



**HAL**  
open science

## A certificate-based approach to formally verified approximations

Florent Bréhard, Assia Mahboubi, Damien Pous

► **To cite this version:**

Florent Bréhard, Assia Mahboubi, Damien Pous. A certificate-based approach to formally verified approximations. ITP 2019 - Tenth International Conference on Interactive Theorem Proving, Sep 2019, Portland, United States. pp.1-19. hal-02088529v1

**HAL Id: hal-02088529**

**<https://laas.hal.science/hal-02088529v1>**

Submitted on 3 Apr 2019 (v1), last revised 1 Jul 2019 (v2)

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

# A certificate-based approach to formally verified approximations

Florent Bréhard, Assia Mahboubi, Damien Pous

LIP, LAAS, CNRS, Inria

## Abstract

We present a library to verify rigorous approximations of univariate functions on real numbers, with the Coq proof assistant. Based on interval arithmetic, this library also implements a technique of validation *a posteriori* based on the Banach fixed-point theorem. We illustrate this technique on the case of operations of division and square root. This library features a collection of abstract structures that organise the specification of rigorous approximations, and modularise the related proofs. Finally, we provide an implementation of verified Chebyshev approximations, and we discuss a few examples of computations.

**2012 ACM Subject Classification** General and reference → General literature; General and reference

**Keywords and phrases** approximation theory, Chebyshev polynomials, Banach fixed-point theorem, interval arithmetic, Coq

## 1 Introduction

While numerical analysis offers sophisticated computational methods to solve various function space problems, the numerical errors caused by floating-point computations, discretisations or finite iterations, are a major concern in domains like safety-critical engineering or computer assisted proofs in mathematics. To address these issues, *rigorous numerics* [34] provides algorithms to compute validated enclosures of the exact solution. However, their correctness is ensured by pen-and-paper mathematical proofs. In particular, there is no guarantee concerning their *implementations*.

In this regard, formal proof offers the highest level of confidence. Several noteworthy works use formally proved rigorous numerics to completely formalise highly nontrivial mathematical results, like the Flyspeck project [16] for the Kepler conjecture or the formal verification [19] of the computer-aided proof of the Lorenz attractor [33]. However, those methods often require intensive computations, which rapidly becomes restrictive inside proof assistants. In the context of formal verification, certificate-based methods is an appealing strategy [1]. It consists in discharging part of the computation work load to external oracles, while correctness remains guaranteed via *a posteriori* validation steps performed inside the proof assistant. This approach has mostly been used for the purpose of verifying symbolic computations, e.g. primality proofs [15], but we illustrate here how it can also be used in the context of rigorous numerical analysis.

**Interval arithmetic.** Invented in the 60s by Moore [29] and significantly developed in the 80s by Kulisch *et al.*, interval arithmetic is an essential building block of rigorous numerics. The key idea consists in using real intervals with representable endpoints (e.g., floating-point numbers) as rigorous enclosures of real numbers, and providing operations preserving correctness. For example, from  $\pi \in [3.1415, 3.1416]$  and  $e \in [2.7182, 2.7183]$ , one obtains  $\pi + e \in [3.1415, 3.1416] \oplus [2.7182, 2.7183] = [5.8597, 5.8599]$ . Efficient implementations are available, as MPFI [30], INTLAB [31], C-XSC [24], ARB [20]. The COQINTERVAL library [28] moreover provides a fully verified implementation inside the COQ proof assistant.

**Rigorous Chebyshev approximations.** Interval arithmetic is however not a panacea, and replacing all operations on real numbers by interval ones should always be considered with caution: the *dependency phenomenon* may lead to disastrous over-approximations. In

46 such cases, *higher order methods* such as rigorous polynomial approximations (RPAs) are  
 47 preferable. A pioneer work is that of Berz and Makino on *Taylor models* [4]. Those provide  
 48 not only a polynomial, but also a *remainder* s.t. the latter contains the difference between the  
 49 former and the represented function. Since then, efforts were made to clarify the definition  
 50 of RPAs and extend them to other bases, in particular the Chebyshev basis [10, 22], due to  
 51 their far better approximation properties than Taylor expansions [32].

52 On the formal proof side, the COQINTERVAL library includes an implementation of Taylor  
 53 models called COQAPPROX [27], allowing in particular for an automated rigorous evaluation  
 54 procedure of definite integrals inside COQ [26]. Unfortunately, an equally accomplished  
 55 equivalent with Chebyshev approximations does not exist now. Our contribution is a first step  
 56 towards a formally proved counterpart of the popular CHEBFUN package [13] for MATLAB.

57 **Fixed-point based *a posteriori* validation.** Some operations in function spaces admit  
 58 straightforward *self-validating* algorithms by replacing all operations in  $\mathbb{R}$  by interval ones.  
 59 Unfortunately, more complicated operations (e.g., division, square root, differential equations)  
 60 face several obstructions: the intervals may fail to give sufficiently tight enclosures, bounds for  
 61 the remainders may be unknown, or only asymptotic, or depend on noneffective quantities.

62 In such cases, *a posteriori* validation techniques are an attractive alternative, widely  
 63 used in rigorous numerics. They consist in reconstructing afterwards an error bound for  
 64 a candidate approximation. Dating back from the works of Kantorovich about Newton’s  
 65 method, they gained prominence with the rise of modern computers and were applied to  
 66 numerous functional analysis problems [23, 36, 35, 25]. Even more recently, those methods  
 67 were used to compute RPAs for solutions of linear ODEs [2, 8]. Broadly speaking, the  
 68 function of interest is characterised as a fixed-point of a contracting operator, from which  
 69 an error bound is recovered thanks to the Banach fixed-point theorem [3, Thm. 2.1]. Such  
 70 techniques are of special interest for formal verification, for they allow one to rely on efficient  
 71 but untrusted external tools while keeping the trusted codebase small: it suffices to formalise  
 72 the theory about contracting operators and provide means of computing with those operators.

73 **Contributions and outline.** We present a COQ library that makes it possible to compute  
 74 rigorous Chebyshev approximations of functions on reals. We support basic operations like  
 75 multiplication or integration in the standard way. For more complex operations like division  
 76 and square root, we resort to *a posteriori* validation techniques, thus making a first step  
 77 towards a potential cooperation between external numerical tools and COQ.

78 We use the interval arithmetic provided by COQINTERVAL, but we design our abstractions  
 79 for RPAs from scratch: this allows us to experiment with different design choices, with more  
 80 flexibility. We first describe the main lines of the hierarchy (Section 2): we rely on canonical  
 81 structures to abstract over the concrete implementation details of interval arithmetic, and  
 82 we use them to denote both real valued functions and their rigorous approximations. We  
 83 also abstract away from the concrete basis for approximations, to work in the future with  
 84 different bases, even non polynomial ones (e.g., Bessel functions). We provide instances for  
 85 the monomial and Chebyshev bases, the latter being described in Section 3.

86 The main theorem we need to perform a posteriori validation is the Banach fixed-point  
 87 theorem, whose formalisation is described in Section 4. We show in Section 5 how to apply  
 88 this theorem to compute rigorous approximations for division and square root using Newton-  
 89 like operators. We finally discuss the benefits of our approach on two examples (Section 6):  
 90 RPAs for the absolute value function, and verified computation of integrals related to the  
 91 second part of Hilbert’s 16th problem.

## 2 Approximating real numbers and functions

Numerical errors come from the estimation of both *real numbers*, e.g. using floating-point numbers, and *real functions*, e.g. using polynomials. Rigorous estimations must take all these uncertainties into account. For this purpose, *interval arithmetic* provides an explicit enclosure and *rigorous polynomial approximations* attach an interval to a polynomial approximant, which bounds the method error on a given domain. Note that the coefficients of polynomial approximations are usually themselves obtained from evaluations of the function or of its derivatives, and therefore also subject to numerical errors. A formal library about rigorous approximation thus implements several variants of each operations, on real numbers, floats, intervals, mathematical functions, approximants, etc., whose relationships are made precise in the various layers of specifications. Our library features a small hierarchy of structures which formalises and organises the dependencies between these variants.

### 2.1 Reals and Intervals

At the bottom of the hierarchy, structure `Ops0` collects the operations available on reals, floats, intervals, but also on polynomials and rigorous approximations. It provides the signature of a ring structure, with symbols `+`, `-`, `*`, `1` and `0` shared by all instances thanks to COQ's system of canonical structures. Yet the ring equational theory is a priori only available for real numbers. These operations are also those trivially self-validating. A super-structure `Ops1` collects other operations required on data-structures used for scalars: reals, floats, interval endpoints, intervals, etc. They are not meant to be implemented on polynomial approximations.

```
Record Ops0 := {
  car:> Type;
  add: car → car → car;
  sub: car → car → car;
  mul: car → car → car;
  zer, one: car }.

Record Ops1 := {
  ops0:> Ops0;
  fromZ: Z → ops0;
  div: ops0 → ops0 → ops0;
  sqrt, cos, abs: ops0 → ops0;
  pi: ops0 }.
```

Structure `Rel0` specifies the relationship between the operations of `Ops0` on reals and those on intervals. The field `rel` is a relation between the two instances `C` and `D`, which share overloaded notations. The relation will eventually be instantiated with the containment relation between intervals and reals. When doing so, the requirements on the relation precisely correspond to the fact that interval operations properly approximate real operations. A record `Rel1` is defined in the very same way for `Ops1`.

```
Record Rel0 (C D: Ops0) := {
  rel:> C → D → Prop;
  radd: ∀ x y, rel x y → ∀ x' y', rel x' y' → rel (x+x') (y+y');
  rsub: ∀ x y, rel x y → ∀ x' y', rel x' y' → rel (x-x') (y-y');
  rmul: ∀ x y, rel x y → ∀ x' y', rel x' y' → rel (x*x') (y*y');
  rzer: rel 0 0;
  rone: rel 1 1 }.
```

As much as possible, we will work with polymorphic functions like the following one:

```
Definition f (C: Ops1) (x: C): C := 1 / (1 + sqrt x).
```

First of all, this allows us to define at once a function on real numbers (here,  $x \mapsto \frac{1}{1+\sqrt{x}}$ ) and a function on intervals, whatever the implementation of intervals. Second, and even more importantly, the corresponding approximation correctness theorem will always hold—by a parametricity meta-result, such a function  $f$  will always satisfy the following lemma:

```

137
138 Lemma rf:  $\forall C D (T: \text{Rel1 } C D), \forall x y, T x y \rightarrow T (f x) (f y)$ .
139

```

140 This is only a meta-result: we need to provide a proof for each function  $f$ ; but the proof is  
 141 always trivial, and we automatise it.

142 There are however operations which cannot be implemented at this level of abstraction,  
 143 even if we were to add some operations to the record `Ops1`. This is typically the case for  
 144 division and square root of rigorous approximations, which require operations on intervals  
 145 that do not make sense on real numbers (e.g., computing the range of a function and checking  
 146 that it is bounded). In order to define those operations while remaining rather agnostic  
 147 about the choice of interval implementation, we setup an intermediate layer of abstraction  
 148 using the structure `NBH` (for neighbourhood):

```

149
150 Record NBH := {
151   II:> Ops1; (* abstract intervals *)
152   contains: Rel1 II ROps1; (* containment relation; ROps1 is the Ops1 instance on R *)
153   convex:  $\forall Z x y, \text{contains } Z x \rightarrow \text{contains } Z y \rightarrow \forall z, x \leq z \leq y \rightarrow \text{contains } Z z$ ;
154   (* additional operations on intervals *)
155   bnd: II  $\rightarrow$  II  $\rightarrow$  II; (* directed convex hull *)
156   is_lt: II  $\rightarrow$  II  $\rightarrow$  bool; (* strict above test *)
157   min,max: II  $\rightarrow$  option II; (* min, max, if any *)
158   bot: II; (* uninformative, contains all reals *)
159   (* specification of the above operations *)
160   bndE:  $\forall X x, \text{contains } X x \rightarrow \forall Y y, \text{contains } Y y \rightarrow \forall z, x \leq z \leq y \rightarrow \text{contains } (\text{bnd } X Y) z$ ;
161   is_ltE:  $\forall X Y, \text{wreflect } (\forall x y, \text{contains } X x \rightarrow \text{contains } Y y \rightarrow x < y) (\text{is\_lt } X Y)$ ;
162   minE, maxE, botE: ... }.

```

164 We will also make use of the two following derived operations:

```

165
166 Definition mag (N: NBH) (X: II): option II := max (abs X).
167 Definition sym (N: NBH) (X: II): II := let X := abs X in bnd (-X) X.

```

169 The first one approximates the magnitude as an interval, if possible; the second one returns  
 170 an interval centered in 0 that contains the argument. Note that we assume that intervals  
 171 are convex. We provide an instance of this structure using the `COQINTERVAL` library, using  
 172 intervals of floating point numbers from the `FLOCC` library [6]. It is actually a family of  
 173 instances indexed by the desired precision.

## 174 2.2 Abstract functions

175 The structure `FunOps` describes inductively the catalogue of expressions that the library can  
 176 approximate.

```

177
178 Record FunOps (C: Type) := {
179   funcar:> Ops0; (* abstract type for functions, and pointwise basic operations *)
180   id: funcar;
181   cst: C  $\rightarrow$  funcar;
182   eval: funcar  $\rightarrow$  C  $\rightarrow$  C;
183   integrate: funcar  $\rightarrow$  C  $\rightarrow$  C  $\rightarrow$  C;
184   div': nat  $\rightarrow$  funcar  $\rightarrow$  funcar  $\rightarrow$  funcar;
185   sqrt': nat  $\rightarrow$  funcar  $\rightarrow$  funcar }.

```

187 It is parameterised by a type `C` of ground values (typically, reals or intervals); it packages a  
 188 set of basic operations on some abstract type for functions (pointwise addition, multiplica-  
 189 tion...), together with operations specific to functions: identity, constant function, evaluation,  
 190 integration. It also asks for division and square root operations; those have an additional  
 191 argument which is used to pass parameters to the oracles used in the implementation of  
 192 those operations (for now, the degree of the interpolants).

When  $C = \mathbb{R}$  this structure is instantiated with the standard operations on  $\mathbb{R} \rightarrow \mathbb{R}$  (ignoring the extra parameters for division and square root); our main goal is to provide instances with intervals for  $C$ , with which it is possible to perform computations. Like for ground values, the structure `FunOps` makes it possible to write polymorphic functions like:

```

197 Definition g (C: Ops1) (F: FunOps C): F :=
198   let f: F := div' 33 1 (1 + sqrt' 33 id) in
199   let a: C := integrate f 0 1 in
200   pi + id * cst a
201
202

```

Such a declaration defines at the same time a function on reals ( $x \mapsto \pi + x \int_0^1 \frac{dt}{1+\sqrt{t}}$ ) and approximations of it, which will be obvious to prove correct whenever the chosen instance `F` satisfies appropriate properties. Those instances are obtained using *rigorous approximations*.

## 2.3 Rigorous Approximations

Approximating a function usually consists in projecting this function onto a finite dimension vector space, by expansion on a basis with appropriate properties. For instance, so-called Taylor models [4], are an instance of *rigorous polynomial approximation*. They attach an interval bounding the remainder to a certain polynomial, in this case represented in monomial basis, so as to describe a set of functions containing the one to be approximated. More generally in this section, a *rigorous approximations* refers to a linear combination of elements in a suitable basis, packaged with an interval remainder. In the code, we will also use the shorter term *models*, by analogy with Taylor models.

A basis is described by a family of functions, non necessarily polynomials, indexed by natural numbers, that is a term  $T: \text{nat} \rightarrow \mathbb{R} \rightarrow \mathbb{R}$ . The structure `BasisOps_on` below describes the signature required on a basis  $T$ . It is parameterised by the type  $C$  of coefficients; sequences of such coefficients (`seq C`) represent linear combinations of elements of  $T$ . Linear operations ( $+$ ,  $-$ ,  $0$ ) need not be provided since they can be implemented independently from the basis. The range operation is important: its role is to bound the range on the given domain; it should be as accurate as possible since it is used at many places to compute error bounds in rigorous approximations (e.g., for multiplication and a posteriori validation). We define `BasisOps` to be a polymorphic function so that we capture with a single object the idealised operations on reals and their concrete implementation with intervals.

```

225 Record BasisOps_on (C: Type) := {
226   lo, hi: C; (* bounds for the domain *)
227   beval: seq C → C → C; (* (efficient) evaluation *)
228   bmul: seq C → seq C → seq C; (* multiplication *)
229   bone, bid: seq C; (* constant to 1, identity *)
230   bprim: seq C → seq C; }. (* primitive *)
231   brange: seq C → C*C; }. (* range *)
232
233 Definition BasisOps := ∀ C: Ops1, BasisOps_on C.

```

Given such operations, we equip `seq C` with the basic operations in `Ops0`. Then we can define rigorous approximations:

```

237 Record Model C := { pol: seq C; rem: C }.
238
239

```

Like with `seq C`, we equip `Model C` with the basic operations in `Ops0`, and then with those from `FunOps`. For instance, addition, evaluation and integration are defined as follows:

```

242 Definition madd (C: Ops1) (M N: Model T C): Model T C :=
243   { | pol := pol M + pol N; rem := rem M + rem N | }.
244
245 Definition meval (C: Ops1) (M: Model T C) (X: C): C := beval (pol M) X + rem M.
246
247 Definition mintegrate (C: Ops1) (M: Model T C) (a b: C): C :=
248   let N := bprim (pol M) in beval N b - beval N a + (b-a)*rem M.

```

249 For those relatively simple operations, it suffices to have the basic operations (`Ops1`) on  $\mathbb{C}$ .  
 250 For other operations like the range of a model, we actually need the additional operations on  
 251 intervals provided by the structure `NBH`:

```
252 Definition mrange (N: NBH) (M: Model) :=  
253   let (a,b) := brange (pol M) in bnd a b + rem M.  
254
```

256 This is also the case for division and square root, which we will discuss in Section 5. All in  
 257 all, we obtain instances `FunOps` through a construction of the following type:

```
258 Canonical Structure MFunOps (N: NBH) (B: BasisOps): FunOps II.  
259 (* with carrier [Model II] *)  
260  
261
```

262 It finally remains to show that those operations defined on rigorous approximations  
 263 properly match the idealised operations on functions over reals. We fix in the sequel an  
 264 instance `N: NBH` of neighbourhood and basis operations `B: BasisOps`, and we write `Model` for  
 265 `Model II`). The central definition to establish this correspondence is the following one, where  
 266 the function `eval` is the obvious evaluation function for linear combinations of elements of  $\mathbb{T}$ .

```
267 Definition mcontains (F: Model) (f: R → R) :=  
268   ∃ p: seq R, scontains (pol F) p ∧ ∀ x, lo ≤ x ≤ hi → contains (rem F) (f x - eval T p x)  
269
```

271 Intuitively, a model contains a real-valued function  $f$  if it contains a generalised polynomial  
 272 which is close enough to  $f$  on the domain of the basis. (The binary predicate `scontains`  
 273 denotes the pointwise extension of the relation `contains` to sequences: in the definition, the  
 274 real coefficients of `p` should be pointwise contained in the interval coefficients of `pol F`.)

275 Equipped with this definition, we prove lemmas like

```
276 Lemma rmmul: ∀ F f G g, mcontains F f → mcontains G g → mcontains (F*G) (f*g).  
277 Lemma rmdiv: ∀ n F f G g, mcontains F f → mcontains G g → mcontains (div' n F G) (div'  
278   n f g).  
279 Lemma rmintegrate: ∀ F f A a B b, (∀ x, lo ≤ x ≤ hi → continuous_at f x) →  
280   mcontains F f → contains A a → contains B b → contains (integrate F A B) (integrate  
281   f a b).  
282
```

284 Of course, we need assumptions on the basis operations in order to do so. Those  
 285 assumptions are summarised in the following structure. Recall that a `B: BasisOps` provides  
 286 us with operations `B ROps1` on reals and operations `B II` on intervals. The structure assumes:  
 287 1/ the expected properties on the operations on reals (i.e, efficient evaluation corresponds  
 288 to evaluation with  $\mathbb{T}$ , multiplication indeed corresponds to pointwise multiplication under  
 289 evaluation, etc.); and 2/ a relationship between the operations on reals and on intervals. This  
 290 separation of concerns is very convenient: the latter containment lemmas are always proved  
 291 in a trivial way (i.e., automatically), and the former properties do not involve intervals at all,  
 292 but only real numbers and functions, for which usual mathematical intuitions apply.

```
293 Record ValidBasisOps (N: NBH) (B: BasisOps) := {  
294   (* properties of operations on reals (B ROps1) *)  
295   lohi: lo < hi;  
296   bevalE: ∀ p x, beval p x = eval T p x;  
297   eval_cont: ∀ p x, continuity_pt (eval T p) x;  
298   eval_mul: ∀ p q x, eval T (bmul p q) x = eval T p x * eval T q x;  
299   eval_prim: ∀ p a b, eval T (bprim p) b - eval T (bprim p) a = RInt (eval T p) a b;  
300   ...  
301   (* relationship between operations on intervals (B II) and on reals (B ROps1) *)  
302   rbeval: ∀ P p X x, scontains P p → contains X x → contains (beval P X) (beval p x);  
303   rbmul: ∀ P p Q q, scontains P p → scontains Q q → scontains (bmul P Q) (bmul p q);  
304   rbprim: ∀ P p, scontains P p → scontains (bprim P) (bprim p);  
305   ... }.  
306
```

### 3 Arithmetic on Chebyshev polynomials

In order to use the previously described rigorous approximations, it remains to provide implementation of operations (`BasisOps`) for certain families  $\mathsf{T}$  of functions. We provide two instances of them: one for the standard monomial basis, where  $\mathsf{T}_n x = x^n$ , and one described in this section for Chebyshev basis, where  $\mathsf{T}_n$  is the  $n$ -th Chebyshev polynomial.

Chebyshev polynomials are defined by the following recurrence, which immediately translates to a fixpoint definition in Coq.

$$T_0 = 1 \qquad T_1 = X \qquad T_{n+2} = 2XT_{n+1} - T_n$$

We can then prove simple properties of those polynomials, for instance:

$$T_n T_m = (T_{n+m} + T_{m-n})/2 \quad (n \leq m) \tag{1}$$

$$T_0 = T'_1 \quad T_1 = \frac{T'_2}{4} \quad T_{n+3} = \frac{T'_{n+3}}{2(n+3)} - \frac{T'_{n+1}}{2(n+1)} \tag{2}$$

$$T_n(\cos t) = \cos(nt) \tag{3}$$

Those are proved in a few lines using existing lemmas about derivation and cosine.

#### 3.1 Clenshaw's evaluation algorithm

The first operation we must implement for `BasisOps` is the evaluation function (`beval`). This operation should be polymorphic and as efficient as possible: it will be executed repeatedly when constructing and using rigorous approximations. We use Horner evaluation scheme for the monomial basis, and Clenshaw's algorithm [14] for Chebyshev, which are both linear in the number of elementary operations. The latter is usually presented as a dynamic programming routine. We translate it into a recursive function with two accumulators:

```

Fixpoint Clenshaw (C: Ops1) b c (p: seq C) x :=
  match p with
  | [] => c - x*b
  | a::q => Clenshaw c (a + 2*x*c - b) q x
  end.
Definition beval (C: Ops1) (p: seq C) x := Clenshaw 0 0 (rev p) x.

```

This code might look mysterious; it is justified by the following invariant on real numbers:

```

Lemma ClenshawR b c p x: Clenshaw b c p x = eval T (catrev p [c - 2*x*b; b]) x.

```

In the right-hand side, `catrev` is the function that reverses its first argument and catenate it with the second one. The proof is done by induction in just three lines, using the `COQ` tactic for ring equations. Correctness (i.e., field `bevalE` from structure `ValidBasisOps`) follows.

Note that while the definition of `beval` can be used with any `Ops1` structure, its correctness is proved only on reals: the lemma `ClenshawR` does not hold in every `Ops1` structure. The behaviour of `beval` on those structures is specified only through the fact that it respects containments (field `rbeval` from structure `ValidBasisOps`, which is proved automatically.)

#### 3.2 Multiplication

Another important operation is multiplication. Again, this operation should be polymorphic, and efficient. A difficulty here is that due to Equation (1), the  $n$ -th coefficient of a multiplication potentially depends on all coefficients of its arguments, not only on the coefficient of smaller rank. We use the following definition, with two auxiliary recursive functions

```

354
355 Fixpoint mul_pls (C: Ops1) (p q: seq C): seq C :=
356   match p,q with
357   | [],_ | _,[] => []
358   | a::p', b::q' => sadd (a*b::(sadd (sscal a q') (sscal b p'))) (0::0::mul_pls p' q')
359   end.
360 Fixpoint mul_mns (C: Ops1) (p q: seq C): seq C :=
361   match p,q with
362   | [],_ | _,[] => []
363   | a::p', b::q' => sadd (a*b::(sadd (sscal a q') (sscal b p'))) (mul_mns p' q')
364   end.
365 Definition smul C (p q: seq C): seq C := sscal (1/2) (sadd (mul_mns p q) (mul_pls p q))

```

(`sscal` is multiplication of a polynomial by a scalar, and `sadd` is addition of polynomials—we cannot yet use the standard notation for this operation since we are in the process of defining an `Ops0` structure on `seq C`.) The function `mul_pls` actually corresponds to multiplication in the monomial basis, it covers the first summand in the right-hand side of (1). The function `mul_mns` differs only in the fact that the recursive call is not pushed away using two ‘cons’ operations; it covers the second summand in the right-hand side of (1). Like previously, that `smul` preserves containments (field `rbmul` of structure `ValidBasisOps`) is obvious: this operation only performs a finite sequence of operations preserving containments. Proving that it behaves correctly on reals numbers is more interesting; the key invariant is the following one:

```

376
377 Lemma eval_mul_:  $\forall$  (p q: seq R) n x,
378   eval_ n p x * eval_ n q x = (eval (mul_mns p q) x + eval_ (n+n) (mul_pls p q) x)/2.

```

Here, `eval_ n p` evaluates `P` padded with `n` zeros in front of it. Again, the difficulty is to find the lemma: it is proved in six lines using (1), and correctness of `smul` on reals immediately follows. Taking primitives in Chebyshev basis follows the same pattern (see Appendix A).

### 3.3 Range

As mentioned above, we need accurate estimations of the range of a given polynomial in order to be able to compute precise rigorous approximations. This range can always be estimated by evaluating the polynomial on the interval representing the domain (i.e., given `p: seq C`, compute `beval p (bnd lo hi)`). This technique is however not sufficient in practice: this tends to produce largely over-estimated bounds. With Chebyshev basis we can proceed differently: indeed, thanks to Equation (3),  $T_n$  ranges over  $[-1; 1]$  on  $[-1; 1]$ . Therefore, the range of a polynomial on  $[-1; 1]$  can be estimated by using the sum of the absolute values of the coefficients in Chebyshev basis (and actually, we do not need to take the absolute value of the first coefficient since  $T_0 = 1$ ).

```

393
394 Definition range_ (C: Ops1): seq C  $\rightarrow$  C := foldr (fun A X => abs A + X) 0.
395 Definition range (C: Ops1) (P: seq C): C*C :=
396   match p with
397   | [] => (0,0)
398   | A::Q => let R := range_ Q in (A-R,A+R)
399   end.

```

### 3.4 Rescaling

Putting everything together, we obtain the polymorphic operations `chebyshev.basis: BasisOps`, which can readily be used to construct rigorous approximations, with the instance `MFunOps` from Section 2.3. This basis however requires to work on the domain  $[-1; 1]$  (for estimating the range as explained in the previous section, but also to perform interpolation, see Section 5.1). In order to use it on other domains, we provide a rescaling function that takes a `B: BasisOps`

407 and rescales it to a given interval  $[a; b]$  using the obvious affine function. We show that this  
408 operation preserves validity of basis operations, so that we can use it whenever needed.

#### 409 **4 Formalisation of Banach fixed-point theorem**

410 Banach fixed-point theorem is the cornerstone of the method discussed here.

411 **► Theorem 1** (Banach fixed-point). *Let  $(X, \|\cdot\|)$  be a Banach space, an operator  $F : X \rightarrow X$ ,*  
412  *$h^\circ \in X$ , and  $\mu, b, r \in \mathbb{R}_+$ , satisfying the following conditions:*

413 (1i)  $\|h^\circ - F \cdot h^\circ\| \leq b$ ;

414 (1ii)  $F$  is  $\mu$ -Lipschitz over the closed ball  $\overline{B}(h^\circ, r) := \{h \in X \mid \|h - h^\circ\| \leq r\}$ :

415  $\forall h_1, h_2 \in X, \quad h_1 \in \overline{B}(h^\circ, r) \wedge h_2 \in \overline{B}(h^\circ, r) \Rightarrow \|F \cdot h_1 - F \cdot h_2\| \leq \mu \|h_1 - h_2\|$ ;

416 (1iii)  $\mu < 1$  :  $F$  is contracting over  $\overline{B}(h^\circ, r)$ ;

417 (1iv)  $b + \mu r \leq r$ .

418 *Then  $F$  admits a unique fixed-point  $h^*$  in  $\overline{B}(h^\circ, r)$ .*

419 This classic result has been formalised in various flavours of logic and proof assistants. In  
420 particular, Boldo et al. have provided a formal proof of a version of this fixed-point theorem,  
421 based on the COQUELICOT library, for the purpose of the formalisation of the Lax-Milgram  
422 theorem [5]. Using the same backbone library, we provide a different statement that is more  
423 suitable for our effective validation purposes. We describe below its formalisation.

424 The COQUELICOT library formalises topological concepts using *filters* [7, 17], which we  
425 briefly recall here. A filter on a type  $T$  is a collection of collections of inhabitants of  $T$  which  
426 is non-empty, upward closed and stable under finite intersections:

```
427 Record Filter (T : Type) (F : (T → Prop) → Prop) := {
428   filter_true : F (fun _ => True) ;
429   filter_and : ∀ P Q : T → Prop, F P → F Q → F (fun x => P x /\ Q x) ;
430   filter_imp : ∀ P Q : T → Prop, (∀ x, P x → Q x) → F P → F Q }.
```

433 While filters are used to formalise neighbourhoods, *balls* allow for expressing the relative  
434 closeness of points in the space. Balls are formalised using a ternary relation between two  
435 points in the carrier type, and a real number, with the following axioms:

```
436 ball : M → ℝ → M → Prop ;
437 ax1 : ∀ x (e > 0), ball x e x ;
438 ax2 : ∀ x y e, ball x e y → ball y e x ;
439 ax3 : ∀ x y z e1 e2, ball x e1 y → ball y e2 z → ball x (e1 + e2) z
```

442 Two points are called *close* when they cannot be separated by balls:

```
443 Definition close (x y : M) : Prop := ∀ eps > 0, ball x eps y.
```

446 A filter is called a *Cauchy filter* when it contains balls of arbitrary (small) radius:

```
447 Definition cauchy (T : UniformSpace) (F : (T → Prop) → Prop) :=
448   ∀ eps > 0, ∃ x, F (ball x eps).
```

451 Finally, a *uniform space* is a type equipped with a ball relation and a *complete space* moreover  
452 has a limit operation on filters, which ensures the convergence of Cauchy sequences via the  
453 following axioms (where **ProperFilter**  $F$  is equivalent to **Filter**  $F \wedge \forall P, F P \rightarrow \exists x, P x$ ):

```
454 lim : ((T → Prop) → Prop) → T ;
455 ax1 : ∀ F, ProperFilter F → cauchy F → ∀ eps > 0, F (ball (lim F) eps) ;
456 ax2 : ∀ F1 F2, F1 ⊆ F2 → F2 ⊆ F1 → close (lim F1) (lim F2)
```

459 The above formal definition of balls does not enforce closedness nor openness. We thus intro-  
 460 duced the relation associated with the *closure* of balls, so as to model *closed* neighbourhoods:

461 **Definition** `cball`  $x\ r\ y := \forall e > 0, \text{ball } x\ (r+e)\ y$ .  
 462  
 463

464 Equipped with this definition, hypothesis (1 ii) of Theorem 1 is formalised as follows:

465 **Definition** `lipschitz_on`  $(F : U \rightarrow U)\ (\mu : \mathbb{R})\ (P : U \rightarrow \text{Prop}) :=$   
 466  $\forall x\ y : U, \forall r \geq 0, P\ x \rightarrow P\ y \rightarrow \text{cball } x\ r\ y \rightarrow \text{cball } (F\ x)\ (\mu * r)\ (F\ y)$ .  
 467  
 468

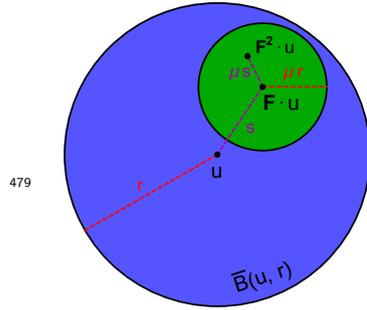
469 We now sketch our formalised proof, using mathematical notations. We consider a com-  
 470 plete space  $X$  and we write  $y \in B(x, r)$  for the formal  $(\text{ball } x\ r\ y)$ , and  $y \in \bar{B}(x, r)$  for  
 471  $(\text{cball } x\ r\ y)$ . The key notion is that of *strongly stable ball*:

472 ► **Definition 2** (Strongly stable ball). A ball  $\bar{B}(u, r)$  is  $\mu$ -strongly stable for  $F$  if  $F$  is  
 473  $\mu$ -Lipschitz on  $\bar{B}(u, r)$  and if there is a non-negative real number  $s$ , called the offset, s.t.:

$$474 \quad F \cdot u \in \bar{B}(u, s) \quad \text{and} \quad s + \mu r \leq r.$$

475 ► **Remark 3** (Stability). For any  $x$  in  $\bar{B}(u, r)$ , a strongly stable ball for  $F$ ,  $F \cdot x \in \bar{B}(u, r)$ .

476 ► **Remark 4** (Contracting case). When  $0 \leq \mu < 1$ , for any  $\mu$ -strongly stable ball  $\bar{B}(v, \rho)$ , with  
 477 offset  $\sigma$ ,  $\bar{B}(F \cdot v, \mu\rho)$  is also a strongly stable ball, with offset  $\mu\sigma$ . Moreover,  $\bar{B}(F \cdot v, \mu\rho)$  is  
 478 included in  $\bar{B}(v, \rho)$ .



479

Assume that  $F$  has a  $\mu$ -strongly stable ball  $\bar{B}(u, r)$  of offset  $s$ ,  
 with  $\mu < 1$ . In particular,  $F$  is contracting on  $\bar{B}(u, r)$ . Consider  
 the sequence of balls defined as follows:

$$\bar{B}_n = \bar{B}(u_n, r_n) \quad \text{with} \quad u_n = F^n \cdot u \quad \text{and} \quad r_n = r\mu^n$$

where  $F^n \cdot u$  denotes the iterated images of  $u$  under  $F$ . By  
 Remark 4,  $(\bar{B}_n)_{n \in \mathbb{N}}$  is a nested sequence of  $\mu$ -strongly stable  
 ball for  $F$ , with offset  $s\mu^n$ . Let  $\mathcal{F}$  be the family of collections  
 of points in  $U$  defined as:

► **Figure 1** Balls  $\bar{B}_0$  and  $\bar{B}_1$       $\mathcal{F} = \{P \subseteq U \mid \exists n, \bar{B}_n \subseteq P\}$ .

480 It is a proper filter:  $\mathcal{F}$  contains  $U$ , it is obviously upward closed, and for  $P, Q \in \mathcal{F}$ ,  $P \cap Q$   
 481 is also in  $\mathcal{F}$  because  $(\bar{B}_n)_{n \in \mathbb{N}}$  is decreasing for inclusion. Thus  $\mathcal{F}$  has a limit  $w$ , such that  
 482 for any  $\varepsilon > 0$ , balls  $\bar{B}_n$  are eventually included in  $B(w, \varepsilon)$ . We provide a formal proof of  
 483 Theorem 5, a reformulation of Theorem 1 using the vocabulary of the COQUELICOT library:

484 ► **Theorem 5.** The limit  $w$  of the filter  $\mathcal{F}$  is in  $\bar{B}_0$ , and  $w$  is a fixed point of  $F$ . Moreover,  
 485  $w$  is close to every other fixed point of  $F$  in  $\bar{B}_0$ .

486 **Proof.** In this statement “ $w$  is a fixed point of  $F$ ” means “ $w$  is close to  $F \cdot w$ ”. First,  $w \in \bar{B}_n$   
 487 for all  $n$ . Indeed, for any  $\varepsilon > 0$ , there is an  $m \geq n$  s.t.  $\bar{B}_m \subseteq B(w, \varepsilon)$ , and since  $\bar{B}_m \subseteq \bar{B}_n$ ,  
 488  $u_m \in \bar{B}_n \cap B(w, \varepsilon)$ . In particular,  $w \in \bar{B}_0$ . It is also clear by stability that  $F \cdot w \in \bar{B}_n$  for  
 489 all  $n$ . Moreover,  $w$  is close to any point  $v$  s.t.  $v \in \bar{B}_n$  for all  $n$  (for any  $\varepsilon > 0$ , choose  $n$  s.t.  
 490  $2\mu r^n < \varepsilon$ ). Taking  $v := F \cdot w$  proves that  $w$  is a fixed point of  $F$ .

491 Finally, if  $w' \in \bar{B}_0$  is another fixed point of  $F$ , then it follows from an easy induction  
 492 that  $w' \in \bar{B}_n$  for all  $n$ . Hence, the foregoing shows that  $w$  is close to  $w'$ . ◀

493 Strongly stable balls model the requirements set on the untrusted data to be formally  
 494 verified. They can also be seen as balls centered at the initial point, and large enough  
 495 to include all its successive iterates, i.e. as instances of the locus at stake in classical  
 496 presentations of the proof. The version proved by Boldo et al. has a slightly more technical  
 497 wording, which seems to be made necessary by its further usage in the verification of the  
 498 Lax-Milgram theorem. Our proof script is significantly shorter, partly because we automate  
 499 proofs of positivity conditions (for radii of balls) using canonical structures for manifestly  
 500 positive expressions. But the key ingredient for concision is to make most of the filter device  
 501 in the proof, and to refrain from resorting to low-level properties of geometric sequences. To  
 502 the best of our knowledge, the other libraries of formalised analysis featuring a proof of this  
 503 result, notably Isabelle/HOL and HOL-Light, are based on variant of proof strategy closer  
 504 to the approach of Boldo et al. than to ours.

## 505 **5 Newton-like validation operators**

506 The purpose of this section is twofold. We first present the general principle of fixed-point  
 507 based *a posteriori* validation methods, and more particularly, the use of Newton-like validation  
 508 operators. Then we apply it to the division and square root of models.

509 Throughout this section, let  $(X, \|\cdot\|)$  denote a Banach space, and  $h^*$  the exact solution of an  
 510 equation in  $X$ . In this article,  $X$  stands for the space  $\mathcal{C}(I)$  of real-valued continuous functions  
 511 defined over a compact segment  $I = [a, b]$ , with the uniform norm  $\|h\| := \sup_{x \in I} |h(x)|$ . The  
 512 division and square root of functions are simple examples of solutions of equations in  $\mathcal{C}(I)$ ,  
 513 but there are also differential equations, integral equations, delay equation, etc. The general  
 514 scheme for Banach fixed-point based *a posteriori* validation methods follows two steps:

- 515 **1. Approximation step.** A numerical approximation  $h^\circ \in X$  of  $h^*$  is obtained by an  
 516 oracle, which may resort to any approximation method. In particular, this step requires  
 517 no mathematical assumption and can be executed purely numerically outside the proof  
 518 assistant, good approximation properties being only desirable for efficiency. In our setting  
 519 with  $X = \mathcal{C}(I)$ , interpolation at Chebyshev nodes (Section 5.1) is an efficient and accurate  
 520 oracle for a wide range of function space problems.
- 521 **2. Validation step.** The initial problem is rephrased in such a way that  $h^*$  is a fixed point  
 522 of a (locally) contracting operator  $\mathbf{F} : X \rightarrow X$ . An *a posteriori* error bound on  $\|h^\circ - h^*\|$   
 523 is deduced from the Banach fixed-point theorem (Theorem 1).

524 We thus need to find a contracting operator  $\mathbf{F}$  of which  $h^*$  is a fixed point. To this end,  
 525 we use Newton-like validation methods, which transform an equation  $\mathbf{M} \cdot h = 0$  into an  
 526 equivalent fixed-point equation  $\mathbf{F} \cdot h = h$  with  $\mathbf{F}$  contracting. More specifically, suppose that  
 527  $\mathbf{M} : X \rightarrow Y$  is differentiable; we use a Newton-like operator  $\mathbf{F} : X \rightarrow X$  defined as:

$$528 \quad \mathbf{F} \cdot h = h - \mathbf{A} \cdot \mathbf{M} \cdot h, \quad h \in X,$$

529 with  $\mathbf{A} : Y \rightarrow X$  an *injective bounded linear* operator, intended to be close to  $(\mathcal{D}\mathbf{M}_{h^\circ})^{-1}$ .  
 530 The operator  $\mathbf{A}$  may be given by an oracle and does not need to be this exact inverse (which  
 531 anyway might be non representable on computers exactly). The mean value theorem yields  
 532 a Lipschitz ratio  $\mu$  for  $\mathbf{F}$  over any convex subset  $S$  of  $X$ :

$$533 \quad \forall h_1, h_2 \in S, \|\mathbf{F} \cdot h_1 - \mathbf{F} \cdot h_2\| \leq \mu \|h_1 - h_2\|, \quad \text{with } \mu = \sup_{h \in S} \|\mathcal{D}\mathbf{F}_h\| = \sup_{h \in S} \|\mathbf{1}_X - \mathbf{A} \cdot \mathcal{D}\mathbf{M}_h\|,$$

534 which is expected to be small over some neighbourhood of  $h^\circ$ .

535 Concretely, in order to apply Theorem 1, one needs to compute the following quantities:

- 536 ■ a bound  $b \geq \|A \cdot M \cdot h^\circ\| = \|h^\circ - F \cdot h^\circ\|$ ;
- 537 ■ a bound  $\mu_0 \geq \|1_X - A \cdot \mathcal{D}M_{h^\circ}\| = \|\mathcal{D}F_{h^\circ}\|$ ;
- 538 ■ a bound  $\mu'(r) \geq \|A \cdot (\mathcal{D}M_h - \mathcal{D}M_{h^\circ})\| = \|\mathcal{D}F_h - \mathcal{D}F_{h^\circ}\|$  valid for any  $h \in B(h^\circ, r)$ ,
- 539 and parameterised by a radius  $r \in \mathbb{R}_+$ .

540 If we are able to find a radius  $r \in \mathbb{R}_+$  satisfying:

$$541 \quad \mu(r) := \mu_0 + \mu'(r) < 1, \quad \text{and} \quad b + r\mu(r) \leq r, \quad (4)$$

542 then Theorem 1 guarantees the existence and uniqueness of a root  $h^*$  of  $M$  in  $B(h^\circ, r)$ .

543 ► **Remark 6.** Finding an  $r$  as small as possible while satisfying (4) may be a nontrivial task  
 544 for automated validation procedures. For many problems,  $\mu'(r)$  is polynomial, hence condi-  
 545 tions (4) are polynomial inequalities over  $r$  (this is called the *radii polynomial approach* [18]  
 546 in rigorous numerics). In our case, division (resp. square root) induces an affine (resp.  
 547 quadratic) equation, which admits closed form solutions.

## 548 5.1 Approximation step: interpolation

549 Since they are certified a posteriori, (non-rigorous) approximations for division and square  
 550 root of given models can be obtained using arbitrary numerical techniques. We use interpol-  
 551 ation at Chebyshev nodes of the second kind for its efficiency and excellent approximation  
 552 properties [32].

553 Ideally, we would implement this operation outside of the proof assistant, in order not  
 554 to pay the price of an interpreted language. This would however require a lot of work in  
 555 order to design a proper interface between COQ values and external values (e.g., converting  
 556 COQ representation of floating points numbers into machine level floating points, and back).  
 557 Instead, and for now, we implement the oracles inside COQ, as unspecified functions. To this  
 558 end, we add a field to the structure `BasisOps_on`, to compute interpolants of a given degree:

```
559 interpolate: nat → (C → C) → seq C;
```

562 We implement this operation for Chebyshev basis using the discrete orthogonality relations  
 563 on Chebyshev polynomials.

564 To reduce the price of staying inside COQ for those computations, we exploit the poly-  
 565 morphism built in our framework to perform those computations on floating-point numbers  
 566 rather than intervals. To this end, we add the following fields to the structure `NBH`:

```
567 FF: Ops1; (* abstract type for floating points and their operations *)
568 I2F: II → FF; (* conversion from floating points to intervals (midpoint) *)
569 F2I: FF → II; (* conversion from intervals to floating points (singleton) *)
```

572 Equipped with these operations, we can define conversion operations between models (on  
 573 intervals) and polynomials with floating point coefficients:

```
574 Definition mcf (M: Model): seq FF := map I2F (pol M).
575 Definition mfc (p: seq FF): Model := { | pol := map F2I p; rem := 0 | }.
```

578 The field `FF`, of type `Ops1`, will moreover make it possible to call the functions `interpolate` and  
 579 `beval` from the basis with `C=FF`, i.e., to let them operate on floating point numbers. By doing  
 580 so we do not have to reimplement Clenshaw's evaluation scheme on floating point numbers.

## 5.2 Validation step for division

581

582 For  $f, g \in \mathcal{C}(I)$  with  $g$  nonvanishing over  $I$ , the quotient  $f/g$  is the unique root of  $\mathbf{M} : h \mapsto$   
 583  $gh - f$ . Let  $h^\circ$  be a candidate approximation given by the approximation step. Constructing  
 584 the Newton-like operator  $\mathbf{F}$  requires an approximation  $\mathbf{A}$  of  $(\mathcal{DM}_{h^\circ})^{-1} : k \mapsto k/g$ . For that  
 585 purpose, suppose  $w \approx 1/g \in \mathcal{C}(I)$  is also given by an oracle, and define:

$$586 \quad \mathbf{F} \cdot h = h - w(gh - f). \quad (5)$$

587 The next proposition computes an upper bound for  $\|h^\circ - f/g\|$ ; it is implemented in `div.newton`.

588 ► **Proposition 7.** *Let  $f, g, h^\circ, w \in \mathcal{C}(I)$ , and  $\mu, b \in \mathbb{R}_+$  such that:*

$$589 \quad (7i) \ \|w(gh^\circ - f)\| \leq b, \quad (7ii) \ \|1 - wg\| \leq \mu, \quad (7iii) \ \mu < 1.$$

590 *Then  $g$  does not vanish over  $I$  and  $\|h^\circ - f/g\| \leq b/(1 - \mu)$ .*

592 **Proof.** Conditions (7ii) and (7iii) imply that  $\mathbf{F}$  (Equation (5)) is contracting over  $\mathcal{C}(I)$  with  
 593 ratio  $\mu$ . The radius  $r := \frac{b}{1-\mu}$  makes the ball  $\bar{B}(h^\circ, r)$  strongly stable with offset  $b$  (7i), since  
 594  $b + \mu r = r$ . Therefore,  $h^*$  is the (global) unique root of  $\mathbf{M}$ , and  $\|h^\circ - h^*\| \leq r$ .

595 Finally,  $w$  and  $g$  do not vanish because  $\|1 - wg\| \leq \mu < 1$ . Hence,  $h^* = f/g$  over  $I$ . ◀

596 The concrete division of models is implemented as follows:

```
597 Definition mdiv_aux (F G H W : Model) : Model :=
598   let K1 := 1-W*G in
599   let K2 := W*(G*H - F) in
600   match mag (mrange K1), mag (mrange K2) with
601   | Some mu, Some b when is_lt mu 1 => { | pol := pol H; rem := rem H + sym (b/(1-mu)) | }
602   | _ => mbot
603   end.
604
605 Definition mdiv n (F G : Model) : Model :=
606   let p, q := mcf F, mcf G in
607   mdiv_aux F G (mfc (interpolate n (fun x => beval p x / beval q x)))
608   (mfc (interpolate n (fun x => 1 / beval q x))).
```

610 Note that we use the trivial model `mbot={|pol:=[];rem:=bot|}` as a default value, when the  
 611 concrete computations fail to validate the guess of the oracle (either because this guess is  
 612 just wrong, or because of over-approximations in the computations). The correctness lemmas  
 613 use the properties of operations on models to prove the assumptions of `div.newton`.

```
614 Lemma rmdiv_aux F f G g H h W w :
615   mcontains F f → mcontains G g → mcontains H h → mcontains W w →
616   mcontains (mdiv_aux F G H W) (f/g).
617
618 Lemma rmdiv n F f G g : mcontains F f → mcontains G g → mcontains (mdiv' n F G) (f/g).
```

## 5.3 Validation step for square root

620

621 Let  $f \in \mathcal{C}(I)$  be strictly positive over  $I$ . The square root  $\sqrt{f}$  is one of the two roots of the  
 622 quadratic equation  $\mathbf{M} \cdot h := h^2 - f = 0$  (the other being  $-\sqrt{f}$ ). Let  $h^\circ$  be a candidate  
 623 approximation. Since  $\mathcal{DM}_h : k \mapsto 2hk$ , one also needs an approximation  $w \approx 1/(2h^\circ) \approx$   
 624  $1/(2\sqrt{f}) \in \mathcal{C}(I)$  in order to define  $\mathbf{A} : k \mapsto wk$ , approximating  $(\mathcal{DM}_{h^\circ})^{-1}$ . Then:

$$625 \quad \mathbf{F} : h \mapsto h - w(h^2 - f). \quad (6)$$

626 The next proposition (implemented by `sqrt.newton`), computes an upper bound for  $\|h^\circ - \sqrt{f}\|$ .

627 ► **Proposition 8.** Let  $f, h^\circ, w \in \mathcal{C}(I)$ ,  $\mu_0, \mu_1, b \in \mathbb{R}_+$  and  $t_0 \in I$  such that:

$$628 \quad (8\text{ i}) \quad \left\| w \left( h^{\circ 2} - f \right) \right\| \leq b, \quad (8\text{ ii}) \quad \|1 - 2wh^\circ\| \leq \mu_0, \quad (8\text{ iii}) \quad \|w\| \leq \mu_1,$$

$$629 \quad (8\text{ iv}) \quad \mu_0 < 1, \quad (8\text{ v}) \quad (1 - \mu_0)^2 - 8b\mu_1 \geq 0, \quad (8\text{ vi}) \quad w(t_0) > 0.$$

631 Then  $f > 0$  over  $I$  and  $\|h^\circ - \sqrt{f}\| \leq r^*$  where  $r^* := \left(1 - \mu_0 - \sqrt{(1 - \mu_0)^2 - 8b\mu_1}\right) / 4\mu_1$ .

632 **Proof.** First, since  $\|1 - 2wh^\circ\| \leq \mu_0 < 1$  (by (8 ii) and (8 iv)) and  $w(t_0) > 0$  (8 vi),  $w$  and  
633  $h^\circ$  are strictly positive over  $I$ , by continuity. Using (8 iii),  $\mu_1 > 0$ .

634 If  $b = 0$ , then  $r^* = 0$  and  $h^\circ = \sqrt{f}$  over  $I$ , because  $w(h^{\circ 2} - f) = 0$  (8 i) and  $w, h^\circ > 0$ .  
635 Hence the conclusion holds.

636 From now on, we assume  $b > 0$ .  $\mathbf{F}$  is Lipschitz of ratio  $\mu(r) := \mu_0 + 2\mu_1 r$  over  $\overline{B}(h^\circ, r)$   
637 for any  $r \in \mathbb{R}_+$ , because:

$$638 \quad \mathbf{F} \cdot h_1 - \mathbf{F} \cdot h_2 = (h_1 - h_2) - w(h_1^2 - h_2^2) = [(1 - 2wh^\circ) + w(h^\circ - h_1) + w(h^\circ - h_2)](h_1 - h_2).$$

639 Therefore, satisfying  $b + \mu(r)r \leq r$  is equivalent to the quadratic inequality:

$$640 \quad 2\mu_1 r^2 + (\mu_0 - 1)r + b \leq 0. \quad (7)$$

641 Condition (8 v) implies that (7) admits solutions, and  $r^*$  is the smallest one. Moreover, since  
642  $b, \mu_1 > 0$ , we get  $r^* > 0$ , so that  $b + \mu(r^*)r^* = r^*$  also implies  $\mu(r^*) < 1$ .

643 Now, all the assumptions of Theorem 1 are fulfilled. Hence,  $\mathbf{F}$  has a unique fixed point  
644  $h^*$  in  $\overline{B}(h^\circ, r^*)$ . To obtain  $h^* = \sqrt{f}$  over  $I$ , it remains to show that  $h^* > 0$ . This follows  
645 from  $w > 0$  and:

$$646 \quad \|1 - 2wh^*\| \leq \|1 - 2wh^\circ\| + \|2w(h^* - h^\circ)\| \leq \mu_0 + 2\mu_1 r^* = \mu(r^*) < 1. \quad \blacktriangleleft$$

647 ► **Remark 9.** Contrary to the case of division where continuity was not needed at all, it is  
648 here used for  $w$ . Therefore, `sqrt.newton` requires  $w$  to be continuous over  $I$ .

649 The CoQ code for the corresponding operations on models `msqrt_aux` and `msqrt`, together  
650 with the statements of their correctness lemmas, are given in Appendix B.

## 651 6 Examples

### 652 6.1 Playing with approximations of the absolute value function

653 Consider the function  $f_\varepsilon : x \mapsto \sqrt{\varepsilon + x^2}$  over  $[-1, 1]$ , with  $\varepsilon > 0$ . When  $\varepsilon \rightarrow 0$ ,  $f_\varepsilon$  converges  
654 uniformly to the absolute value function  $x \mapsto |x|$  (which is not analytic at 0), with:

$$655 \quad |f(x) - |x|| = \left| \sqrt{\varepsilon + x^2} - \sqrt{x^2} \right| = \left| \frac{\varepsilon}{\sqrt{\varepsilon + x^2} + \sqrt{x^2}} \right| \leq \sqrt{\varepsilon}. \quad (8)$$

656 **Rigorous uniform approximations.** Approximating  $f_\varepsilon$  with polynomials becomes harder  
657 for small  $\varepsilon$ , due to the complex singularities  $\pm i\sqrt{\varepsilon}$  getting closer to the interval  $[-1, 1]$ .  
658 Nevertheless, Chebyshev interpolation still works and our implementation computes rigorous  
659 approximations as accurate as desired (see Figure 2b), of exponential convergence with ratio  
660 determined by  $\varepsilon$ . Note that for too small degree, the computed approximation of the square  
661 root is too far from the solution, and the a posteriori validation returns an infinite remainder.

662 In order to provide a comparison with COQAPPROX's Taylor models, we used the  
663 tactic `interval with (i_depth 1, i_bisect_taylor x N, i_prec p)` to build a Taylor model of

664 degree  $N$  with precision  $p$ . Timings given in Table 2c reveal a significant advantage of  
 665 our implementation (there we use  $\varepsilon = 2$  to avoid convergence issues of Taylor models).  
 666 Concerning accuracy, our experiments tend to show that when  $\varepsilon \leq 1$ , COQAPPROX fails to  
 667 compute *converging* Taylor models. Indeed, even with large  $L$ , a goal like:

668 **Goal Fail** :  $\forall x : \mathbb{R}, -1 \leq x \leq 1 \rightarrow \text{sqrt}(1/100+x*x) \leq L$   
 669

671 is not solved when the degree  $N$  becomes too large, probably indicating that the Taylor  
 672 models *diverge* due to complex singularities inside the unit disk. (Note that the `interval`  
 673 tactic can solve this goal, but only by resorting to subdivision techniques.)

674 **Error bounding.** We want to bound  $|f_\varepsilon(x) - |x||$  for  $x \in [-1, 1]$  without making use of  
 675 any symbolic manipulation like (8). At first glance, one can choose to use the rigorous  
 676 approximations over  $[-1, 1]$  obtained previously, and evaluate  $f_\varepsilon(x) - x$  (resp.  $f_\varepsilon(x) + x$ )  
 677 over  $[0, 1]$  (resp.  $[-1, 0]$ ) using Clenshaw’s algorithm. However, even if the approximations  
 678 are quite good, this evaluation strategy gives huge overestimations because  $[0, 1]$  and  $[-1, 0]$   
 679 are not small intervals. Instead, we compute separately two approximations for  $f_\varepsilon$ : one  
 680 over  $[0, 1]$  and one over  $[-1, 0]$ , and we evaluate  $f_\varepsilon(x) - x$  (resp.  $f_\varepsilon(x) + x$ ) over  $[1, 0]$  (resp.  
 681  $[-1, 0]$ ) using the Chebyshev `range` function. This approach yields bounds that are rather  
 682 close to the optimal  $\sqrt{\varepsilon}$  (see Figure 2d). However, this does not allow for arbitrary accuracy:  
 683 a subdivision procedure would be necessary here.

## 684 6.2 Evaluating an Abelian integral

685 Abelian integrals naturally appear when computing the number of limit cycles bifurcating  
 686 from a Hamiltonian polynomial vector field in the plane. Indeed, the number of sign  
 687 alternations of those contour integrals (parameterised by the energy level of the potential  
 688 function) gives a lower bound on the number of limit cycles of the perturbed system, which  
 689 is a hard question related to Hilbert’s 16th problem.

690 In [21], the author claims to prove the existence of 26 limit cycles for a well-constructed  
 691 quartic system, whereas the previous record for degree 4 was 22 [12]. However, the im-  
 692 plementation with which the Abelian integrals were “rigorously” computed was erroneous,  
 693 which led to apparently more sign alternations than in reality. By tuning the coefficients and  
 694 computing the integrals with another rigorous numerics library, the authors of the ongoing  
 695 work [9] obtain 24 limit cycles, which, if not 26, is still greater than the current record 22.

696 To conclude this article, we rigorously evaluate some of these integrals inside COQ to  
 697 show how our implementation behaves on non-crafted examples. Below are the formulas  
 698 defining a family of integral  $\mathcal{J}_{ij}(r)$  which need to be computed precisely for several values of  
 699  $r$ . Table 1 summarises the results of our computational experiments. In each line, we chose  
 700 parameters that were enough to obtain the desired precision. These encouraging results give  
 701 us hope that it will be possible to fully verify the critical computations involved in recent  
 702 work of the first author [9].

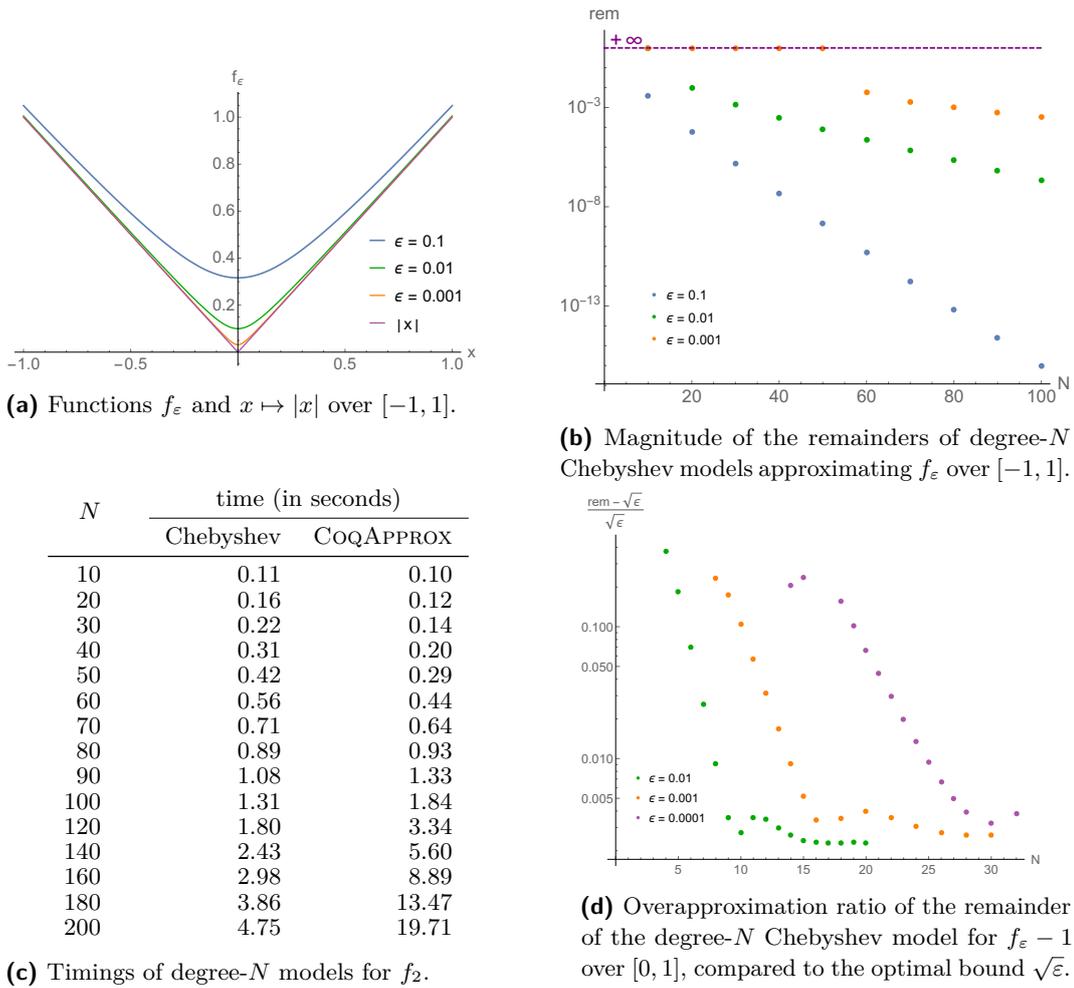
703

$$704 \mathcal{J}_{ij}(r) = \int_{x_-}^{x_+} x^i (y^+(x)^{j-1} - y^-(x)^{j-1}) dx + \int_{y_-}^{y_+} (x^-(y)^{i-1} + x^+(y)^{i-1}) y^j \frac{y^2 - y_0}{\delta_x(y)} dy.$$

$$706 x_0 = \frac{9}{10}, \quad x_\pm = \sqrt{x_0 \pm r/\sqrt{2}}, \quad \delta_y(x) = \sqrt{r^2 - (x^2 - x_0)^2}, \quad x^\pm(y) = \sqrt{x_0 \pm \delta_x(y)},$$

$$708 y_0 = \frac{11}{10}, \quad y_\pm = \sqrt{y_0 \pm r/\sqrt{2}}, \quad \delta_x(y) = \sqrt{r^2 - (y^2 - y_0)^2}, \quad y^\pm(x) = \sqrt{y_0 \pm \delta_y(x)}.$$

709



■ **Figure 2** Approximating functions  $f_\epsilon$  and  $x \mapsto |x|$  with Chebyshev models.

$r$	$N$	$p$	time (s)	$\mathcal{I}_{00}$	$\mathcal{I}_{20}$	$\mathcal{I}_{22}$	$\mathcal{I}_{40}$	$\mathcal{I}_{04}$
0.5	13	32	0.38	2,4e-05	2,9e-05	4,1e-05	3,0e-05	4,8e-05
0.78	15	32	0.47	4,6e-05	2,0e-05	2,7e-05	2,4e-05	1,1e-04
0.88	65	128	17.34	2,5e-08	5,0e-11	8,5e-11	5,3e-11	6,3e-08
0.89	95	128	35.13	2,0e-08	1,8e-11	2,9e-11	2,0e-11	5,1e-08
0.895	135	300	173.23	2,6e-08	1,7e-11	1,8e-11	1,3e-11	6,7e-08

■ **Table 1** Reached precision for  $\mathcal{I}_{ij}(r)$  for different values of  $r$ , computed with degree- $N$  Chebyshev models and floating-point precision  $p$  (in each cell we display the width of the computed enclosure).

## 7 Conclusion and future work

The COQ development is available online [11]. It consists of around 1300 lines of specifications and 1500 lines of proofs. We leave several directions for future work: integrate it with COQINTERVAL to benefit from its automatic subdivision techniques; interface the library with external tools for the approximation steps; implement other bases. Applying this approach to verify solutions of linear ODEs in a systematic way [2, 8] is also a longer-term perspective.

## 716 — References —

- 717 1 H. Barendregt and E. Barendsen. Autarkic computations in formal proofs. *Journal of*  
718 *Automated Reasoning*, 28(3):321–336, Apr 2002.
- 719 2 A. Benoit, M. Joldeş, and M. Mezzarobba. Rigorous uniform approximation of D-finite  
720 functions using Chebyshev expansions. *Math. Comp.*, 86(305):1303–1341, 2017.
- 721 3 V. Berinde. *Iterative approximation of fixed points*, volume 1912 of *Lecture Notes in Mathem-*  
722 *atics*. Springer, Berlin, 2007.
- 723 4 M. Berz and K. Makino. Verified integration of odes and flows using differential algebraic  
724 methods on high-order Taylor models. *Reliable Computing*, 4(4):361–369, 1998.
- 725 5 S. Boldo, F. Clément, F. Faissolle, V. Martin, and M. Mayero. A Coq formal proof of the  
726 Lax–Milgram theorem. In *Proc. 6th ACM SIGPLAN Conference on Certified Programs and*  
727 *Proofs*, Paris, France, Jan. 2017.
- 728 6 S. Boldo and G. Melquiond. *Verifying Floating-point Algorithms with the Coq System*. Elsevier,  
729 2017.
- 730 7 N. Bourbaki. *General Topology*. Springer, 1995. Original French edition published by MASSON,  
731 Paris, 1971.
- 732 8 F. Bréhard, N. Brisebarre, and M. Joldeş. Validated and numerically efficient Chebyshev  
733 spectral methods for linear ordinary differential equations. *ACM Transactions on Mathematical*  
734 *Software*, 2018.
- 735 9 F. Bréhard, N. Brisebarre, M. Joldeş, and W. Tucker. A New Lower Bound on the Hilbert  
736 Number for Quartic Systems, 2019.
- 737 10 N. Brisebarre and M. Joldeş. Chebyshev interpolation polynomial-based tools for rigorous  
738 computing. In *Proc. Proceedings of the 2010 International Symposium on Symbolic and*  
739 *Algebraic Computation*, pages 147–154. ACM, 2010.
- 740 11 F. Bréhard, A. Mahboubi, and D. Pous. Web appendix to the present paper. [http://perso.](http://perso.ens-lyon.fr/florent.brehard/chebapprox/)  
741 [ens-lyon.fr/florent.brehard/chebapprox/](http://perso.ens-lyon.fr/florent.brehard/chebapprox/).
- 742 12 C. Christopher. Estimating limit cycle bifurcations from centers. In *Differential equations*  
743 *with symbolic computation*, Trends Math., pages 23–35. Birkhäuser, Basel, 2005.
- 744 13 T. A. Driscoll, N. Hale, and L. N. Trefethen. *Chebfun guide*, 2014.
- 745 14 L. Fox and I. B. Parker. *Chebyshev polynomials in numerical analysis*. Oxford University  
746 Press, London-New York-Toronto, Ont., 1968.
- 747 15 B. Grégoire and L. Théry. A purely functional library for modular arithmetic and its application  
748 to certifying large prime numbers. In *Proc. Automated Reasoning, Third International Joint*  
749 *Conference, IJCAR 2006, Seattle, WA, USA, August 17-20, 2006, Proceedings*, volume 4130 of  
750 *Lecture Notes in Computer Science*, pages 423–437. Springer, 2006.
- 751 16 T. C. Hales, M. Adams, G. Bauer, D. T. Dang, J. Harrison, T. L. Hoang, C. Kaliszyk,  
752 V. Magron, S. McLaughlin, T. T. Nguyen, T. Q. Nguyen, T. Nipkow, S. Obua, J. Pleso,  
753 J. Rute, A. Solovyev, A. H. T. Ta, T. N. Tran, D. T. Trieu, J. Urban, K. K. Vu, and  
754 R. Zumkeller. A formal proof of the Kepler conjecture. *CoRR*, abs/1501.02155, 2015.
- 755 17 J. Hölzl, F. Immler, and B. Huffman. Type classes and filters for mathematical analysis in  
756 isabelle/hol. In S. Blazy, C. Paulin-Mohring, and D. Pichardie, editors, *Interactive Theorem*  
757 *Proving*, pages 279–294, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.
- 758 18 A. Hungria, J.-P. Lessard, and J. D. Mireles James. Rigorous numerics for analytic solutions  
759 of differential equations: the radii polynomial approach. *Math. Comp.*, 85(299):1427–1459,  
760 2016.
- 761 19 F. Immler. A verified ODE solver and the Lorenz attractor. *Journal of automated reasoning*,  
762 pages 1–39, 2018.
- 763 20 F. Johansson. Arb: efficient arbitrary-precision midpoint-radius interval arithmetic. *IEEE*  
764 *Transactions on Computers*, 66(8):1281–1292, 2017.
- 765 21 T. Johnson. A quartic system with twenty-six limit cycles. *Exp. Math.*, 20(3):323–328, 2011.
- 766 22 M. Joldeş. *Rigorous Polynomial Approximations and Applications*. PhD thesis, École normale  
767 supérieure de Lyon – Université de Lyon, Lyon, France, 2011.

- 768 23 E. W. Kaucher and W. L. Miranker. *Self-validating numerics for function space problems: Computation with guarantees for differential and integral equations*, volume 9. Elsevier, 1984.
- 769
- 770 24 R. Klatté, U. Kulisch, A. Wiethoff, and M. Rauch. *C-XSC: A C++ class library for extended scientific computing*. Springer Science & Business Media, 2012.
- 771
- 772 25 J.-P. Lessard and C. Reinhardt. Rigorous numerics for nonlinear differential equations using Chebyshev series. *SIAM J. Numer. Anal.*, 52(1):1–22, 2014.
- 773
- 774 26 A. Mahboubi, G. Melquiond, and T. Sibut-Pinote. Formally verified approximations of definite integrals. *Journal of Automated Reasoning*, 62(2):281–300, Feb 2019.
- 775
- 776 27 É. Martin-Dorel and G. Melquiond. Proving tight bounds on univariate expressions with elementary functions in coq. *Journal of Automated Reasoning*, 57(3):187–217, Oct 2016.
- 777
- 778 28 G. Melquiond. Proving bounds on real-valued functions with computations. In *Proc. International Joint Conference on Automated Reasoning*, pages 2–17. Springer, 2008.
- 779
- 780 29 R. E. Moore. *Interval Analysis*. Prentice-Hall, 1966.
- 781
- 782 30 N. Revol and F. Rouillier. Motivations for an arbitrary precision interval arithmetic and the mpfi library. *Reliable computing*, 11(4):275–290, 2005.
- 783
- 784 31 S. M. Rump. Intlab—interval laboratory. In *Developments in reliable computing*, pages 77–104. Springer, 1999.
- 785
- 786 32 L. N. Trefethen. *Approximation Theory and Approximation Practice*. SIAM, 2013. See <http://www.chebfun.org/ATAP/>.
- 787
- 788 33 W. Tucker. A rigorous ODE solver and Smale’s 14th problem. *Found. Comput. Math.*, 2(1):53–117, 2002.
- 789
- 790 34 W. Tucker. *Validated numerics: a short introduction to rigorous computations*. Princeton University Press, 2011.
- 791
- 792 35 J. B. Van Den Berg and J.-P. Lessard. Chaotic braided solutions via rigorous numerics: Chaos in the Swift–Hohenberg equation. *SIAM Journal on Applied Dynamical Systems*, 7(3):988–1031, 2008.
- 793
- 794 36 N. Yamamoto. A numerical verification method for solutions of boundary value problems with local uniqueness by Banach’s fixed-point theorem. *SIAM J. Numer. Anal.*, 35(5):2004–2013, 1998.
- 795
- 796

**A** Coq code for primitive in Chebyshev basis

797

```

798 Fixpoint prim_ (C: Ops1) (n: nat) (p: seq C): seq C :=
799   match P,n with
800     | [],_ => []
801     | a::q,0 => sadd [0; a] (prim_ 1 q)
802     | a::q,1 => 0 :: (sadd [0; a/4] (prim_ 2 q))
803     | a::q,_ => sadd [-a/(2(n-1)); 0; a/(2(n+1))] (0 :: prim_ (n+1) Q)
804   end.
805 Definition prim (C: Ops1) (P: seq C) := prim_ 0 P.
806

```

808 The key lemma is the following one:

```

809 Lemma eval_prim_ n (p: seq R) x: Derive (eval_ T (n-1) (prim_ n P)) x = eval_ T n P x.
810
811

```

**B** Coq code for the square root of a model

812

```

813 Let msqrt_aux (F H W: Model) (x: II): Model :=
814   let Wx := meval W x in
815   if ~ (is_lt lo x && is_lt x hi && is_lt 0 Wx) then mbot else
816   let K1 := 1 - 2*W*H in
817   let K2 := W*(H*H-F) in
818   match mag (mrange K1), mag (mrange W), mag (mrange K2) with
819     | Some mu0, Some mu1, Some b =>
820       let delta := (1 - mu0)^2 - 8*b*mu1 in
821       let rmin := (1 - mu0 - sqrt delta)/(4*mu1) in
822       let mu := mu0 + 2*mu1*rmin in
823       if is_lt mu 1 && is_lt 0 delta && is_lt mu' 1 then
824         { | pol := pol H; rem := rem H + sym rmin' | }
825       else mbot
826     | _ => mbot
827   end.
828
829 Let msqrt n (F: Model): Model :=
830   let p: seq FF := mcf F in
831   let h: seq FF := interpolate n (fun x => sqrt (beval p x)) in
832   msqrt_aux M (mfc h) (mfc (interpolate n (fun x => 1/(2*beval h x)))) ((lo+hi)/2).
833
834 Lemma rmsqrt_aux (F H W: Model) (X: II) (f h w : R → R) (x: R):
835   mcontains F f → mcontains H h → mcontains W w → contains X x → lo ≤ x ≤ hi →
836   (∀ x, lo ≤ x ≤ hi → continuity_pt w x) →
837   mcontains (msqrt_aux F H W X) (sqrt f).
838
839 Lemma rmsqrt n F f: mcontains F f → mcontains (msqrt' n F) (sqrt f).
840
841

```