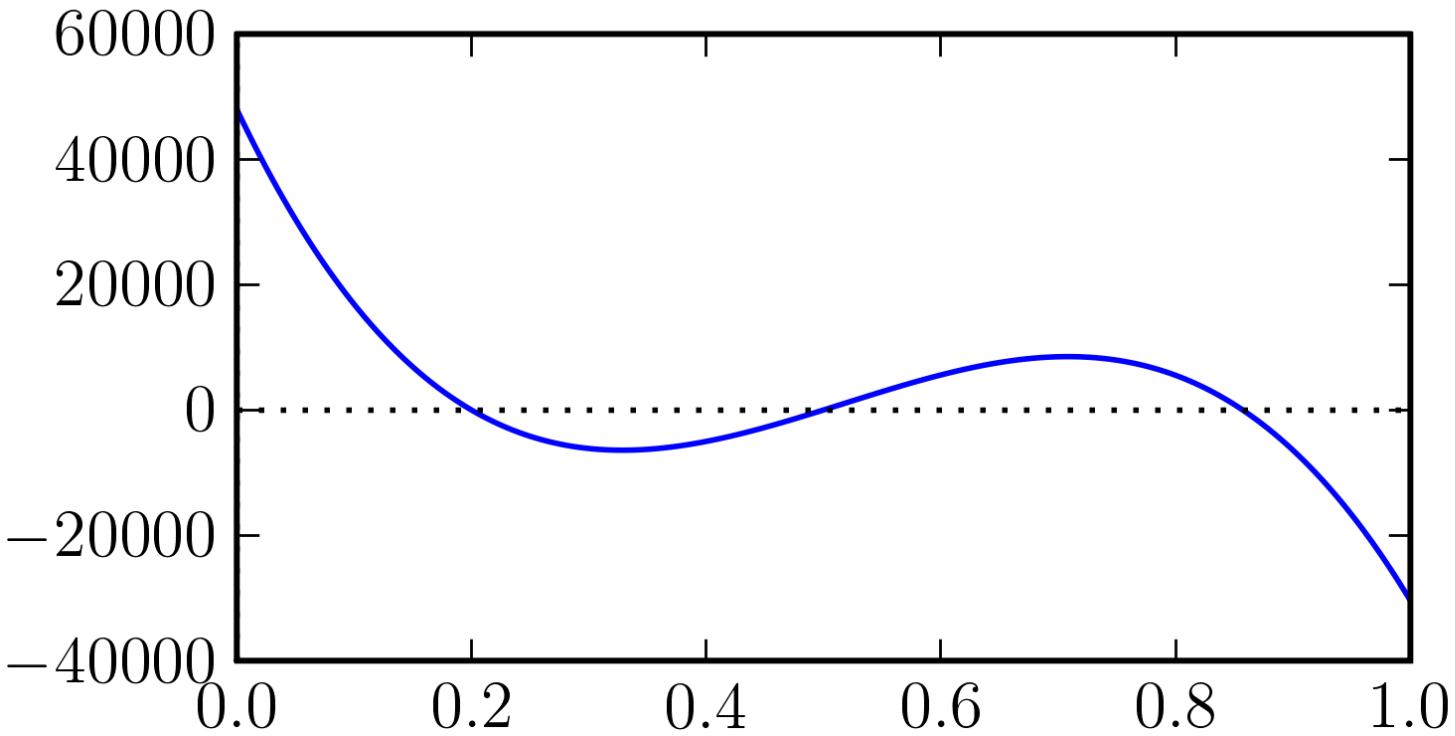
- To hold an interval Bernstein polynomial with less than 53 bits of precision, we can use a vector of IEEE double-precision floats. We can carry out de Casteljau's algorithm in floating point, and maintain error bounds in much the same way we do for the integer version (assuming correct, IEEE-compliant rounding).
- This only gains a constant factor in performance over using GMP—but it's a huge constant factor.

Bernstein polynomials in floating point

- To hold an interval Bernstein polynomial with less than 53 bits of precision, we can use a vector of IEEE double-precision floats. We can carry out de Casteljau's algorithm in floating point, and maintain error bounds in much the same way we do for the integer version (assuming correct, IEEE-compliant rounding).
- This only gains a constant factor in performance over using GMP—but it's a huge constant factor.

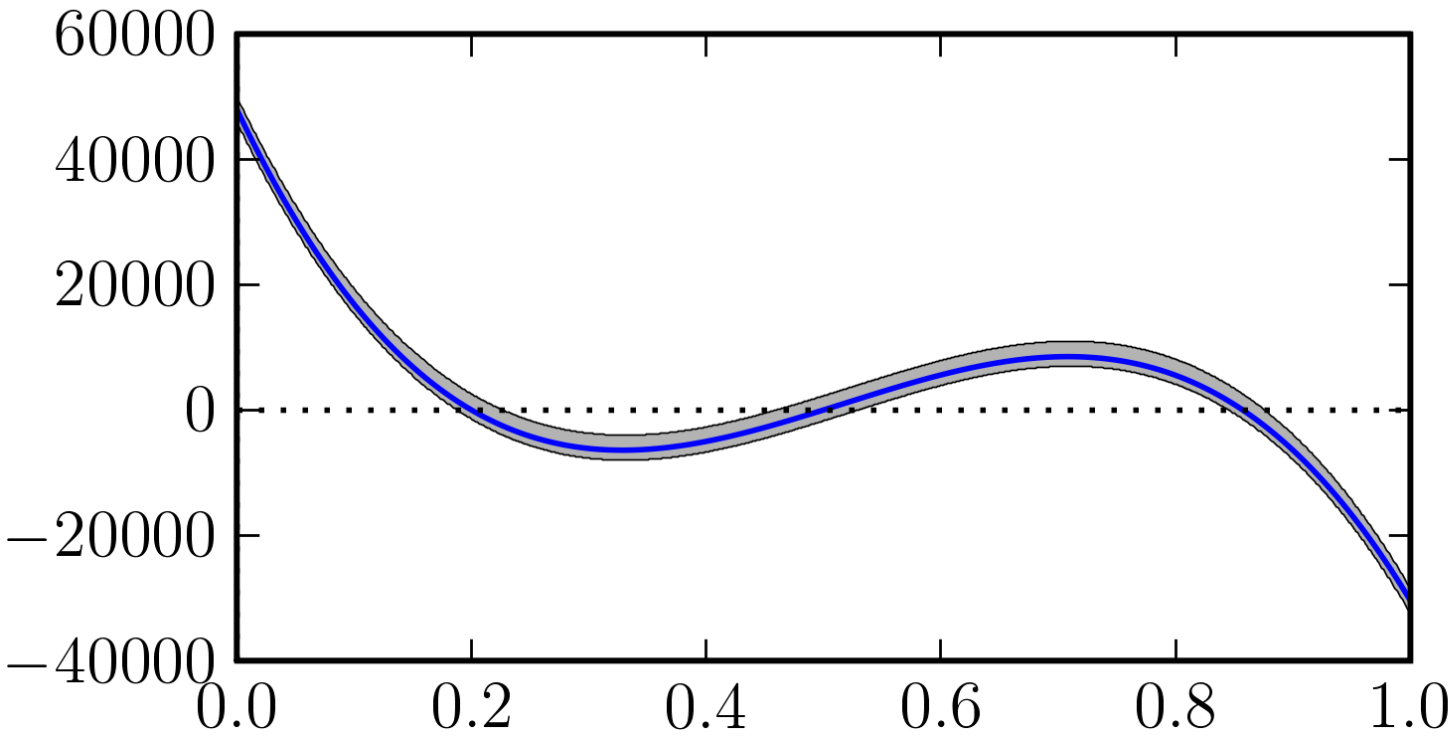
Outline

Degree reduction—an example



- Consider the quartic polynomial $[48000, -50600, -600, 47100, -30400]$ (with roots at $\frac{1}{5}$, $\frac{1}{2}$, $\frac{6}{7}$, and 20).
- Between 0 and 1, it is bounded by the cubic interval polynomial $[45999, -82801, 73599, -32401] + [0..4001]$.

Degree reduction—an example



- Consider the quartic polynomial $[48000, -50600, -600, 47100, -30400]$ (with roots at $\frac{1}{5}$, $\frac{1}{2}$, $\frac{6}{7}$, and 20).
- Between 0 and 1, it is bounded by the cubic interval polynomial $[45999, -82801, 73599, -32401] + [0..4001]$.

Degree reduction considerations

- For any polynomial and any interval, there is a fixed absolute error caused by degree reduction.
- When reducing to a degree-d polynomial, shrinking the interval by a factor of x will reduce this absolute error by at least x^d.

Degree reduction considerations

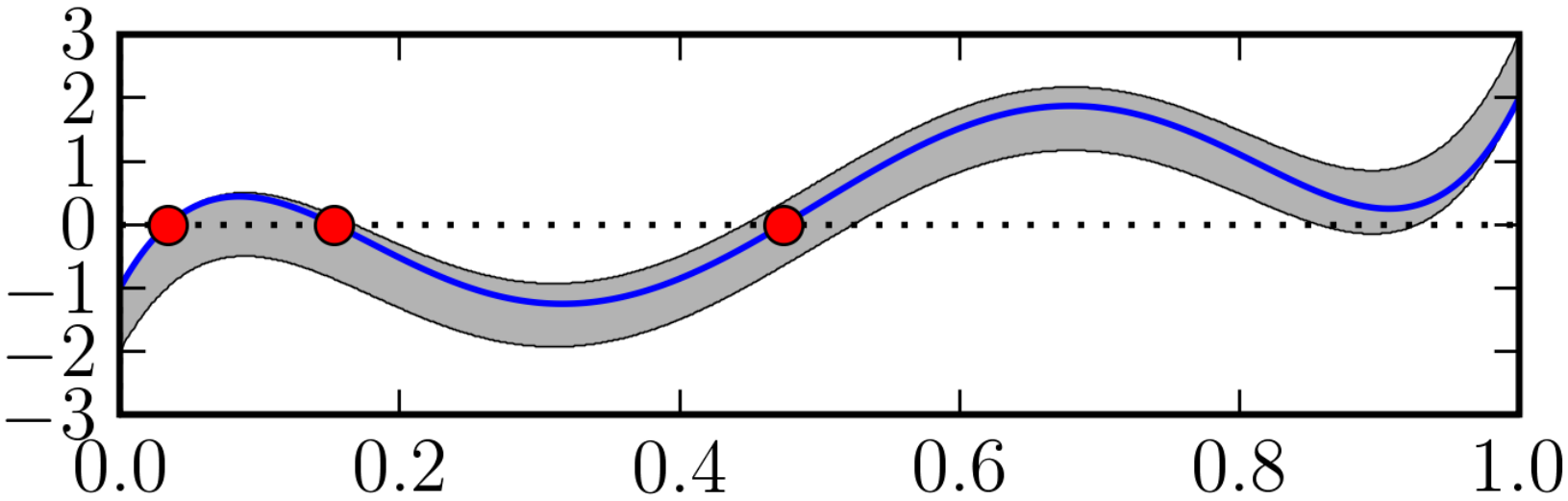
- For any polynomial and any interval, there is a fixed absolute error caused by degree reduction.
- When reducing to a degree-d polynomial, shrinking the interval by a factor of x will reduce this absolute error by at least x^d.

Degree reduction limitations

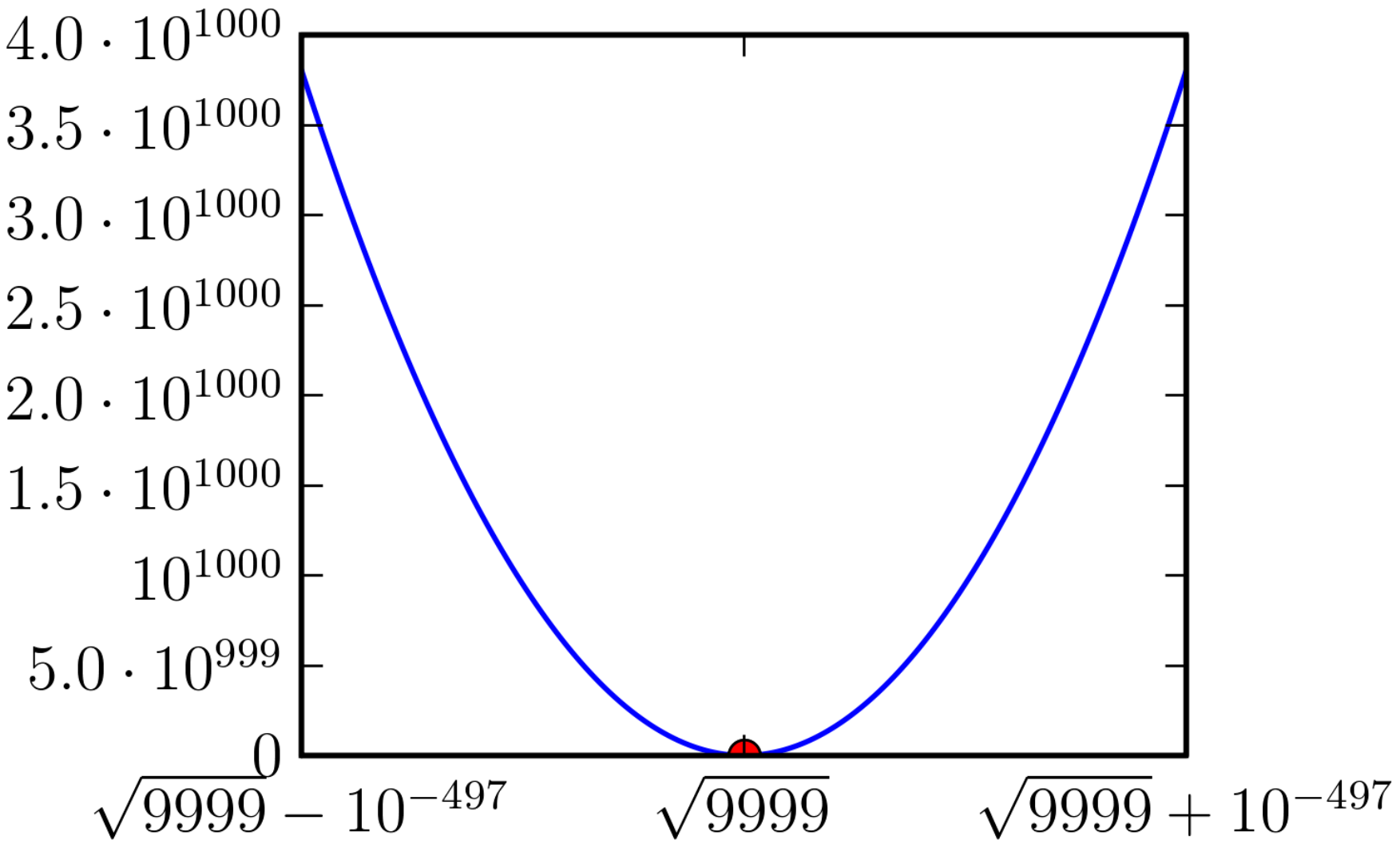
- With my algorithm, performing a degree reduction from degree a to degree b involves a matrix $\mathbf{R}_{a,b}$. Computing this matrix requires an exact matrix inversion of a $b \times b$ matrix of rationals.
- To avoid spending too much time in this matrix inversion, I only perform degree reduction to polynomials of degree ≤ 30.

Outline

Visual aid #1



Visual aid #2

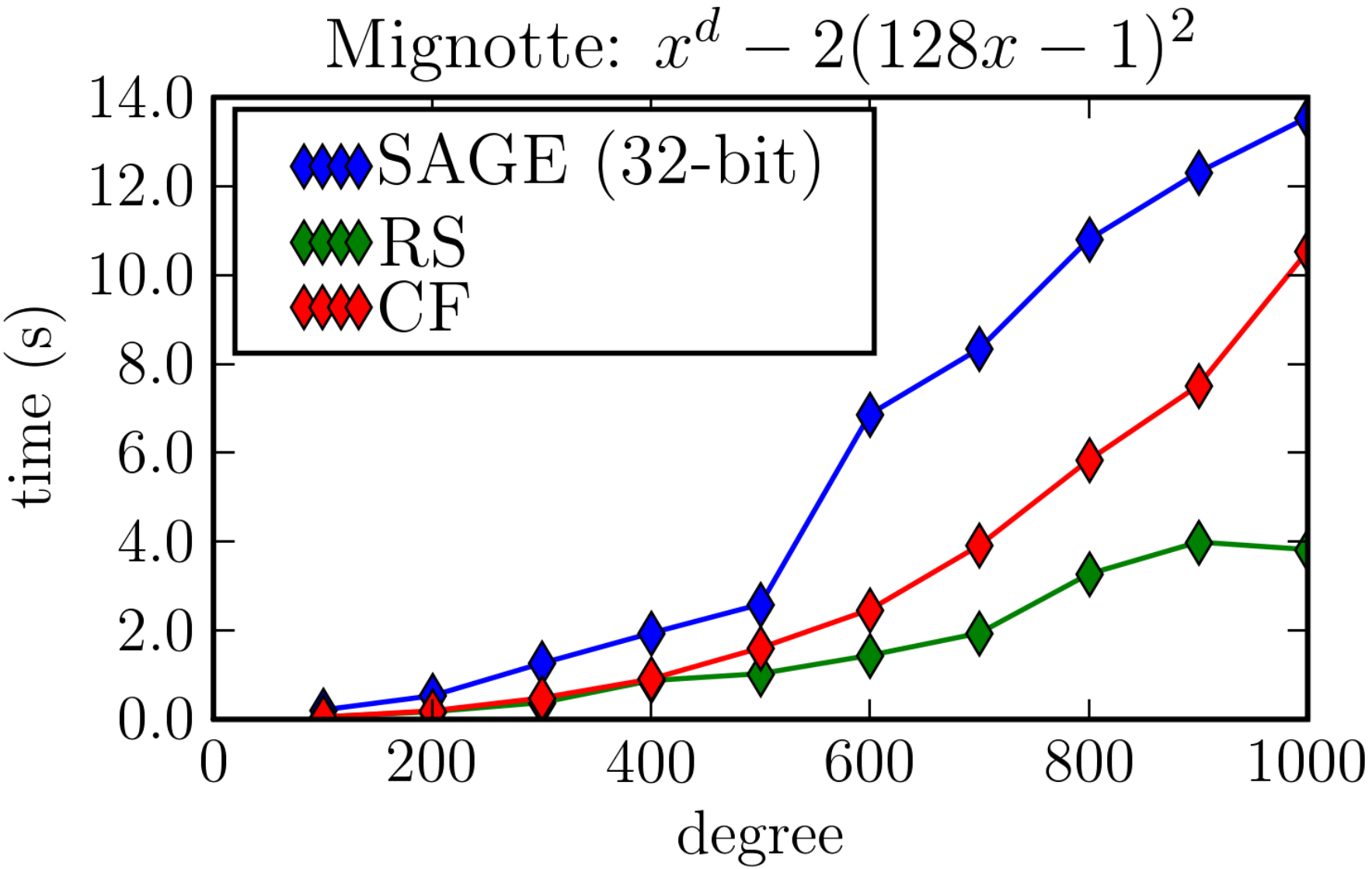

$4.0 \cdot 10^{1000}$
$3.5 \cdot 10^{1000}$
$3.0 \cdot 10^{1000}$
$2.5 \cdot 10^{1000}$
$2.0 \cdot 10^{1000}$
$1.5 \cdot 10^{1000}$
10^{1000}
$5.0 \cdot 10^{999}$
0
$\sqrt{9999} - 10^{-497}$
$\sqrt{9999}$
$\sqrt{9999} + 10^{-497}$

Outline

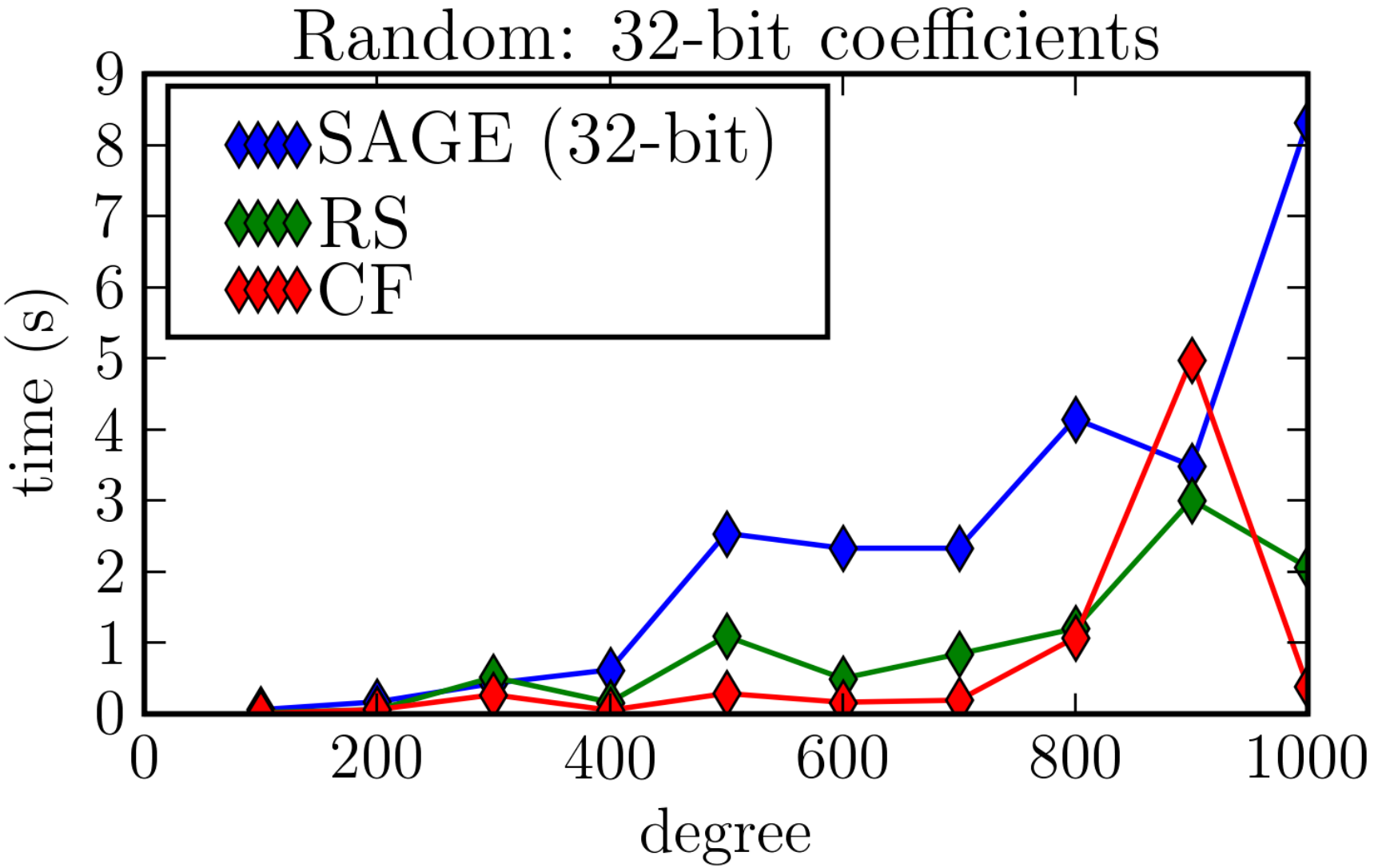
Preliminary benchmarks

Disclaimer: Each number in the following graphs comes from a single run of the associated program (rather than "best of 3" or "average of several runs").
The "random polynomials" graph uses only a single polynomial of each degree. (All algorithms are tested on the same polynomial).

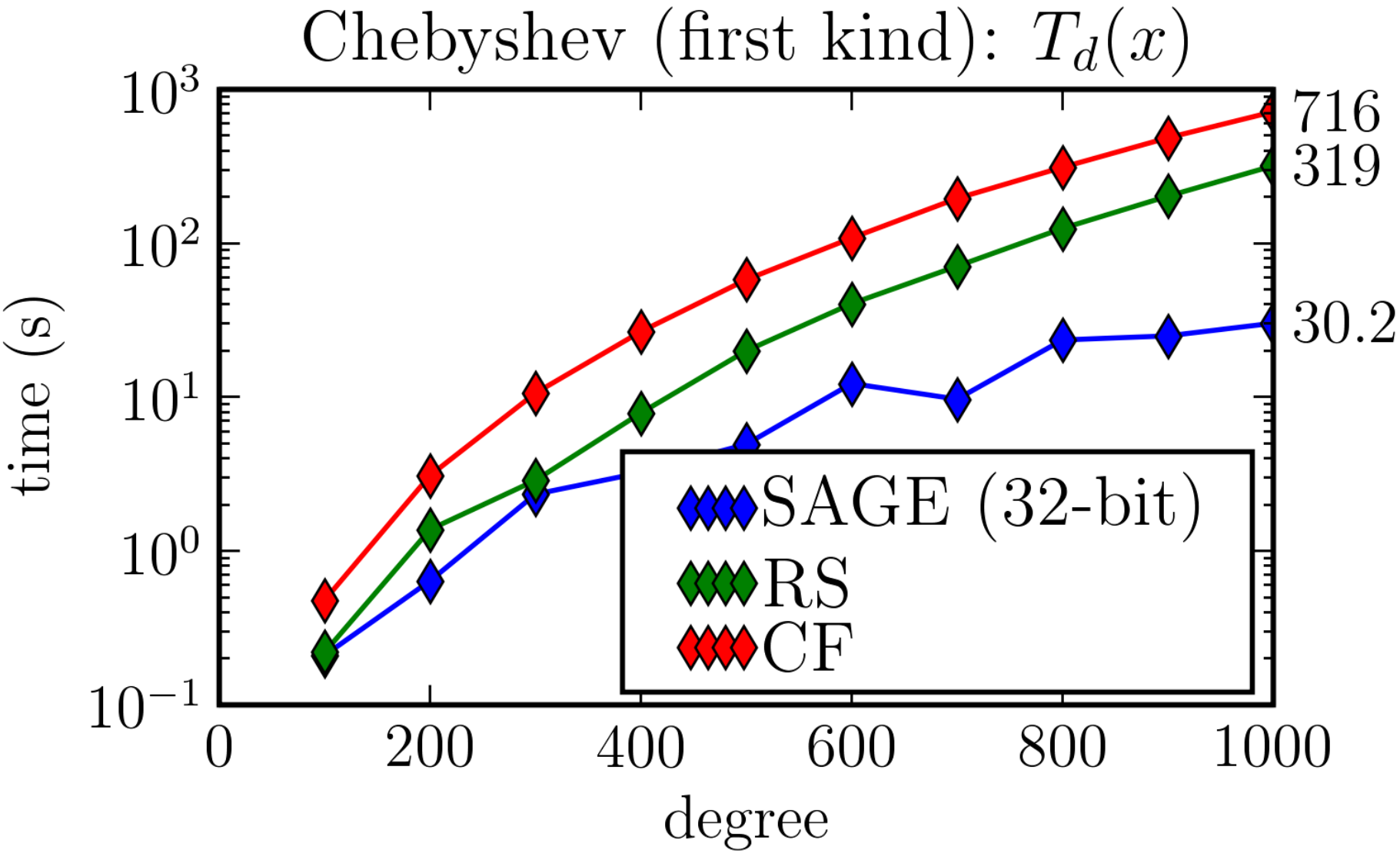
The bad news


Mignotte: $x^d - 2(128x - 1)^2$
SAGE (32-bit)
RS
CF
time (s)
14.0
12.0
10.0
8.0
6.0
4.0
2.0
0.0
0
200
400
600
800
1000
degree

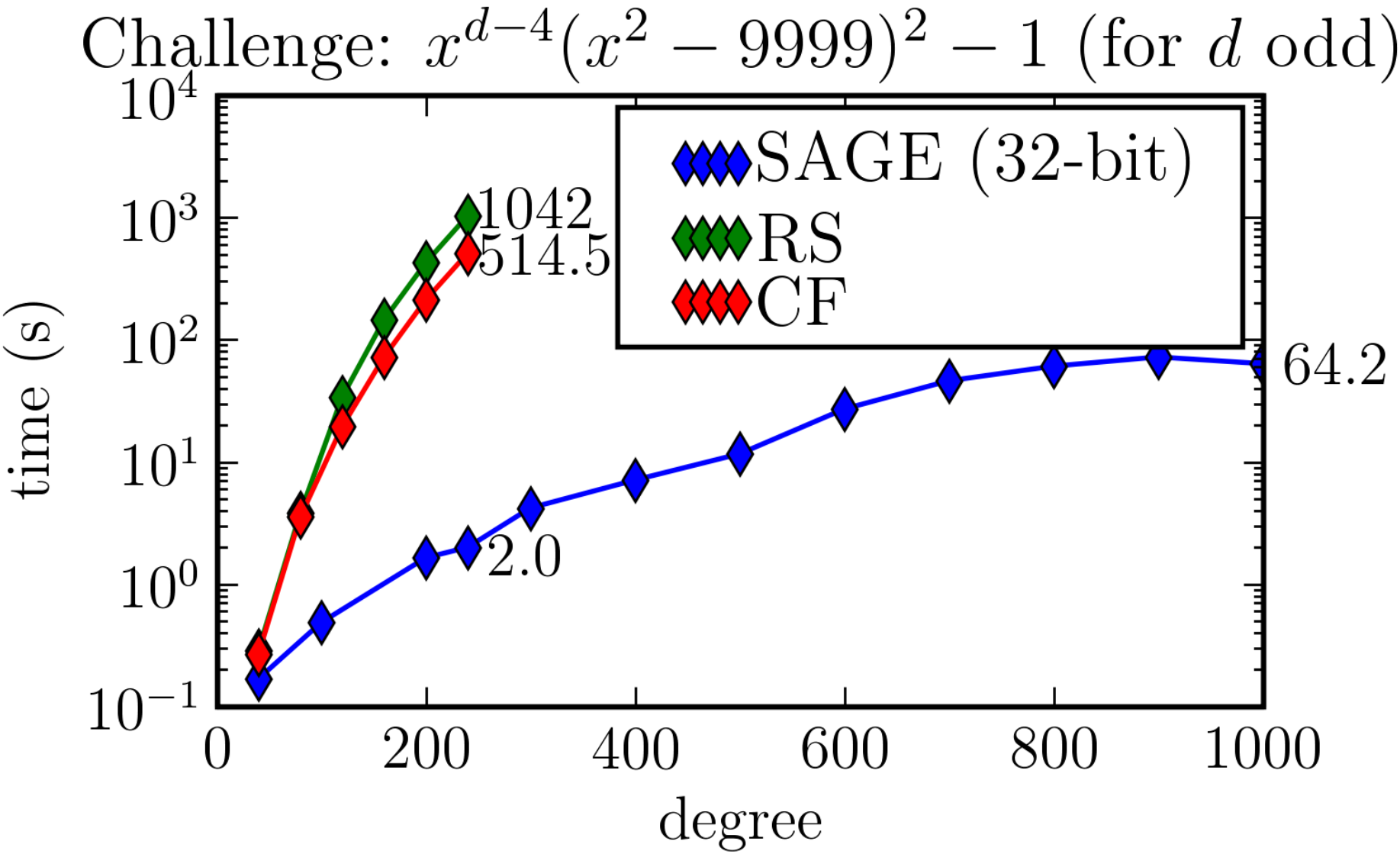
The bad news


Random: 32-bit coefficients
SAGE (32-bit)
RS
CF
time (s)
0
1
2
3
4
5
6
7
8
9
0
200
400
600
800
1000
degree

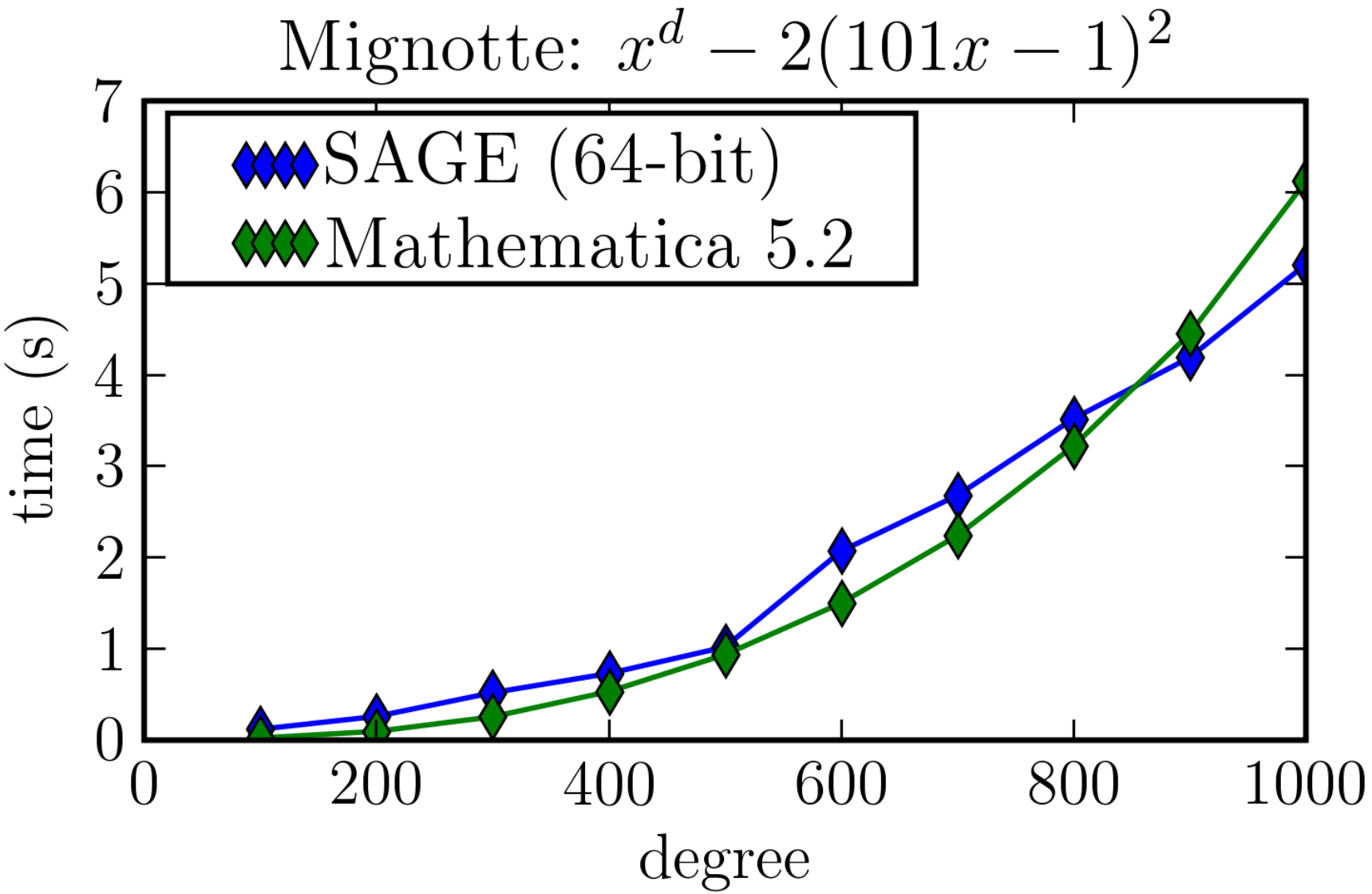
The good news



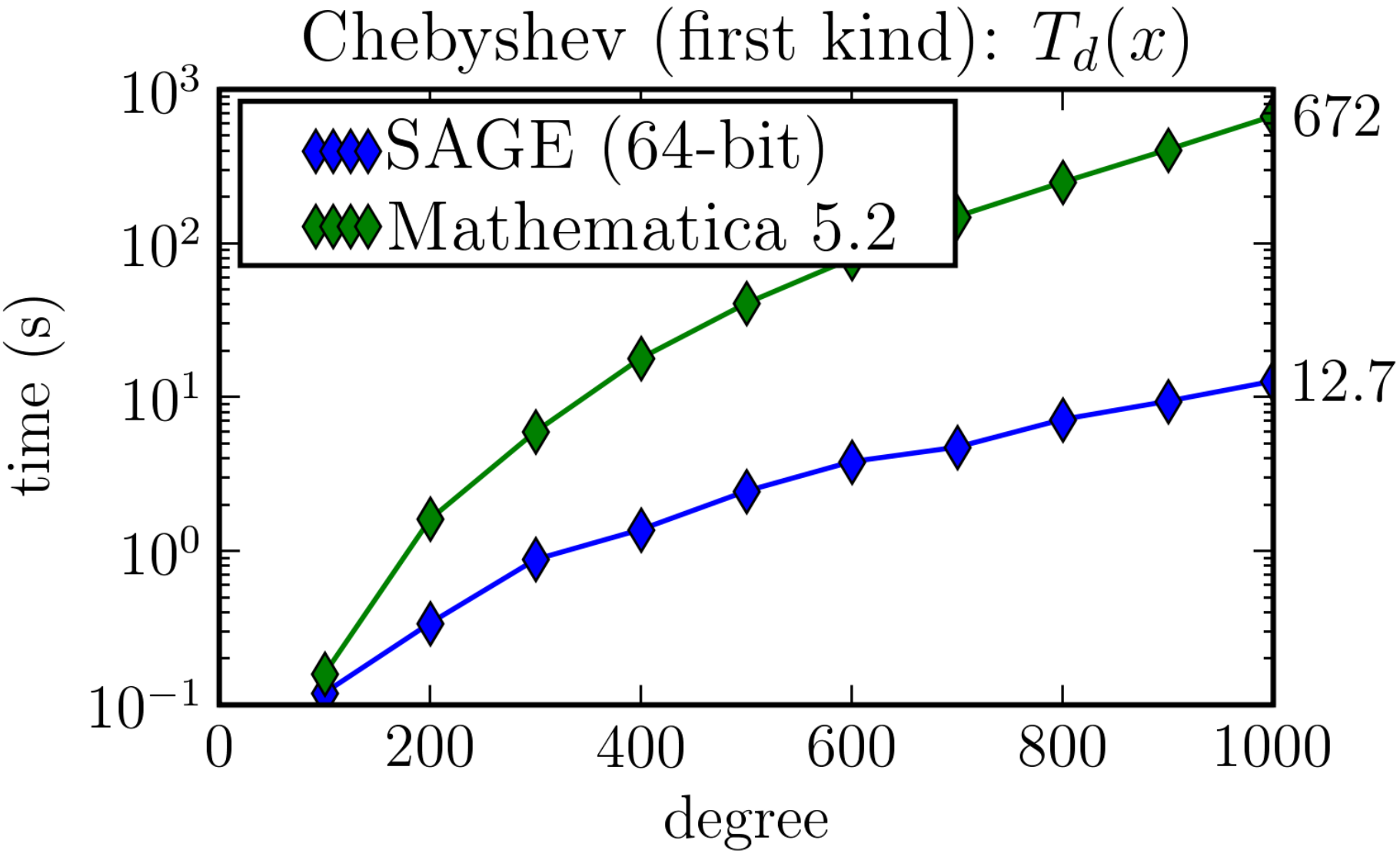
The good news



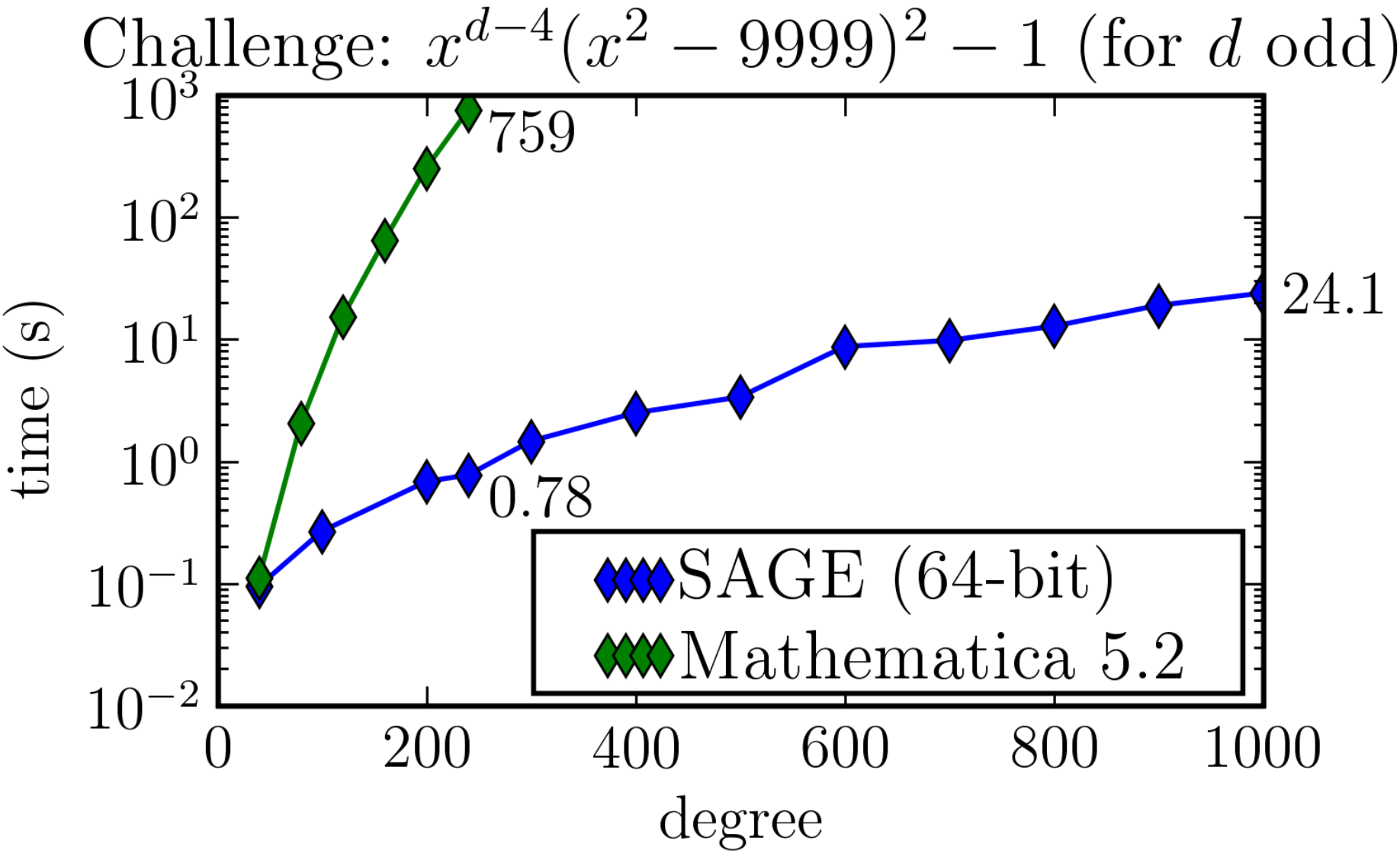
The good news



The good news



The good news



Outline

Ideas for speed improvements

- Faster de Casteljau's algorithm
  - SSE2
  - Anatole Ruslanov's register tiling
- Partial degree reduction
- Faster degree reduction
- Faster degree elevation
- Better choices for when to degree reduce, when to do high-resolution splitting
- Better split point selection
- Better choice of initial Möbius transformation
- More choices of interval Bernstein polynomial representation types (between GMP and native float): double-double? quad-double? fixed-precision multi-word arithmetic?
- Start with exact strategy and switch over to intervals
- Move more to Pyrex; local optimizations

Ideas for speed improvements

- Faster de Casteljau's algorithm
  - SSE2
  - Anatole Ruslanov's register tiling
- Partial degree reduction
- Faster degree reduction
- Faster degree elevation
- Better choices for when to degree reduce, when to do high-resolution splitting
- Better split point selection
- Better choice of initial Möbius transformation
- More choices of interval Bernstein polynomial representation types (between GMP and native float): double-double? quad-double? fixed-precision multi-word arithmetic?
- Start with exact strategy and switch over to intervals
- Move more to Pyrex; local optimizations

Ideas for speed improvements

- Faster de Casteljau's algorithm
  - SSE2
  - Anatole Ruslanov's register tiling
- Partial degree reduction
- Faster degree reduction
- Faster degree elevation
- Better choices for when to degree reduce, when to do high-resolution splitting
- Better split point selection
- Better choice of initial Möbius transformation
- More choices of interval Bernstein polynomial representation types (between GMP and native float): double-double? quad-double? fixed-precision multi-word arithmetic?
- Start with exact strategy and switch over to intervals
- Move more to Pyrex; local optimizations

Ideas for speed improvements

- Faster de Casteljau's algorithm
  - SSE2
  - Anatole Ruslanov's register tiling
- Partial degree reduction
- Faster degree reduction
- Faster degree elevation
- Better choices for when to degree reduce, when to do high-resolution splitting
- Better split point selection
- Better choice of initial Möbius transformation
- More choices of interval Bernstein polynomial representation types (between GMP and native float): double-double? quad-double? fixed-precision multi-word arithmetic?
- Start with exact strategy and switch over to intervals
- Move more to Pyrex; local optimizations

Ideas for speed improvements

- Faster de Casteljau's algorithm
  - SSE2
  - Anatole Ruslanov's register tiling
- Partial degree reduction
- Faster degree reduction
- Faster degree elevation
- Better choices for when to degree reduce, when to do high-resolution splitting
- Better split point selection
- Better choice of initial Möbius transformation
- More choices of interval Bernstein polynomial representation types (between GMP and native float): double-double? quad-double? fixed-precision multi-word arithmetic?
- Start with exact strategy and switch over to intervals
- Move more to Pyrex; local optimizations

Ideas for speed improvements

- Faster de Casteljau's algorithm
  - SSE2
  - Anatole Ruslanov's register tiling
- Partial degree reduction
- Faster degree reduction
- Faster degree elevation
- Better choices for when to degree reduce, when to do high-resolution splitting
- Better split point selection
- Better choice of initial Möbius transformation
- More choices of interval Bernstein polynomial representation types (between GMP and native float): double-double? quad-double? fixed-precision multi-word arithmetic?
- Start with exact strategy and switch over to intervals
- Move more to Pyrex; local optimizations

Ideas for speed improvements

- Faster de Casteljau's algorithm
  - SSE2
  - Anatole Ruslanov's register tiling
- Partial degree reduction
- Faster degree reduction
- Faster degree elevation
- Better choices for when to degree reduce, when to do high-resolution splitting
- Better split point selection
- Better choice of initial Möbius transformation
- More choices of interval Bernstein polynomial representation types (between GMP and native float): double-double? quad-double? fixed-precision multi-word arithmetic?
- Start with exact strategy and switch over to intervals
- Move more to Pyrex; local optimizations

Ideas for speed improvements

- Faster de Casteljau's algorithm
  - SSE2
  - Anatole Ruslanov's register tiling
- Partial degree reduction
- Faster degree reduction
- Faster degree elevation
- Better choices for when to degree reduce, when to do high-resolution splitting
- Better split point selection
- Better choice of initial Möbius transformation
- More choices of interval Bernstein polynomial representation types (between GMP and native float): double-double? quad-double? fixed-precision multi-word arithmetic?
- Start with exact strategy and switch over to intervals
- Move more to Pyrex; local optimizations

Ideas for speed improvements

- Faster de Casteljau's algorithm
  - SSE2
  - Anatole Ruslanov's register tiling
- Partial degree reduction
- Faster degree reduction
- Faster degree elevation
- Better choices for when to degree reduce, when to do high-resolution splitting
- Better split point selection
- Better choice of initial Möbius transformation
- More choices of interval Bernstein polynomial representation types (between GMP and native float): double-double? quad-double? fixed-precision multi-word arithmetic?
- Start with exact strategy and switch over to intervals
- Move more to Pyrex; local optimizations

Ideas for speed improvements

- Faster de Casteljau's algorithm
  - SSE2
  - Anatole Ruslanov's register tiling
- Partial degree reduction
- Faster degree reduction
- Faster degree elevation
- Better choices for when to degree reduce, when to do high-resolution splitting
- Better split point selection
- Better choice of initial Möbius transformation
- More choices of interval Bernstein polynomial representation types (between GMP and native float): double-double? quad-double? fixed-precision multi-word arithmetic?
- Start with exact strategy and switch over to intervals
- Move more to Pyrex; local optimizations

Interface improvements

- Allow input polynomial coefficients to be intervals, or computable reals (like the algebraic reals)
- Specify a minimal width: once the intervals are this narrow, stop even if roots are not isolated
- Specify a maximum width: keep going after you've isolated a root until intervals are this narrow (to actually compute the roots, instead of just isolate them)
- Provide a floating-point, inexact version of the algorithm

Outline

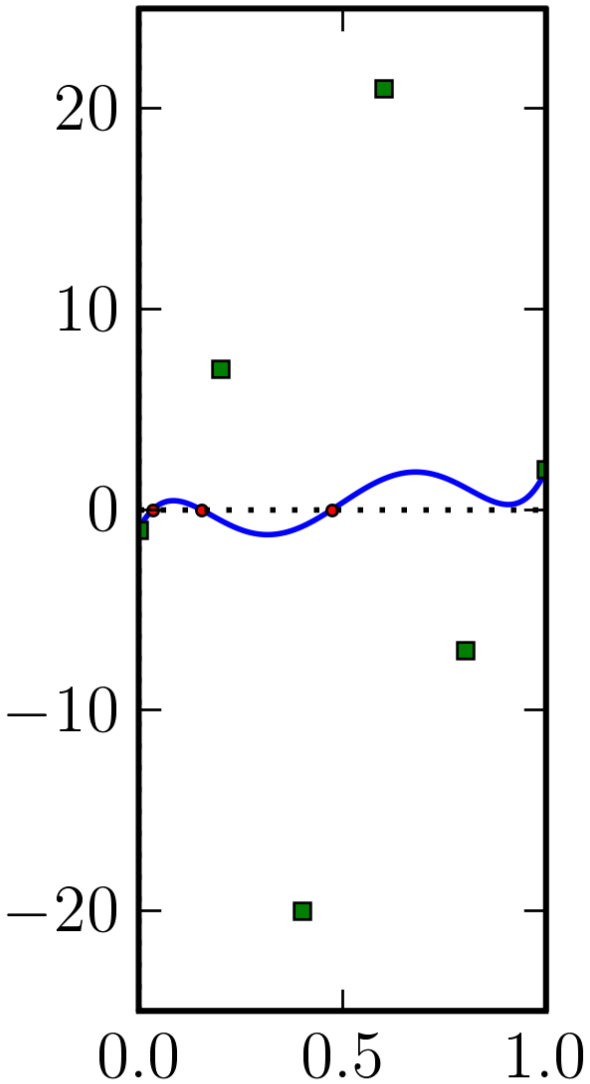
Summary

- Interval Bernstein polynomials (with careful precision control) and degree reduction are both very important for this problem.
- Experimental results indicate that my algorithm is asymptotically faster than other implementations I could find; I conjecture this is true.
- Lots more improvements to make—someone please take over!

Outline

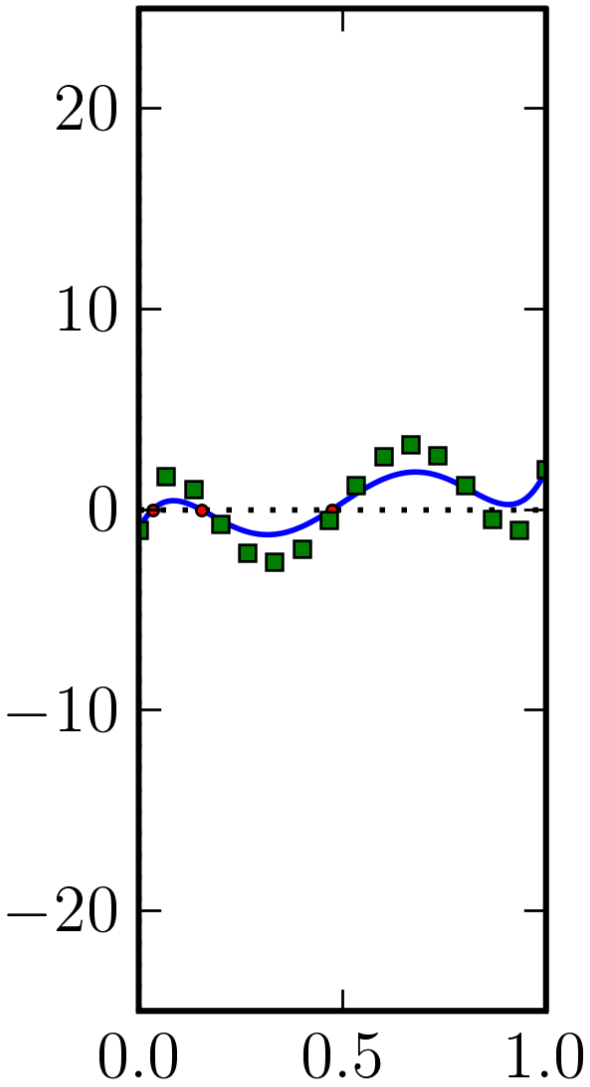
Outline

Taylor expansion and low-degree approximations



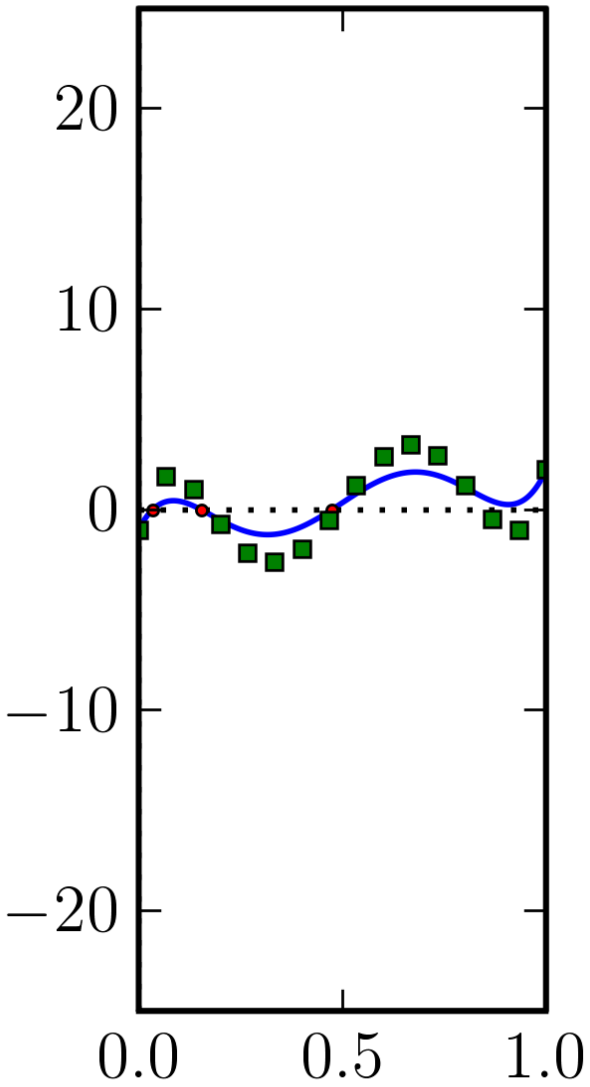
- Remember our plot of $[-1, 7, -20, 21, -7, 2]$ with its Bernstein coefficients.
- If we use the degree-15 Bernstein basis instead of the degree-5 Bernstein basis, the graph of the coefficients matches the graph of the polynomial much more closely.
- Also, these degree-15 coefficients all lie on a degree-5 polynomial.

Taylor expansion and low-degree approximations



- Remember our plot of $[-1, 7, -20, 21, -7, 2]$ with its Bernstein coefficients.
- If we use the degree-15 Bernstein basis instead of the degree-5 Bernstein basis, the graph of the coefficients matches the graph of the polynomial much more closely.
- Also, these degree-15 coefficients all lie on a degree-5 polynomial.

Taylor expansion and low-degree approximations



- Remember our plot of $[-1, 7, -20, 21, -7, 2]$ with its Bernstein coefficients.
- If we use the degree-15 Bernstein basis instead of the degree-5 Bernstein basis, the graph of the coefficients matches the graph of the polynomial much more closely.
- Also, these degree-15 coefficients all lie on a degree-5 polynomial.