Plotting in a Formally Verified Way

Guillaume Melquiond

Université Paris-Saclay, CNRS, ENS Paris-Saclay, Inria
Laboratoire Méthodes Formelles

May 24, 2021

Implementing a Mathematical Library

Traditional approach: the case of $\exp x$

1. Argument reduction: $x \simeq t + (k/N) \log 2$ with $k \in \mathbb{Z}$.
2. Polynomial approx: $\exp t \simeq \sum_i p_i \cdot t^i$ for $|t| \leq \log 2/(2N)$.
3. Result reconstruction: $\exp x \simeq 2^{k/N} \cdot \exp t$.

Implementing a Mathematical Library

Traditional approach: the case of $\exp x$

1. Argument reduction: $x \simeq t + (k/N)\log 2$ with $k \in \mathbb{Z}$.
2. Polynomial approx: $\exp t \simeq \sum_i p_i \cdot t^i$ for $|t| \le \log 2/(2N)$.
3. Result reconstruction: $\exp x \simeq 2^{k/N} \cdot \exp t$.

The GNU libc implementation

$$
\begin{array}{lcll}
p_0 & = & 1 + c_k & \text{with } 0 \le k < N = 128 \\
p_1 & = & 1 & \\
p_2 & = & \texttt{0x1.fffffffffdbdp} - 2 & \not\simeq 1/2 \\
p_3 & = & \texttt{0x1.555555555543cp} - 3 & \not\simeq 1/6 \\
p_4 & = & \texttt{0x1.55555cf172b91p} - 5 & \not\simeq 1/24 \\
p_5 & = & \texttt{0x1.1111167a4d017p} - 7 & \not\simeq 1/120
\end{array}
$$

Implementing a Mathematical Library

Traditional approach: the case of $\exp x$

1. Argument reduction: $x \simeq t + (k/N) \log 2$ with $k \in \mathbb{Z}$.
2. Polynomial approx: $\exp t \simeq \sum_i p_i \cdot t^i$ for $|t| \leq \log 2/(2N)$.
3. Result reconstruction: $\exp x \simeq 2^{k/N} \cdot \exp t$.

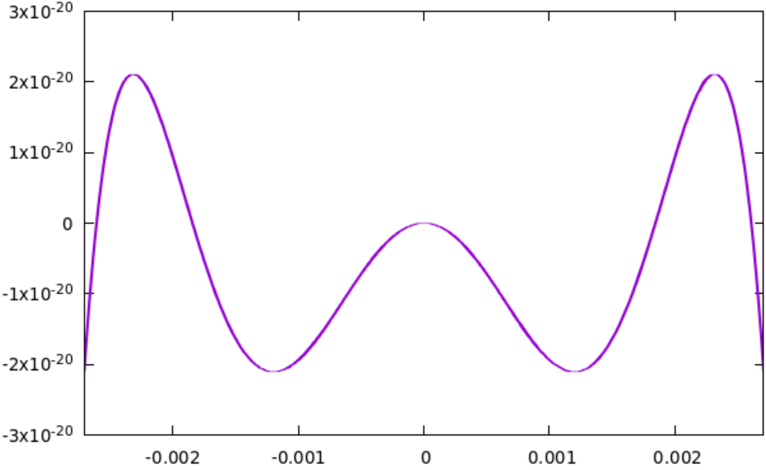
The GNU libc implementation

$$
\begin{array}{lcll}
p_0 & = & 1 + c_k & \text{with } 0 \leq k < N = 128 \\
p_1 & = & 1 & \\
p_2 & = & \texttt{0x1.fffffffffdbdp}-2 & \not\simeq 1/2 \\
p_3 & = & \texttt{0x1.555555555543cp}-3 & \not\simeq 1/6 \\
p_4 & = & \texttt{0x1.55555cf172b91p}-5 & \not\simeq 1/24 \\
p_5 & = & \texttt{0x1.1111167a4d017p}-7 & \not\simeq 1/120
\end{array}
$$

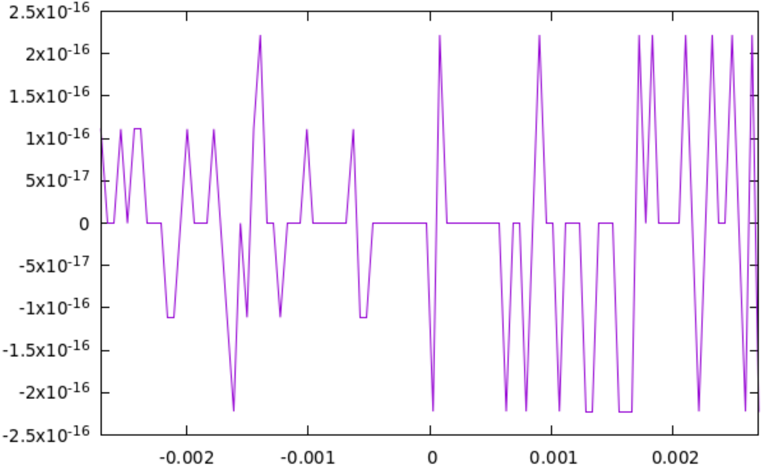
How to check the quality of a polynomial?

By plotting the error between the polynomial and the function.

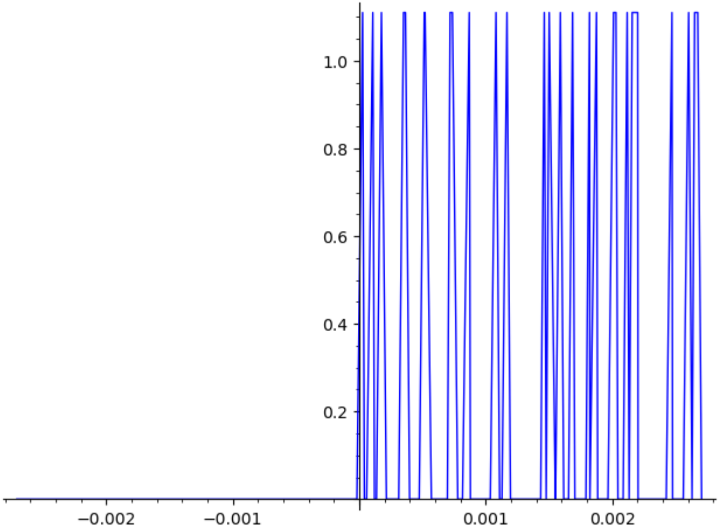
Plotting the Error: What the Developer Hopes for



Plotting the Error: What the Developer Gets (Gnuplot)

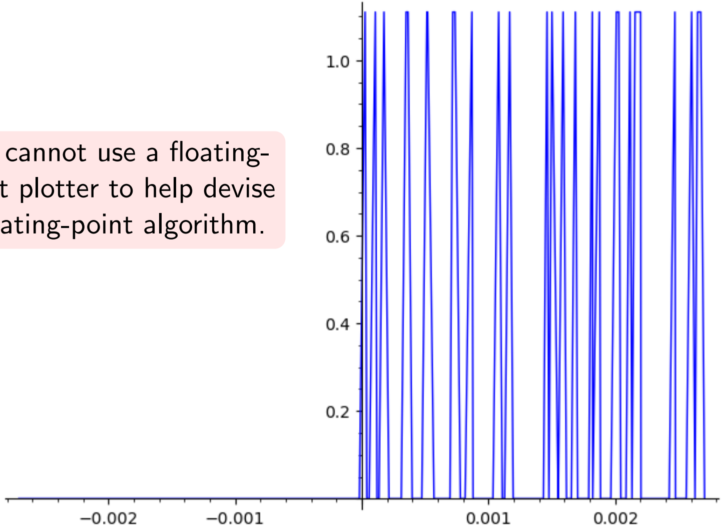


Plotting the Error: What the Developer Gets (SageMath)



Plotting the Error: What the Developer Gets (SageMath)

One cannot use a floating-point plotter to help devise a floating-point algorithm.

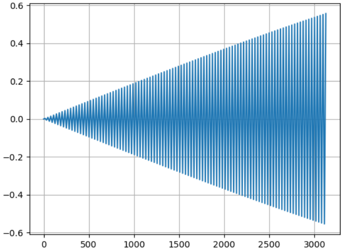


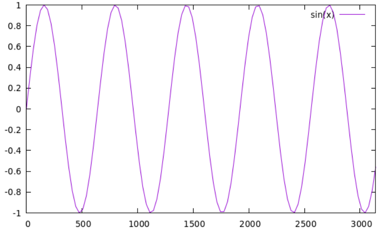
Accuracy is not the Only Issue

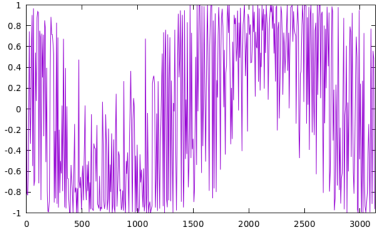
$\sin(x)$ for $x \in [0; 3141]$

How to sample?

- Gnuplot: 150 points
- Matplotlib: 200 points
- Sollya: 501 points + noise







Introduction

Current situation

When plotting, computer algebra systems might fall short.

- Accuracy might not be sufficient.
- Sampling might miss some features.
- Bugs might occur.

Introduction

Current situation

When plotting, computer algebra systems might fall short.

- Accuracy might not be sufficient.
- Sampling might miss some features.
- Bugs might occur.

Objectives

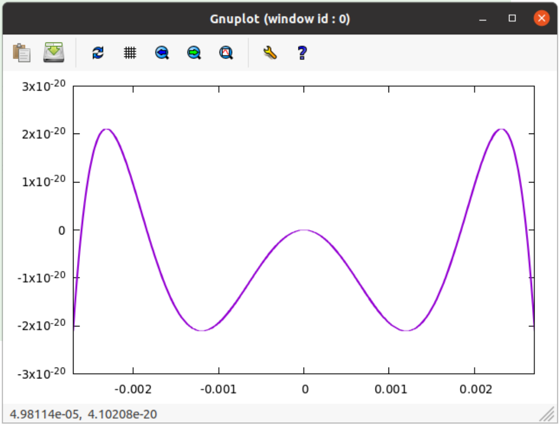
1. Formally characterize what a correct plot is.
2. Devise an efficient algorithm and formally verify it.
3. Execute it inside the logic of the Coq proof assistant.
4. Try to make it usable.

Plotting with Coq

```
From Coq Require Import Reals.
From Interval Require Import Plot Tactic.
Open Scope R_scope.

Definition g := ltac:(plot
  (fun x => (1 + x +
    9007199254740413 * powerRZ 2 (-54) * x^2 +
    1501199875790095 * powerRZ 2 (-53) * x^3 +
    6004801545907089 * powerRZ 2 (-57) * x^4 +
    4803841055051799 * powerRZ 2 (-59) * x^5)
   - exp x)
  (-ln 2 / 256) (ln 2 / 256)     (* domain    *)
  with (i_prec 90)).             (* precision *)

Plot g.
```

Plotting with Coq

```
From Coq Require Import Reals.
From Interval Require Import Plot Tactic.
Open Scope R_scope.

Definition g := ltac:
  (fun x => (1 + x +
    9007199254740413
    1501199875790095
    6004801545907089
    4803841055051799
   - exp x)
  (-ln 2 / 256) (ln 2
  with (i_prec 90)).

Plot g.
```



Outline

Outline

Some Concepts

A function plot is. . .

- correct if blank pixels are not traversed by the function graph;
- complete if filled pixels are traversed by the function graph.

Some Concepts

A function plot is. . .

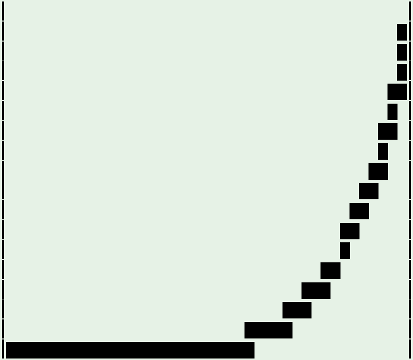
- correct if blank pixels are not traversed by the function graph;
- complete if filled pixels are traversed by the function graph.

We will guarantee correctness and strive for completeness.

Some Preexisting Work

O'Connor, 2008, A computer verified theory of compact sets

```
ball (m:=Compact stableQ2) (324 # 2592)
  graphCR (boundBelow 0 ∘ boundAbove 3 ∘ exp_bound_uc 1) [- (6) .. 1]
  (Cunit
     (- (6), 3)⇱
```



```
                                                              ⇲(1, 0))
```

Coq Formalization of Correctness

Plots as lists of intervals

$\forall x \in X_i,\ f(x) \in \ell_i$ with $X_i = [ox + dx \cdot i; ox + dx \cdot (i+1)]$.

Internal definition

```
Definition plot1 (f : R -> R) (ox dx : R)
                 (l : list I.type) :=
  forall i x, ox + dx * i <= x <= ox + dx * (i+1) ->
  I.contains (nth i l I.nai) (f x).
```

Coq Formalization of Correctness

Plots as lists of intervals

$\forall x \in X_i,\ f(x) \in \ell_i$ with $X_i = [ox + dx \cdot i; ox + dx \cdot (i+1)]$.

Internal definition

```
Definition plot1 (f : R -> R) (ox dx : R)
                 (l : list I.type) :=
  forall i x, ox + dx * i <= x <= ox + dx * (i+1) ->
  I.contains (nth i l I.nai) (f x).
```

Interface with the outer world

```
Definition plot2 (f : R -> R) (ox dx oy dy : R)
                 (h : Z) (l : list (Z * Z)) :=
  forall i x, ox + dx * i <= x <= ox + dx * (i+1) ->
  oy <= f x <= oy + dy * h ->
  let r := nth i l (0, h) in
  oy + dy * (fst r) <= f x <= oy + dy * (snd r).
```

General Process

1. Reify ox, dx, and f.
2. Reify oy and dy, or compute tentative values by sampling f.
3. Compute a list ℓ of intervals that satisfies `plot1`.
4. Compute oy and dy, if not reified.
5. Convert ℓ to a list that satisfy `plot2`.

General Process

1. Reify ox, dx, and f.
2. Reify oy and dy, or compute tentative values by sampling f.
3. Compute a list ℓ of intervals that satisfies `plot1`.
4. Compute oy and dy, if not reified.
5. Convert ℓ to a list that satisfy `plot2`.

Plot of $x \mapsto x^2$ between 0 and 1, resolution 10×100

A proof term of type

```
plot2 (fun x => x^2) 0 (820/8192)
      (-5/16384) (665/65536) 100
  ((0, 2) :: (0, 5) :: (3, 9) :: (8, 16) :: (15, 25)
     :: (24, 36) :: (35, 49) :: (48, 64) :: (62, 81)
     :: (79, 100) :: nil)
```

Outline

1. Introduction
2. Plot correctness
3. Computing the graph
   - Definite integrals
   - Interval arithmetic
   - Polynomials
4. Conclusion

Plotting is no Harder than Integrating

How to integrate f between u and v?

1. Split $[u; v]$ into smaller subintervals W_k.
2. Compute a polynomial approximation (p_k, Δ_k) of f over W_k:
$$\forall x \in W_k,\ \ p_k(x) - f(x) \in \Delta_k.$$
3. If Δ_k is not thin enough, go back to step 1.
4. Integrate p_k over W_k.

Plotting is no Harder than Integrating

How to integrate f between u and v?

1. Split $[u; v]$ into smaller subintervals W_k.
2. Compute a polynomial approximation (p_k, Δ_k) of f over W_k:
$$\forall x \in W_k,\ p_k(x) - f(x) \in \Delta_k.$$
3. If Δ_k is not thin enough, go back to step 1.
4. Integrate p_k over W_k.

How to plot f between u and v?

4. Plot p_k over W_k, one horizontal pixel at a time.

How to Plot a Polynomial?

Interval arithmetic

1. Define interval operators such that
   $\forall U, V \in \mathbb{I},\ \forall u, v \in \mathbb{R},\ u \in U \wedge v \in V \Rightarrow u \diamond v \in U \diamond V.$
2. Compose them to compute $Y_i = p_k(X_i) + \Delta_k$.
3. Deduce $\forall x \in X_i,\ f(x) \in Y_i$ from $X_i \subseteq W_k$.

How to Plot a Polynomial?

Interval arithmetic

1. Define interval operators such that
   $\forall U, V \in \mathbb{I},\ \forall u, v \in \mathbb{R},\ u \in U \wedge v \in V \Rightarrow u \diamond v \in U \diamond V.$
2. Compose them to compute $Y_i = p_k(X_i) + \Delta_k$.
3. Deduce $\forall x \in X_i,\ f(x) \in Y_i$ from $X_i \subseteq W_k$.

Could we have directly computed $f(X_i)$?

The resulting plot would have been useless,
because of the dependency effect of interval arithmetic.

With $f(x) = (1 + x + ...) - \exp x$ and $W = [0.002697; 0.002708]$,

$$\begin{aligned} f(W) &\rightsquigarrow [-1.06 \cdot 10^{-5}; 1.06 \cdot 10^{-5}], \\ p(W) + \Delta &\rightsquigarrow [-2.11 \cdot 10^{-20}; -1.83 \cdot 10^{-20}]. \end{aligned}$$

How to Plot a Polynomial?

Interval arithmetic

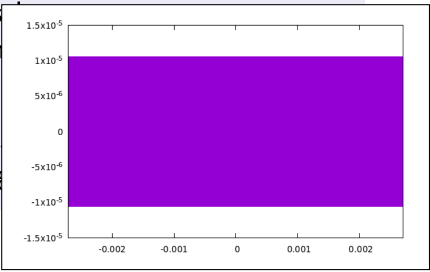
1. Define interval operators such that
   $\forall U, V \in \mathbb{I},\ \forall u, v \in \mathbb{R},\ u \in U \wedge v \in V \Rightarrow u \diamond v \in U \diamond V.$
2. Compose them to compute $Y_i = p_k(X_i) + \Delta_k$.
3. Deduce $\forall x \in X_i,\ f(x) \in Y_i$ from $X_i \subseteq W_k$.

Could we have directly computed $f(X_i)$?

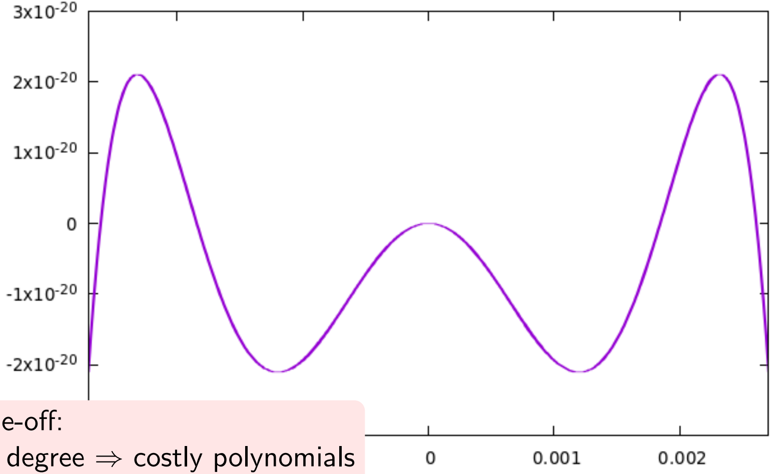
The resulting plot would have been us
because of the dependency effect of in

With $f(x) = (1 + x + \ldots) - \exp x$ and

$$f(W) \rightsquigarrow [-1.06 \cdot 10^{-5}; 1.06$$
$$p(W) + \Delta \rightsquigarrow [-2.11 \cdot 10^{-20}; -1.$$



Plotting Using Polynomials



Trade-off:
high degree ⇒ costly polynomials
low degree ⇒ lots of polynomials

Outline

User Interface

Tactic `plot`

Produce a proof term whose type denotes a correct plot.

`plot` *f* *x1* *x2* [*y1* *y2*] [`with` *options*]

Main options:

- output resolution (default: 512×384),
- degree of polynomials (default: 10),
- precision of computations (default: machine numbers).

Command `Plot`

Convert the type of a given term into a Gnuplot file and open it.

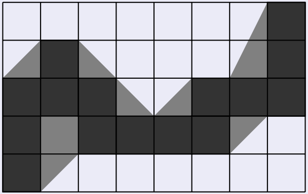
`Plot` *p* [`as "`*file*`"`].

Raster or Vector Graphics?

The issue with bitmaps

It does not look that good in practice,
especially when zoomed in.

Solution: Pessimize the plot to make it vector-based



Conclusion and Perspectives

This work

- A formally verified plotting algorithm, run inside Coq.
- Correct plots, but not always complete.
- Fast enough to be usable in practice.

Conclusion and Perspectives

This work

- A formally verified plotting algorithm, run inside Coq.
- Correct plots, but not always complete.
- Fast enough to be usable in practice.

Long-term goal

Turn Coq into a computer algebra system.

CoqInterval

`https://coqinterval.gitlabpages.inria.fr/`