Formal Proofs for Theoretical Properties of Newton's Method

Ioana Pasca

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Formal Proofs for Theoretical Properties of Newton's Method

Ioana Paşca

N° 7228

February 2010

ISSN 0249-6399 ISRN INRIA/RR--7228--FR+ENG

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA

centre de recherche
SOPHIA ANTIPOLIS - MÉDITERRANÉE

Formal Proofs for Theoretical Properties of Newton's Method

Ioana Paşca

Thème : Systèmes symboliques
Équipe-Projet Marelle

Rapport de recherche n° 7228 — February 2010 — 28 pages

Abstract: We discuss a formal development for the certification of Newton's method. We address several issues encountered in the formal study of numerical algorithms: developing the necessary libraries for our proofs, adapting paper proofs to suite the features of a proof assistant, and designing new proofs based on the existing ones to deal with optimizations of the method. We start from Kantorovitch's theorem that states the convergence of Newton's method in the case of a system of equations. To formalize this proof inside the proof assistant Coq we first need to code the necessary concepts from multivariate analysis. We also prove that rounding at each step in Newton's method still yields a convergent process with an accurate correlation between the precision of the input and that of the result. An algorithm including rounding is a more accurate model for computations with Newton's method in practice.

Key-words: proof assistants, formalization of mathematics, multivariate analysis, Kantorovitch's theorem, Newton's method with rounding

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles, BP 93, 06902 Sophia Antipolis Cedex
Téléphone : +33 4 92 38 77 77 — Télécopie : +33 4 92 38 77 65

Preuves formelles pour les propriétés théoriques de la méthode de Newton

Résumé : Ce rapport présente un développement formel pour la certification de la méthode de Newton. On s'intéresse à plusieurs problèmes rencontrés dans l'étude formelle des algorithmes numériques : développer les bibliothèques nécessaires pour nos preuves, adapter des preuves papier aux caractéristiques d'un assistant à la preuve, concevoir des nouvelles preuves basées sur les preuves existantes pour certifier des optimisations de la méthode. Notre point de départ est le théorème de Kantorovitch qui établit la convergence de la méthode de Newton dans le cas d'un système d'équations. Pour formaliser ce théorème dans l'assistant à la preuve Coq on a besoin d'abord de coder les concepts nécessaires d'analyse multivariée. On démontre aussi qu'arrondir à chaque itération de la méthode de Newton donne lieu à un processus qui est encore convergent, avec une corrélation bien determinée entre la précision des données d'entrée et celle du résultat. Un algorithme avec des arrondis est un modèle plus fidèle pour les calculs pratiques par la méthode de Newton.

Mots-clés : assistants à la preuve, formalisation des mathématiques, analyse multivariée, théorème de Kantorovitch, méthode de Newton avec arrondis

1 Formal systems and numerical methods

Often, in verifying mathematical theorems in proof assistants we start with a paper proof of some (famous) theorem and try to obtain a formal model of the theorem inside the system. The concepts are coded in a manner that keeps the balance between mathematical accuracy and handiness of use. This encoding process is not always trivial as the mathematical concepts, expressed in general in set theory, need to be translated into type theory or higher order logic. The limitations and benefits of the formal framework need to be taken into account. Once the concepts inside the system, we try to reproduce the reasoning steps to get the desired proof. Automatization is often possible for some (small) parts of the problem, depending on the field. A good example of a field where mechanization is wide spread is algebra while calculus is less prone to automatization. As a consequence formal developments in real or numerical analysis are more tedious. This is a set back for proof assistants in comparison to computer algebra system which support a wide variety of numerical methods. However, it is sometimes the case that these systems produce erroneous output [17, 7]. So, when a high level of correctness is required, choosing a proof assistant for the task could be a good solution. The aim of this paper is to describe several aspects of doing a formal development around a numerical algorithm. We discuss the problems encountered, possible solutions and potential applications, in order to allow a better understanding of what such a development entails. Among others, we address the following issues:

- providing the proof assistant with the necessary concepts to support all reasoning steps that we are interested in;
- formalizing a piece of mathematics stating the desired properties for our algorithms;
- designing new proof based on the existing ones to offer theoretical basis for optimizations of our algorithms.

We do a case study on Newton's method and detail all the points above. Widely used as an approximation method to determine the root of a given function or, equivalently, the solution of a system of equations, Newton's method has good performance with respect to the speed of convergence and the stability of the process. These performances are backed by theoretical results in numerical analysis establishing sufficient conditions for the convergence of the method. Among such results we have Kantorovitch's theorem, which we chose as a basis for our formal development. The statement of the theorem according to [9] is as follows:

Theorem 1 (Kantorovitch) *Consider a system of non-linear algebraic or transcendent equations $f(x) = 0$, where the vector function $f : \mathbb{R}^p \to \mathbb{R}^p$ has continuous first and second partial derivatives in a certain domain ω, i.e. $f(x) \in C^{(2)}(\omega)$. Let $x^{(0)}$ be a point with its closed ε-neighborhood $\overline{U_\varepsilon}(x^{(0)}) = \{\|x - x^{(0)}\| \leq \varepsilon\}$ included in ω. If the following conditions hold:*

1. *the Jacobian matrix $W(x) = [\frac{\partial f_i(x)}{\partial x_j}]$ has an inverse for $x = x^{(0)}$, $\Gamma_0 = W^{-1}(x^{(0)})$ with $\|\Gamma_0\| \leq A_0$;*

2. $\|\Gamma_0 f(x^{(0)})\| \leq B_0 \leq \frac{\varepsilon}{2}$;

3. $\sum_{k=1}^{p} |\frac{\partial^2 f_i(x)}{\partial x_j \partial x_k}| \leq C$ *for* $i, j = 1, 2, ..., p$ *and* $x \in \overline{U_\varepsilon}(x^{(0)})$;

4. *the constants* A_0, B_0, C *satisfy the inequality* $2pA_0B_0C \leq 1$.

then, for the initial approximation $x^{(0)}$*, the Newton process*

$$x^{(n+1)} = x^{(n)} - W^{-1}(x^{(n)})f(x^{(n)}) \tag{1}$$

$(n = 1, 2, ...)$ *converges and the limit vector* $x^* = \lim_{n\to\infty} x^{(n)}$ *is a solution of the initial system, so that* $\|x^* - x^{(0)}\| \leq 2B_0 \leq \varepsilon$. *Moreover, in the domain* $\{\|x - x^{(0)}\| \leq 2B_0\}$ *the solution is unique.*

A quick look at the theorem reveals that a formal model of the problem needs to include formalizations of real numbers and real analysis in one and several dimensions. The background results in analysis need to deal with continuity, differentiation, convergence, etc. Properties on matrices are also used as concepts of inverse or Jacobian matrix need to be handled.

For our work we chose the proof assistant Coq [1, 4] which provides a library of real analysis that is developed enough to cover the concepts needed in the proof of Kantorovitch's theorem in the simplified case of a real function. For the multivariate case, however, Coq does not offer a library, so we need to encode all the necessary concepts. We provide a reusable formalization of multivariate analysis concepts and we present the details in section 2. Section 3 discusses the formalization of the Kantorovitch theorem, insisting on the relation between paper proofs and formal proofs. In particular, we discuss an optimized version of Newton's method, where we perform rounding at each step. Based on the proof of Kantorovitch's theorem we prove in section 3.1 that this optimized version converges to the root. The last section presents the conclusions and perspectives of our work.

2 Developping libraries

The only proof assistant that already contains a formalization of multivariate analysis is HOL Light. In this formalization, documented in [16], vectors are implemented as functions N→real, where N is a type with finite cardinality and real is the type of real numbers in HOL. Vectors are of type (N)finite_image→real, where (N)finite_image has the same size as N when N is a finite type and 1 otherwise. An indexing operator is defined that allows the use of natural numbers as indexes. The topics covered include linear algebra: operators, matrices, determinants; topology: open, closed, compact, convex sets; sequences, continuity, differentiability; basic calculus theorems: mean value theorem, inverse function theorem.

To find the best suited formalization for multivariate analysis inside the Coq system we first took a look at what basic concepts we need to manipulate, at how we need to manipulate them and at what facilities Coq offers in comparison to HOL Light. The basic objects we work with are vectors of length p of real numbers or elements of $\mathbb{R}^p$. On these vectors, we want to easily define operations

like addition and prove the properties of these operations. We also want to easily handle norms on vectors or matrices.

Since Coq has a richer type system than HOL Light, in particular it supports dependent types, it is worth looking for a new representation for our vectors and not just copying the HOL Light version. There are two main possibilities for implementing vectors, either as lists or as functions. In dependent functional programming tradition vectors are often implemented as dependent lists of length p (see for example [21]). This implementation makes functions like the norm easy to define as folding the maximum or addition function on a list. However, coding even basic operations like addition requires complicated forms of dependent recursion. On the other hand, implementing vectors as functions makes it easy to define operations and to do proofs, as the HOL Light development suggests.

We wanted to benefit from both representations by having different views for vectors. This is well supported by the libraries of Coq's SSReflect extension [13, 14]. SSReflect provides a formalization for finite types as records containing a type, a list with all the elements of the type and a predicate saying the list is duplicate free. The finite type we used is ordinal p (notation 'I_p) which encodes the set of natural numbers smaller than p. The vectors can be encoded as finite functions 'I_p→R. This gives the vectors as functions view. The list in the structure of the finite type 'I_p gives the vectors as lists view, which allows iterating operations over the components of a vector.

The definition of finite types is only one aspect of the SSReflect libraries that we used in this paper. As we want to have a self contained document, we will detail in the next section the features and concepts of SSReflect that played a role in our development. The reader familiar with the library may skip this part. We also present in short the main elements of Coq's standard library for real numbers before continuing with the description of our development.

2.1 SSReflect extension and libraries

SSReflect (**S**mall **S**cale **Reflect**ion) is an extension of Coq that offers new syntax features for the proof shell and basic libraries that make use of small scale reflection in various respects. The proof of the Four Color Theorem and the on-going effort to prove Feit-Thompson theorem illustrate the power of SSReflect. An extended presentation for the tactics can be found in [13].

SSReflect disposes of a mechanism that provides dual views for decidable predicates. This is done by reflection between decidable propositions and booleans. The propositional version is appropriate when doing structured proofs while the boolean view is used for computing. The user can move from one view to the other by a simple rewrite. This makes the framework particularly appropriate for working with structures equipped with a decidable equality, as in this case various properties can be reflected by boolean values. In the library, a type with decidable equality is implemented as a type T together with relation eq_op: T→T→bool that reflects the Leibniz equality on that type, i.e. eq_op x y is true exactly when x = y.

The library proposes a methodology for defining structures based on the notion of mixin [12]. A structure is a type and a mixin that packages together all the information we need on the type. For types with decidable equality, for

example, the mixin packages the relation eq_op and the proof that this relation reflects the Leibniz equality.

```
Module Equality.
Record mixin_of (T : Type) : Type :=
  Mixin {op : rel T; _ : forall x y, reflect (x = y) (op x y)}.
Structure type : Type :=
  Pack {sort :> Type; mixin : mixin_of sort}.
End Equality.
Notation eqType := Equality.type.
Notation EqMixin := Equality.Mixin.
Notation EqType := Equality.Pack.
Definition eq_op T := Equality.op (Equality.mixin T).
Notation "x == y" := (@eq_op _ x y).
```

The sort :> Type declaration above makes the sort projection into a coercion. This form of explicit subtyping allows any T : eqType to be used as a Type. In the other direction, we can use a concrete Type as an eqType thanks to Coq's Canonical Structure mechanism. We take an example to illustrate the way it works: natural numbers. In Coq they are inductively defined as Peano integers with zero and successor function. Based on this definition we can build a decidable equality predicate eqn : nat→nat→bool. This predicate reflects the Leibniz equality:

```
Lemma eqnP : forall x y : nat, reflect (x = y) (eqn x y).
```

Now we can build our Equality mixin and Equality type for the natural numbers.

```
Canonical Structure nat_eqMixin := EqMixin eqnP.
Canonical Structure nat_eqType := EqType nat_eqMixin.
```

The Canonical Structure declaration will make that every time an expression requires an eqType, but gets a nat instead, Coq will automatically infer the type nat_eqType for the expected argument. The expression will type-check without intervention from the user. This means the generic theorems and notations for eqTypes can directly be applied to natural numbers.

Following the same design pattern the library implements choiceType, a type T with a choice function choose that returns a canonical representant of any non-empty subset of elements of type T. Natural numbers, for example, are a choiceType as we can define the function choose as the function that starts from zero and checks all numbers until it finds an element of the given non-empty set. The set being non-empty the function will only need a finite number of steps. This remark is more general, any countable type can be endowed with a canonical choice function.

Finite types play a central role in the development. The mixin for a finType contains the list of all its elements and the property that in this list each element appears exactly once. As an example we saw in the previous section ordinal p, notation 'I_p, the type of natural numbers smaller than p.

Functions with finType as domain benefit from a special treatment in the library. Such a function can be represented as the list of all its values. It is coerced to the corresponding arrow type. To define a finfun we use the notation {ffun aT→rT}. If the return type rT is an eqType then the finite function type will also be an eqType as the extensional equality on functions will reflect the

Leibniz equality. Similarly, if rT is a choiceType, then {ffun aT→rT} will also be a choiceType.

Once these basic structures are in place, we can build our algebraic structures. In version 1.2 of SSReflect the hierarchy contains groups, abelian groups, rings, commutative rings and fields.

The development also contains a library that treats in a general fashion indexed operations. By this, we mean we have a uniform way of writing:

$$\sum_{i=0}^{n} x_i \quad \text{or} \quad \prod_{i \in I} v_i \quad \text{or} \quad \max_{i, v_i \neq w} \|v_i - w\|$$

Formally, the general notation is:

```
\big[op/nil]_(i <- r | P i) F
```

where r represents the list of indexes i for which the operation op is to be repeated; nil is the value to be return for the empty list of indexes (usually the neutral element for the operation, if it exists) while P is the property that the indexes have to respect; F is the expression over which the operation is iterated.

When translating the above formulas in Coq, in the first case we write:

```
\big[+/0]_(i < n) x i
```

Supposing that I and r are lists of indexes, the second formula is:

```
\big[*/1]_(i <- I) v i
```

and the third:

```
\big[Rmax/0]\_(i <- r | v i != w) (norm (v i) - w)
```

We note that sums and products indexed over natural numbers can be written with a more natural \sum or \prod notation. For example, the first formula can alternatively be written as: \sum_(i < n) x i .

The lemmas in the library are organized according to the properties of the operator op. Some lemmas work for any operator, others work only if op is a monoid law, others require an abelian monoid law and so on. Canonical structures play an important role here also. Details can be found in [2].

Making use of the indexed operations, a formalization of matrices with elements of type R is given. Matrices in $M_{p \times q}(\mathsf{R})$ are represented as finite functions {ffun 'I_p * 'I_q→R}. For operations on rows and columns (e.g deleting a row, swapping two rows etc) no additional properties are required for R. Once one starts talking about operations on matrices like addition or multiplication, the type of elements R has to be a ring. The library provides all the basic operations and their properties, the notions of determinant and inverse. The appropriate canonical structure of (non-commutative) ring is defined for matrices with addition and multiplication. Notations are provided for all definitions: a matrix is defined using \matrix, addition and multiplication of two matrices is +m and *m respectively, multiplication of a matrix by a scalar is *m: and the determinant is \det. A detailed description of the matrix library can be found in [3].

2.2 Coq's standard real numbers

The standard library of Coq provides an axiomatic definition of the real numbers. The formalization is based on 17 axioms which introduce the reals as a

complete, archimedean, ordered field that satisfies the least upper bound principle. This choice of implementation has as a positive effect that we can treat real numbers in a manner close to classical textbook mathematics. In particular, we can reason on cases thanks to the trichotomy axiom: for two real numbers x, y exactly one of the following relations hold: $x < y$, $x = y$ or $x > y$. This gives the classical flavor of the library. We note that there exists a Coq library on constructive real analysis, C-CoRN [6], where only intuitionistic logic is used and where trichotomy is not provided.

The standard library contains a lot of results for real analysis concerning: sequences and series, transcendental function, concepts of limit, continuity, differentiation, integration, calculus theorems like the mean value theorem, the fundamental theorem of calculus etc.

Though the library is well suited for expressing real analysis proofs, the real numbers defined here carry no (or little) computational meaning: since addition is just a parameter with some properties given by axioms (associativity, commutativity, etc.) there is no actual algorithm for adding two real numbers. This is a limitation as, often, in proofs on real numbers, we use computations to establish a certain property. To be able to integrate real number computations to proof assistants there has been a considerable effort put in the design and implementation of exact real number arithmetic libraries [18, 25, 20].

2.3 Multivariate analysis

For our work of formalizing multivariate analysis concepts we are in the following settings: Coq with the SSReflect extension and libraries, `Reals` library (which brings a series of axioms for the real numbers and classical logic), the axiom of extensionality and the axiom of choice. We have explained above our decision to use SSReflect and the libraries developed on it. The classical logic settings are imposed by the use of the `Reals` library. Also it makes all reasoning closer to the one in "paper mathematics". The need for the axiom of extensionality and the axiom of choice will be explained in the following section. The description of the mathematical concepts presented here follows [22, 23, 9].

2.3.1 The structure of R

Since we are combining the standard Reals library and the SSReflect libraries we need to get them to work together well. To this end, we endow the standard reals with the algebraic structure of field, as defined in the SSReflect libraries. By using Canonical Structure for the definitions we can work with the type R as usual and have the system infer the necessary structures where they are needed. The hierarchy of algebraic structures is built on types with decidable equality and with a choice operator, so we have to begin by defining an eqType and a choiceType for R. As we noted in section 2.2, the trichotomy axiom from the standard library implies that we can reason on cases on whether two reals are equal or not. In particular, we can define a function R→R→bool that returns true if the two numbers are equal and false if they are not.

```
(* lemma derived from the trichotomy axiom *)
Lemma Req_case : forall x y: R, {x = y} + {x <> y}.
(* definition for the boolean equality function *)
Definition eqr (x y : R) : bool := match (Req_case x y) with
```

```
  |left _ ⇒ true |right _ ⇒ false end.
(* lemma proving the equivalence between boolean and Leibniz equality
    *)
Lemma eqrP : forall x y, reflect (x = y) (eqr x y).
(* the canonical mixin and type for reals with a decidable equality *)
Canonical Structure real_eqMixin := EqMixin eqrP.
Canonical Structure real_eqType := EqType real_eqMixin.
```

In order endow to R with a choiceType structure we need additional axioms in our logic, i.e. a version of the axiom of choice and the axiom of functional extensionality. The latter is needed because the choice operator on R needs to produce the same canonical element for two sets that are extensionally equal and for two proofs that the set is non-empty. Though present in our context, we limit the use of extensionality to this only instance. The axiom of choice will be necessary in some other points of our development, that we will also point out.

Now we have the base properties on R needed to define the algebraic hierarchy. We endow the real numbers with canonical structures for group, ring, commutative ring and field. These canonical structure declarations make all theorems regarding the algebraic structures directly available for the reals.

2.3.2 Vectors

For p, a positive natural number, we define Rvec p, the type of vectors of length p, as a finfun from 'I_p to R.

The vectors are coercible to functions 'I_p→R. This corresponds to the familiar way of viewing vectors as $(x_0, x_1, \ldots, x_{p-1})$, where $x_i \in \mathbb{R}, \forall i \in \{0, 1, \ldots, p-1\}$. The theorems on finfuns makes that real vectors are canonically an eqType because R is, and the Leibniz equality is equivalent to the extensional equality of the corresponding functions.

```
Lemma vecP : forall u v : Rvec p, (∀ i, u i = v i) ↔ u = v.
```

This is consistent with the way of understanding vector equality in "paper mathematics". We define the operations on our vectors like addition, subtraction or multiplication by a scalar component wise.

```
Definition add_v (u v: Rvec p) := \vec_(i) (u i + v i).
Definition dif_v (u v: Rvec p) := \vec_(i) (u i − v i).
Definition mult_sv (a: R) (v: Rvec p) := \vec_(i) (a ∗ v i).
```

The notation \vec_(i) hides the function applied to transform a function into the corresponding finfun. We also introduce notations for these operations to make the scripts more readable: $+\hat{}$, $-\hat{}$ and $*\hat{}$ respectively. Thanks to the interpretation of equality, properties of these operations are proved by simply reducing them to properties on the real numbers. For the latter we benefit from tactics like ring, field and fourier provided by Coq, which automatically solve a large variety of equalities and inequalities on the reals. We also define the vectors of the canonical basis by base_v i and the nul vector by vect0 p.

The next step is to define a norm on the vectors. We have a structure that defines a norm as an application from Rvec p to R that respects: positive definedness, positive homogeneity and triangle inequality. As a concrete norm on vectors we choose the following: $\|v\| = \max_i |v_i|$ to respect the conventions in [9].

Nevertheless, the structure of the proof does not depend on the particular norm used and it can be adapted to any other. Defining this norm is straightforward using the library on indexed operations:

```
Definition norm (v: Rvec p) := \big[Rmax/0]_(i<p) Rabs (v i).
```

Proving the good properties for the norm is done by using properties already proved for \big and induction on the structure of the list of indexes. For example, a lemma stating the positivity of the norm

```
Lemma norm_pos : forall v, 0 ≤ norm v.
```

can easily proved by applying a generic lemma named big_prop. It states that a property which is closed with respect to the operator, satisfied by the nil value (here 0), and by the formula for every index is also satisfied by the complete big operation. In this case, the property is positiveness.

Nevertheless, the use of the maximum as an indexed operation posed some difficulties. As stated before, the lemmas on big operations are organized in a sort of hierarchy following the algebraic structure given by the operator. In the case of the maximum, we have associativity and commutativity, but we do not have a neutral element on the type of real numbers. Since we work only with positive numbers (and the maximum on this subset has 0 for neutral element), we would like to be able to use the lemmas that deal with an abelian monoid structure, as we know that this is the case on the subset we work on.

There are two possible solutions for this problem. The first is to have a new type for positive reals. We can define the canonical structure of abelian monoid on this new type, manipulate the indexed operation as desired and inject the result in the original type. The second solution is to define a new operator that gives the type the desired structure. This operator has to be equal to the original one on the target subset (here, the positive reals). We can then move freely between the two operators thanks to the lemmas in the indexed operations library. We adopted this second approach, as we had a construction at hand:

$$\max' \; x \; y = \begin{cases} \max \; x \; y & \text{if } x \vee y > 0; \\ \min \; x \; y & \text{if } x \wedge y \leq 0 \end{cases}$$

Basic lemmas on the norm are proved by moving to this equivalent operator and using the indexed operation library.

The norm induces the corresponding distance:

```
Definition dist_Rp u v := norm (u −^ v).
```

The properties for the distance follow naturally from those of the norm to ensure that, in our representation, $\mathbb{R}^p$, equipped with the above defined distance, is a metric space.

2.3.3 Sequences and functions in a metric space

We remark that in Coq's standard library there is a definition for a metric space:

```
Structure Metric_Space: Type := Build_Metric_Space {
  Base: Type;
  dist: Base → Base → R;
```

```
  dist_pos: forall x y: Base, dist x y ≥ 0;
  dist_sym: forall x y: Base, dist x y = dist y x;
  dist_refl: forall x y: Base, dist x y = 0 ↔ x = y;
  dist_tri: forall x y z: Base, dist x y ≤ dist x z + dist z y}.
```

However, there are no properties proved on a general metric space. This structure is only used to define the limit in a point of a function between two metric spaces. The definition is then instantiated for the real numbers and all results on limits are established in the special case of a real function. Also, convergence of sequences and Cauchy criterion are defined just for sequences of real numbers, without using the Metric_Space structure.

To have a more homogeneous formalization and to avoid duplication of proofs we define all these concepts and prove the corresponding properties in a general metric space. For a more comfortable use of the structure, we first declare a coercion from a Metric_Space to its Base type and make the Metric_Space an implicit argument of dist.

```
Definition Base_to_Type (X: Metric_Space): Type := Base X.
Coercion Base_to_Type: Metric_Space >-> Sortclass.
Implicit Arguments dist[m].
```

Then we declare the corresponding Canonical Structures for the metric spaces R with distance $|x-y|$ (or, in Coq Rabs (x−y)) and Rvec p with distance dist_Rp defined above.

```
Canonical Structure metricSpace_R :=
  Build_Metric_Space R Rabs ...
Canonical Structure metricSpace_Rp p :=
  Build_Metric_Space (Rvec p) (@dist_Rp p) ...
```

These constructs (coercions, implicit arguments and canonical structures) will help us automatically infer the Metric_Space structure from the context and make the script more readable.

We prove general properties for the distance operator and we prove that all metric spaces are separated spaces (or Hausdorff spaces). We consider sequences in metric spaces. We can talk about convergence and Cauchy criterion for these sequences:

```
Definition conv (Ms: Metric_Space) (xn: nat → Ms) (l: Ms) :=
  forall eps: R, 0 < eps → exists N: nat,
    (forall n:nat, N ≤ n → dist (xn n) l < eps).
Definition Cauchy_crit (Ms: Metric_Space) (xn: nat → Ms) :=
    forall eps: R, 0 < eps → exists N : nat,
        (forall n m: nat, N ≤ n → N ≤ m → dist (xn n) (xn m) < eps).
```

In all metric spaces, the limit of a sequence is unique and convergent sequences satisfy Cauchy's criterion. However, it is not always the case that a metric space is complete (complete metric space = all Cauchy sequences are convergent). We prove completeness in the case of the metric space $\mathbb{R}^p$ and we also prove that convergence in $\mathbb{R}^p$ is a convergence on components.

For the study of functions, we describe the concepts of limit of a function in a point and continuity of a function in a point:

```
Definition limit (X X': Metric_Space) (f: X → X') (x0: X) (l: X') :=
```

```
  forall eps, eps > 0 → exists del, del > 0 ∧
    (forall x, 0 < dist x x0 < del → dist (f x) l < eps).
Definition cont (X X': Metric_Space) (f: X → X') (x0: X) :=
  forall eps, eps > 0 → exists del, del > 0 ∧
    (forall x, dist x x0 < del → dist (f x) (f x0) < eps).
```

In the special case of $\mathbb{R}^p$ our development contains the following results: the limit of the sum of two functions is the sum of the two limits, the limit in $\mathbb{R}^p$ is unique, relations between convergence and continuity in $\mathbb{R}^p$, the limit of a function is a limit on components.

2.3.4 Matrices: norms and invertibility

For concepts on matrices we used the library described in section 2.1. Though most of the results on matrices were already available, we needed extra work in two cases. First, we needed to define the multiplication between a matrix and a vector. The SSReflect library on matrices defines multiplication for the general case $A * B, A \in M_{n\times p}(R), B \in M_{p\times q}(R)$. We gave a separate definition for the multiplication between a matrix and a vector:

```
Definition mult_mv (A: 'M_p) (v:Rvec p) := \vec_(i<p) (\sum_j A i j * v j).
```

We also needed specific results for matrices of real numbers, in particular canonical norms for matrices. A canonical matrix norm is defined using the following structure:

```
Structure can_norm: Type := MNorm {
  anorm: 'M_p → R;
  pos_norm: forall A, 0 ≤ anorm A;
  hom_pos_norm: forall A r, anorm (r *m: A) = Rabs r * anorm A;
  trin_ineq_norm: forall A B, anorm (A +m B) ≤ anorm A + anorm B;
  prod_ineq_norm: forall A B, anorm (A *m B) ≤ anorm A * anorm B;
  can_norm_abs: forall (A: 'M_p) i j, Rabs (A i j) ≤ anorm A;
  can_norm_compat: forall (A B: 'M_p), (forall i j, Rabs (A i j) ≤ Rabs (B i j))
                                       → anorm A ≤ anorm B }.
Coercion anorm: can_norm >-> Funclass.
```

Properties on the matrix norm can be related to the regularity of a matrix:

$$\|A\| < 1 \Rightarrow \det(E_p - A) \neq 0$$

where E_p denotes the identity square matrix of dimension p.

In the remainder of this section we will explain what was formalized in order to prove this result.

We begin by introducing concepts on sequences and series of matrices.

Given a sequence of matrices $A_k = \left[a_{ij}^{(k)}\right]$, $(k = 1, 2, \ldots)$ we define the limit of this sequence

$$A = \lim_{k\to\infty} A_k = \left[\lim_{k\to\infty} a_{ij}^{(k)}\right]$$

We get the following relation between canonical matrix norms and convergence:

$$\lim_{k\to\infty} A_k = A \Leftrightarrow \lim_{k\to\infty} \|A - A_k\| = 0$$

Series of matrices are defined as

$$\sum_{k=1}^{\infty} A_k = \lim_{N\to\infty} \sum_{k=1}^{N} A_k$$

If the above limit exists, the series is called convergent. A series of matrices is absolutely convergent if the series

$$\sum_{k=1}^{\infty} |A_k| = \left[\sum_{k=1}^{\infty} \left|a_{ij}^{(k)}\right|\right]$$

is convergent.
We get the following relation between norm and absolute convergence:

$$\sum_{k=1}^{\infty} \|A_k\| \ \textit{convergent} \Rightarrow \sum_{k=1}^{\infty} A_k \ \textit{absolutely convergent}$$

For the special case when the series is of the form $\sum_{k=1}^{\infty} A^k$ we get that the series converges if $\|A\| < 1$.
We consider the identity

$$(E_p + A + A^2 + \ldots + A^k)(E_p - X) = E_p - A^{k+1}$$

Passing at the limit in this identity gives

$$S(E_p - A) = E_p, \text{ where } S = \sum_{k=1}^{\infty} A^k$$

Therefore

$$\det S * \det(E_p - A) = \det E_p = 1$$

and we conclude

$$\det(E_p - A) \neq 0$$

To summarize, we have accomplished our goal and proved

```
Lemma matr_inv_norm: forall A, norm_m A < 1 → \det (1 − A) <> 0.
```

All the above definitions and results are formalized using an abstract canonical norm for matrices. We instantiate this abstract norm to :

$$\|A\| = \max_i \sum_j |a_{ij}|$$

We prove that we indeed have a canonical norm and this matrix norm is compatible to the vector norm:

$$\|Av\| \leq \|A\|\|v\|$$

The SSReflect libraries on matrices and indexed operations [2] played a central role in our development on matrices.

2.3.5 Derivation

We define partial derivatives for functions f : Rvec p→R.

```
Definition part_deriv_pt_1 (f : Rvec p → R)(a : Rvec p)(i : 'I_p)(dp : R) : Prop :=
  limit (fun t => (f (a +^ t *^ (base_v i)) − f a) / t) 0 dp.
Definition dbl_pt_1 (f : Rvec p → R)(i : 'I_p)(a : Rvec p):=
  {dp & part_deriv_pt_1 f a i dp}.
Definition dp_1 f i a (pr: dbl_pt_1 f i a) := projT1 pr.
```

The definition part_deriv_ pt_1 expresses that the function f is partially derivable at a point a with respect to the i-th component and the value of the partial derivative is dp using the concept of limit on real functions.

$$dp = \lim_{t \to 0} \frac{f(a + t \cdot e_i) - f(a)}{t} := \frac{\partial f(a)}{\partial x_i}$$

where $e_i = (0, \ldots, 0, 1, 0, \ldots, 0)$ is the i-th vector of the canonical base. The definition dbl_pt_1 describes a real number db with the property of being the value of the i-th partial derivative of f in a and the definition db_1 gives this real number.

In the same manner we define partial derivatives for functions Rvec p→Rvec p. Second order partial derivatives are defined as the derivative of the first order derivative, so the definition takes in argument a proof that the first order partial derivatives are partially derivable.

```
Definition part_deriv_pt_1_2 f a i j (pr1: forall v, dbl_pt_1 f i v) dp2 :=
 part_deriv_pt_1 (fun v => dp_1 f i v (pr1 v)) a j dp2.
```

We show basic properties of the derivation operator like linearity. We also relate the different notions between them and to derivation in one dimension. For instance, a function that has second order partial derivatives will trivially have first order partial derivatives; the partial derivative of a vectorial function is the vector of the partial derivatives of the component functions. An elegant example of a "paper" proof is the following:

$$\begin{aligned} f(x_1, \ldots, x_p) \quad - \quad & f(y_1, \ldots, y_p) = f(x_1, \ldots, x_p) - f(y_1, x_2, \ldots, x_p) + \\ + \quad & f(y_1, x_2, \ldots, x_p) - f(y_1, y_2, x_3, \ldots, x_p) + \ldots + \\ + \quad & f(y_1, \ldots, y_{p-1}, x_p) - f(y_1, \ldots, y_p) = \\ = \quad & \sum_{i=1}^{p} (x_i - y_i) \frac{\partial f(y_1, \ldots, y_{i-1}, c_i, x_{i+1}, \ldots, x_p)}{\partial x_i} \end{aligned}$$

It is also an example of the implicit or intuitive reasoning a human reader makes to replace the ... or to realize that the indexes $i - 1$, $i + 1$ are only used where they make sense. Another implicit view is interpreting the difference $f(x_1, \ldots, x_p) - f(y_1, x_2, \ldots, x_p)$ of a vector function varying in the first argument as a real function. All these are non-trivial reasoning steps for a mechanized system.

The most relevant result we needed for Kantorovitch's theorem is Taylor's formula for functions of class $C^{(2)}$. The statement and the proof are as follow:

Lemma 1 (Taylor second degree) *Let $f : \mathbb{R}^p \to \mathbb{R}$ be twice partially derivable with continuous first and second partial derivatives, then for all $a \in \mathbb{R}^p$*

and $v \in \mathbb{R}^p$ *there exists* $c \in (a, a+v)$ *so that* $f(a+v) = f(a) + \sum_{i=1}^{p} \frac{\partial f(a)}{\partial x_i} v_i + \frac{1}{2!} \sum_{i,j=1}^{p} \frac{\partial^2 f(c)}{\partial x_i \partial x_j} v_i v_j$.

Proof. Consider

$$g : [0,1] \to \mathbb{R},\ g(t) = f(a+tv)$$

then g is twice derivable on $[0,1]$ and

$$g'(t) = \sum_{i=1}^{p} \frac{\partial f(a+tv)}{\partial x_i} v_i \tag{2}$$

$$g''(t) = \sum_{i,j=1}^{p} \frac{\partial^2 f(a+tv)}{\partial x_i \partial x_j} v_i v_j \tag{3}$$

From the Taylor formula in one dimension we get that there exists $\eta \in (0,1)$ so that

$$g(1) = g(0) + g'(0) + \frac{1}{2!} g''(\eta)$$

which gives us the desired result for $c = a + t\eta \in (a, a+v)$.

The proof of this theorem is based on the proof of the Taylor formula in one dimension, which we also formalized. Also, an important issue for this proof is to show some relations between various concepts of differentiability, i.e. to prove equalities (2) and (3).

We present an excerpt of code: the definition for sums of first and second order partial derivatives that appear in Taylor's formula as well as the statement of Lemma 1:

```
Definition sum_dp_v f a v (pr : forall i x, dbl_pt_1 f i x) :=
  \sum_i dp_1 f i a (pr i a) * v i. (* = ∑_{i=1}^{p} ∂f(a)/∂x_i v_i *)
Definition sum_dp2_v
  f a v (pr: forall i x, dbl_pt_1 f i x) (pr2: forall i j, dbl_pt_1_2 f i j a (pr i)) :=
  \sum_j \sum_i dp_1_2 f i j a (pr i) (pr2 i j) * v i * v j. (* = ∑_{i,j=1}^{p} ∂²f(c)/∂x_i∂x_j v_i v_j *)

Lemma Taylor_2p:
  forall f a v pr pr2,
  (forall i v , cont (fun x: Rvec p ⇒ dp_1 f i x (pr i x)) v) →
  (forall i (v0:Rvec p) ,
    cont (fun x: Rvec p ⇒ dp_1 (fun w: Rvec p ⇒ sum_dp_v f w v pr) i x prs v0)
         →
  exists c, f (a+^v) − f a − sum_dp_v f a v pr = 1/2 * sum_dp2_v f c v pr (pr2 c
       ).
```

In the statement of the lemma we have pr, pr2 which are the proofs that the function is once and twice derivable. The following two hypotheses say that the first and second order derivatives are continuous.

3 Paper proofs vs. formal proofs

This section brings us to our original goal, the formalization of Kantorovitch's theorem which gives sufficient conditions for the convergence of Newton's method towards the root of a given function and establishes the unicity of this root in a certain domain. A version of this theorem as well as results concerning the speed for the convergence of the process and its stability are discussed in [9]. Preliminary results around a formalization of these theorems inside the Coq proof assistant are described in [26]. At present all the theorems listed in this section are verified in the Coq proof assistant.

We present here the theorems in the case of a function with one variable. This one dimensional case is a simplified model of the problem. The interest of treating it separately is to allow a better understanding for the structure of the proof. The one dimensional case reveals the key points of the proof as well as the places where the reasoning in the paper proof is difficult to pass on a machine.

Theorem 2 (Convergence) *Consider an equation $f(x) = 0$, where $f :]a, b[\rightarrow \mathbb{R}$, a, $b \in \mathbb{R}$ $f(x) \in C^{(1)}(]a, b[)$. Let $x^{(0)}$ be a point contained in $]a, b[$ with its closed ε-neighborhood $\overline{U_\varepsilon}(x^{(0)}) = \{|x - x^{(0)}| \leq \varepsilon\} \subset]a, b[$. If the following conditions hold:*

1. *$f'(x^{(0)}) \neq 0$ and $|\frac{1}{f'(x^{(0)})}| \leq A_0$;*
2. *$|\frac{f(x^{(0)})}{f'(x^{(0)})}| \leq B_0 \leq \frac{\varepsilon}{2}$;*
3. *$\forall x, y \in]a, b[, |f'(x) - f'(y)| \leq C|x - y|$*
4. *the constants A_0, B_0, C satisfy the inequality $\mu_0 = 2A_0B_0C \leq 1$.*

then, for an initial approximation $x^{(0)}$, the Newton process

$$x^{(n+1)} = x^{(n)} - \frac{f(x^{(n)})}{f'(x^{(n)})},\ n = 0, 1, 2, \ldots \tag{4}$$

converges and $\lim_{n\rightarrow\infty} x^{(n)} = x^*$ is a solution of the initial system, so that $|x^* - x^{(0)}| \leq 2B_0 \leq \varepsilon$.

Theorem 3 (Uniqness) *Under the conditions of Theorem 2 the root x^* of the function f is unique in the interval $[x^{(0)} - 2B_0, x^{(0)} + 2B_0]$.*

Theorem 4 (Speed of convergence) *Under the conditions of Theorem 2 the speed of the convergence of Newton's method is given by*

$$|x^{(n)} - x^*| \leq \frac{1}{2^{n-1}}\mu_0^{2^n-1}B_0$$

Theorem 5 (Local stability) *If the conditions of Theorem 2 are satisfied and if, additionally, $0 < \mu_0 < 1$ and $[x^{(0)} - \frac{2}{\mu_0}B_0, x^{(0)} + \frac{2}{\mu_0}B_0] \subset]a, b[$, then for any initial approximation $x'^{(0)}$ that satisfies $|x'^{(0)} - x^{(0)}| \leq \frac{1-\mu_0}{2\mu_0}B_0$ the associated Newton's process converges to the root x^*.*

The theorem of convergence of Newton's method shows that this method is indeed appropriate for determining the root of the function. The unicity of the solution in a certain domain is used in practice for isolating the roots of the function. The result on the speed of the convergence means we know a bound for the distance between a given element of the sequence and the root of the function. This distance represents the precision at which an element of the sequence approximates the root. In practice this theorem is used to determine the number of iterations needed in order to achieve a certain precision for the solution. The result on the stability of the method helps with efficiency issues as it allows the use of an approximation instead of the exact value as we shall see in section 3.1.

We do not present here the proofs of the theorems, we just give a few elements of these proofs that will help understand how paper proofs relate to formal proofs and how formalized proofs can help discover new proofs. For detailed proofs we refer the reader to [9]. The outline of the proof for theorem 2 as presented in [9] is as follows:

- prove a collection of properties for each element of the Newton sequence;
- infer that it is a Cauchy sequence;
- use the completeness of $\mathbb{R}$ to prove the convergence;
- prove that the limit of the sequence is a root of the given function.

The proof introduces the auxiliary sequences $\{A_n\}_{n\in\mathbb{N}}$, $\{B_n\}_{n\in\mathbb{N}}$ and $\{\mu_n\}_{n\in\mathbb{N}}$:

$$A_n = 2A_{n-1} \tag{5}$$

$$B_n = A_{n-1}B_{n-1}^2C = \frac{1}{2}\mu_{n-1}B_{n-1} \tag{6}$$

$$\mu_n := 2A_nB_nC = \mu_{n-1}^2 \tag{7}$$

For each element of the Newton sequence, we are able to verify properties that are similar to those for $x^{(0)}$. Reasoning by induction we get the following:

$$\overline{U}_{\varepsilon}(x^{(0)}) \supset \overline{U}_{\frac{\varepsilon}{2}}(x^{(1)}) \supset \ldots \supset \overline{U}_{\frac{\varepsilon}{2^n}}(x^{(n)}) \supset \ldots \tag{8}$$

furthermore

$$f'(x^{(n)}) \neq 0 \text{ and } \left|\frac{1}{f'(x^{(n)})}\right| \leq A_n \tag{9}$$

$$\left|\frac{f(x^{(n)})}{f'(x^{(n)})}\right| \leq B_n \leq \frac{\varepsilon}{2^{n+1}} \tag{10}$$

$$\mu_n \leq 1 \tag{11}$$

Notice that hypothesis 3. is a property of the function and it does not depend on the elements of Newton's sequence.

From (8) we can infer that $x^{(n)}$ is a Cauchy sequence:

$$x^{(n+m)} \in \overline{U}_{\frac{\varepsilon}{2^n}}(x^{(n)}) \Rightarrow \|x^{(n+m)} - x^{(n)}\| \leq \frac{\varepsilon}{2^n}$$

The latter quantity can be made arbitrary small for $n > N$ and $m \in \mathbb{N}$, which is equivalent to Cauchy's criterion. We use the result that $\mathbb{R}$ is a complete metric space to deduce that the sequence converges. By taking the limit in (4) we get that the limit of the sequence is a root of function f.

To prove the uniqueness of the solution, we suppose that there exists another solution of the equation and prove that it is also the limit of the sequence. By uniqueness of this limit we have the desired result.

For Theorem 5 (local stability) we prove that the new initial approximation $x'^{(0)}$ satisfies similar hypotheses as those for $x^{(0)}$. The new constants are $A' = \frac{4}{3+\mu_0}A_0$ and $B' = \frac{3+\mu_0}{4\mu_0}B_0$. This makes that $\mu' = 2A'B'C = 1$ and we can verify that

- $f'(x'^{(0)}) \neq 0$ *and* $|\frac{1}{f'(x'^{(0)})}| \leq A'$
- $|f(x'^{(0)})/f'(x'^{(0)})| \leq B'$
- $\mu' \leq 1$

We are thus in the hypotheses of Theorem 2 and by applying this theorem we conclude that the process converges to the same root x^*.

Notice, however, that for the new constants we get $\mu' = 1$. If we do a Newton iteration, we would get the new $\mu'' = \mu'^2 = 1$ (cf. equation (7)) and we would not be able to do an approximation again, because Theorem 5 requires $\mu'' < 1$. To correct this, we impose a finer approximation $|x_0 - x'_0| \leq \frac{(1-\mu_0)}{4\mu_0}B_0$. This new approximation yields the following formulas for the constants:

$$A' = \frac{8}{7+\mu_0}A_0 \tag{12}$$

$$B' = \frac{\mu_0^2 + 46\mu_0 + 17}{8(7+\mu_0)\mu_0}B_0 \tag{13}$$

this makes that

$$\mu' = \frac{\mu_0^2 + 46\mu_0 + 17}{(7+\mu_0)^2} < 1 \tag{14}$$

We summarize these results in:

Corollary 1 *If the conditions of Theorem 2 are satisfied and if, additionally,* $0 < \mu_0 < 1$ *and* $[x^{(0)} - \frac{2}{\mu_0}B_0, x^{(0)} + \frac{2}{\mu_0}B_0] \subset]a, b[$*, then for any initial approximation* $x'^{(0)}$ *that satisfies* $|x'^{(0)} - x^{(0)}| \leq \frac{1-\mu_0}{4\mu_0}B_0$ *the associated Newton's process converges to the root* x^**.*

3.1 Newton's method with rounding

We now have all the necessary tools to state and prove a theorem on the behavior of Newton's method if we consider rounding at each step. The rounding we do is just good enough to ensure the convergence. This theorem is particularly interesting for computations in arbitrary or multiple precision, as it relates number of iterations with the precision of the input and that of the result. This

means that for the first iterations we need a lower precision, as we are not close to the root. We will later increase the precision of our input with the desired precision for the result.

Theorem 6 (Convergence with rounding) *We consider a function $f :]a, b[\to \mathbb{R}$ and an initial approximation $x^{(0)}$ satisfying the conditions in Theorem 2. We also consider a function $rnd : \mathbb{N} \times \mathbb{R} \to \mathbb{R}$ that models the approximation we will make at each step in the perturbed Newton sequence:*

$$t^{(0)} = x^{(0)} \text{ and } t^{(n+1)} = rnd_{n+1}\left(t^{(n)} - \frac{f(t^{(n)})}{f'(t^{(n)})}\right)$$

If

1. $\forall n \forall x, x \in]a, b[\Rightarrow rnd_n(x) \in]a, b[$
2. $\frac{1}{2} \leq \mu_0 < 1$
3. $[x^{(0)} - 3B_0, x^{(0)} + 3B_0] \subset]a, b[$
4. $\forall n \forall x, |x - rnd_n(x)| \leq \frac{1}{3^n} R_0$, *where* $R_0 = \frac{1-\mu_0^2}{8\mu_0} B_0$

then

a. *the sequence $\{t^{(n)}\}_{n\in\mathbb{N}}$ converges and $\lim_{n\to\infty} t^{(n)} = x^*$ where x^* is the root of the function f given by Theorem 2*
b. $\forall n, |x^* - t^{(n)}| \leq \frac{1}{2^{n-1}} B_0$

The first hypothesis makes sure that the new value will also be in the range of the function. The second and third hypotheses come from the use of the stability property of the Newton sequence (see Corollary 1). The fourth hypothesis controls the approximation we are allowed to make at each iteration. The conclusion gives us the convergence of the process to the same limit as Newton's method without approximations. Also we give an estimate of the distance from the computed value to the root at each step.

Proof. Our proof is based on those for theorems 2 - 5 and corollary 1. To give the intuition behind the proof, we decompose Newton's perturbed process $t^{(n)}$ as follows:

1. set $t^{(0)} := x^{(0)}$
2. do a Newton iteration to get $x^{(1)} := t^{(0)} - \frac{f(t^{(0)})}{f'(t^{(0)})}$
3. do an approximation of the result to get $t^{(1)} := rnd(x^{(1)})$
4. set $t^{(0)} := t^{(1)}$ and go to step 2.

Now let's look at these steps individually:

- At step 1. we start with the initial $x^{(0)}$ that satisfies the conditions in Theorem 2. This means that Newton's method from this initial point converges to the root x^* (cf. Theorem 2).

- At step 2. we consider a Newton sequence starting with $x^{(1)}$. This sequence is the same as the sequence at step 1. except that we "forget" the first element of the sequence and start with the second. It is trivial that this sequence converges to the root x^*. We note that (cf. proof of Theorem 2) we can associate the constants A_1, B_1 to the initial iteration of this sequence and get the corresponding hypotheses from Theorem 2.

- At step 3. we consider Newton's sequence starting from $t^{(1)}$. This initial point is just an approximation of the initial point of the previously considered sequence. From Corollary 1 we get the convergence of the new sequence to the same root x^*. Moreover, the proof of Corollary 1 gives us the constants A', B' associated to the initial point that also satisfy the hypotheses of Theorem 2. This means we can start the process over again.

If we take $x^{(0)}$ and then all the initial iterations of the sequences formed at step 3. we get back our perturbed Newton's sequence. But decomposing the problem as we did gives the intuition of why this sequence should converge. However, just having a set of sequences that all converge to the same root does not suffice to prove that the sequence formed with all initial iterations of these sequences will also converge to the same root. The reason is simple, the approximation at step 3. could bring us back to the initial point $x^{(0)}$ which would still yield a convergent Newton's sequence, but which would not make the new element of the perturbed sequence any closer to the root than the previous one. To get the convergence of the perturbed sequence we need to control the approximation we make. We will see in what follows that hypothesis 4. suffices to ensure the convergence of the new process.

To make the intuitive explanation more formal we consider the sequence of sequences of real numbers $\{Y_p\}_{p\in\mathbb{N}}$ defined as follows:

$Y_0^n = x^{(n)}$ is the original Newton's sequence;

Y_1 is given by

$Y_1^0 = rnd_1(x^{(1)})$;

$Y_1^{n+1} = Y_1^n - f(Y_1^n)/f'(Y_1^n)$ is the Newton's sequence associated to the initial iteration Y_1^0;

we continue in the same manner and for an arbitrary p we define Y_p as follows

$Y_{p+1}^0 = rnd_{p+1}(Y_p^1)$;

$Y_{p+1}^{n+1} = Y_{p+1}^n - f(Y_{p+1}^n)/f'(Y_{p+1}^n)$.

We notice that taking the first element in each of these sequences forms our perturbed Newton's process:

$$Y_0^0 = x^{(0)} = t^{(0)} \text{ and}$$

$$Y_{n+1}^0 = rnd_{n+1}(Y_n^0 - f(Y_n^0)/f'(Y_n^0)) = rnd_{n+1}(t^{(n)} - f(t^{(n)})/f'(t^{(n)})) = t^{(n+1)}$$

Following our plan, we now show that for each p the sequence $\{Y_p^n\}_{n\in\mathbb{N}}$ converges to x^* and ensures a certain bound in the error.

- We start with sequence $\{Y_0^n\}_{n\in\mathbb{N}}$. Since it coincides with the initial sequence, the properties from Theorem 2 are trivially satisfied. For the initial point Y_0^0 we have the associated constants A_0, B_0. Applying Theorem 2 we get that $\lim_{n\to\infty} Y_0^n = x^*$ and $|x^* - Y_0^0| \leq 2B_0$.

- Before considering $\{Y_1^n\}_{n\in\mathbb{N}}$, we note that the sequence $\overline{Y}_0^n = Y_0^{n+1}$ (i.e. the previously considered sequence where we start from the second element) also satisfies the conditions, with initial point $\overline{Y}_0^0 = Y_0^1$ and constants $\overline{A}_0 = 2A_0$ and $\overline{B}_0 = A_0B_0^2C$. The laws for these constants are deduced from relations (5), (6). We get that $\lim_{n\to\infty} \overline{Y}_0^n = x^*$ and $|x^* - \overline{Y}_0^0| = |x^* - Y_0^1| \leq 2\overline{B}_0 = 2(A_0B_0^2C)$.

- Now we consider $\{Y_1^n\}_{n\in\mathbb{N}}$. The initial point of this sequence is $Y_1^0 = rnd_1(Y_0^0 - f(Y_0^0)/f'(Y_0^0)) = rnd_1(\overline{Y}_0^0)$. We are in the situation of Corollary 1, where we have a converging sequence ($\{\overline{Y}_0^n\}_{n\in\mathbb{N}}$) and we introduce an approximation in the initial iteration. To be able to apply this corollary we need to verify $0 < \overline{\mu}_0 < 1$, $[\overline{Y}_0^0 - \frac{2}{\overline{\mu}_0}\overline{B}_0, \overline{Y}_0^0 + \frac{2}{\overline{\mu}_0}\overline{B}_0] \subset]a, b[$ and $|rnd_1(\overline{Y}_0^0) - \overline{Y}_0^0| \leq \frac{1-\overline{\mu}_0}{4\overline{\mu}_0}\overline{B}_0$.
  We will show later on that under our hypotheses these three conditions are indeed verified. From Corollary 1 we get the new constants according to relations (12), (13). This makes that we find ourselves again in the conditions of Theorem 2 and we can deduce that $\lim_{n\to\infty} \overline{Y}_1^n = x^*$ and $|x^* - Y_1^0| \leq 2B' = 2\frac{\overline{\mu}_0^2+46\overline{\mu}_0+17}{8(7+\overline{\mu}_0)\overline{\mu}_0}\overline{B}_0$.

We are in the appropriate conditions to start this process again and explain in the same manner the properties for $\{Y_2^n\}_{n\in\mathbb{N}}$, $\{Y_3^n\}_{n\in\mathbb{N}}$, etc. The auxiliary sequences are given by the following relations:

$$A'_0 = A_0 \text{ and } A'_{n+1} = \frac{8}{7 + \overline{\mu}_n(2A'_n)}$$

$$B'_0 = B_0 \text{ and } B'_{n+1} = \frac{{\overline{\mu}_n}^2 + 46\overline{\mu}_n + 17}{8(7+\overline{\mu}_n)\overline{\mu}_n}(A'_n {B'_n}^2 C)$$

$$\overline{\mu}_n = 2(2A'_n)(A'_n {B'_n}^2 C)C = (2A'_n B'_n C)^2$$

we also consider

$$\mu'_{n+1} = 2A'_{n+1}B'_{n+1}C = \frac{\overline{\mu}_n^2 + 46\overline{\mu}_n + 17}{(7+\overline{\mu}_n)^2} = \frac{{\mu'_n}^2{}^2 + 46{\mu'_n}^2 + 17}{(7+{\mu'_n}^2)^2}$$

$$R_n = \frac{1-\overline{\mu}_n}{4\overline{\mu}_n}\overline{B}_n = \frac{1-{\mu'_n}^2}{4{\mu'_n}^2}(\frac{1}{2}\mu'_n B'_n) = \frac{1-{\mu'_n}^2}{8\mu'_n}B'_n$$

Using the above reasoning steps, we get by induction that $|Y_n^0 - x^*| \leq 2B'_n$ and we also manage to show $\forall n, B'_{n+1} \leq \frac{1}{2}B'_n \leq \frac{1}{2^{n-1}}B_0$. The latter relations is deduced from the above formulas by basic manipulations. It trivially implies the convergence of the perturbed sequence to the root x^*.

We need some auxiliary results to ensure that Corollary 1 is applied in the appropriate conditions each time we make a rounding. These results are as follow:

- $0 < \frac{1}{2} \leq \mu_0 = \mu'_0 \leq \mu'_n \leq \mu'_{n+1} \leq \ldots < 1$
- $R_{n+1} \leq \frac{1}{3}R_n \leq \ldots \leq \frac{1}{3^n}R_0 = \frac{1}{3^n}\frac{1-\overline{\mu}_0}{4\overline{\mu}_0}\overline{B}_0 = \frac{1}{3^n}\frac{1-\mu_0^2}{8\mu_0}B_0$

- $|Y^0_{n+1} - Y^0_n| \leq \frac{1}{2^n} B_0 + \frac{1}{3^n} R_0$
- $[\overline{Y}^0_n - \frac{2}{\overline{\mu}_n} \overline{B}_n, \overline{Y}^0_n + \frac{2}{\overline{\mu}_n} \overline{B}_n] \subseteq [Y^0_0 - 3B_0, Y^0_0 + 3B_0] \subset]a, b[$

We do not discuss all the details as they are elementary reasoning steps concerning inequalities, second degree equations or geometric series. All these results have been formalized in Coq to ensure that no steps are overlooked.

Remarks
This proof of convergence of Newton's method with rounding has an interest from a proof engineering point of view. We were able to come up with the proof because we had formalized theorems 2 - 5 inside a proof assistant. Such a formalization forces the user to understand the structure of the proof on one hand and to handle details with care on the other. Thus, an assisted proof is usually more structured and more detailed than a paper proof (especially in domains where automatic techniques are difficult to implement, like real analysis). For example, while on paper the auxiliary sequences $\{A_n\}_{n\in\mathbb{N}}$, $\{B_n\}_{n\in\mathbb{N}}$ appear during the proof, on the computer they are defined appart from the proof, allowing the user to better understand their importance and use similar sequences in the new proof. A proof assistant is also helpful with syntactic aspects like properly constructing the induction hypothesis and doing the bookkeeping to make sure all needed details are taken into consideration.

3.2 Formal proof in the one dimensional case

In this section we will show how we formalized the theorems from the previous section inside the Coq proof assistant. We first discuss the issues raised by working with derivatives inside a proof assistant.

3.2.1 Derivation

Let $f : \mathbb{R} \to \mathbb{R}$ be a derivable function on $[a, b]$. "On paper" we can write the corresponding Newton's sequence $x^{(n+1)} = x^{(n)} - \frac{f(x^{(n)})}{f'(x^{(n)})}$, without worrying whether the term $x^{(n)}$ is in the interval $[a, b]$ where the function is derivable. The Coq formalization of derivatives requires that when we talk about the derivative of a function in a point, we provide a proof that the function is derivable at that point. The goal is to ensure that derivatives are properly used. In the case if Newton's sequence, for writing the definition of the sequence in Coq we would have to provide a proof that f is derivable in $x^{(n)}$. We can prove this for every n, but we need to define the sequence before being able to do this proof.

We worked around this impediment by defining a total function f' to use in the definition of Newton's sequence. We then imposed that on the interval $[a, b]$ f' is equal to the derivative of f. This enables us to define our sequence and at the same time prevents us from using properties of the derivative in a point before proving the function is derivable there.

3.2.2 Statements and proofs

To formalize the reasoning steps necessary for Kantorovitch's proof, we place ourselves in the context of the theorem, by expressing our working hypotheses.

In the code, the Variable declarations describe objects that are assumed to exist, the Hypothesis declarations describe properties that are assumed to hold and the Fixpoint declaration defines the Newton sequence as a recursive function.

```
Variables a b: R. (*the ends of the interval*)
Variables f f' f'': R → R. (*the function and its derivatives*)
Variables A0 B0 C: R. (*the constants from the theorem*)
Definition mu0 := 2 * A0 * B0 * C.
Variable X0: R. (*the initial approximation*)
Fixpoint Xn (n:nat): R := (*the Newton sequence*)
  match n with | 0 ⇒ X0 |S n ⇒ Xn n − f (Xn n) / f' (Xn n) end.

(*the hypothesis on the function and the constants*)
Hypothesis Hincl: a < X0 < b.
Hypothesis pr: forall t:R, o_I a b t → derivable_pt f t.
Hypothesis Hder_f: forall (t: R) (H: o_I a b t), derive_pt f t (pr t H) = f' t.
Hypothesis pr2: forall t:R, o_I A B t → derivable_pt f' t.
Hypothesis Hder_f': forall (t: R) (H: o_I a b t), derive_pt f' t (pr2 t H) = f'' t.
Hypothesis Hbound_f'': forall t (H: o_I A B t), Rabs (f'' t) ≤ C.
Hypothesis Hdif_f': f' X0 <> 0 .
Hypothesis Habs_a: Rabs (/f' X0) ≤ A0.
Hypothesis Habs_b: Rabs (f X0 / f' X0) ≤ B0 .
Hypothesis Hb_eps: included (c_disc X0 (2*B0)) (o_I a b).
Hypothesis A0_b0_c: 2*A0*B0*C ≤ 1.
```

To clarify the statements above, we mention that o_I a b and c_disc X0 (2∗B0) denote the open interval (a, b) and the closed disc centered in X0 and of radius 2B0, respectively. Hypothesis pr says that function f is derivable on interval (a, b), hypothesis Hder_f states that f' is equal to the derivative of f on the same interval, while hypotheses pr2, Hder_f' and Hbound_f'' give similar conditions for the first and second derivative.

We now start the proof by introducing the sequences $(A_n)_{n\in\mathbb{N}}$ and $(B_n)_{n\in\mathbb{N}}$ (see section 3).

```
Fixpoint An n := match n with |0 ⇒ a0 |S n ⇒ 2*An n end.
Fixpoint Bn n := match n with |0 ⇒ b0 |S n ⇒ An n*(Bn n)^2*c end.
```

We also introduce the sequence $(E_n)_{n\in\mathbb{N}}$ for $\varepsilon/2^n$ to enhance readability.

```
Fixpoint En n := match n with |0 ⇒ eps |S n ⇒ (En n)/2 end.
```

We are now able to express the reasoning made in the proof, more precisely relations (9), (10) and (8) from section 3. By induction on n we get the desired properties.

```
Lemma newton_cv_aux: forall n,
  included (c_disc (Xn n) (En n)) (c_disc (Xn (n − 1)) (En (n − 1))) ∧
  f' (Xn n) <> 0 ∧
  Rabs (/ f' (Xn n)) ≤ An n ∧
  Rabs (f (Xn n) /f' (Xn n)) ≤ Bn n ∧
  Bn n ≤ (En n)/2.
```

We are now able to deduce that $(X_n)_{n\in\mathbb{N}}$ is convergent and the limit of the sequence is the root of f in the desired domain.

```
Theorem newton_exist: {xs:R & Un_cv Xn xs ∧ c_disc X0 (2∗b0) xs ∧ f xs = 0}.
Definition xs:= projT1 kanto_exist.
```

In the domain $\{|x - x^{(0)}| \leq 2B_0\}$, the root xs is unique.

```
Theorem newton_uniq: forall xs2, c_disc X0 (2∗b0) xs2 → f xs2 = 0 → xs = xs2.
```

We can give the speed of convergence:

```
Theorem newton_speed: forall n,
  Rabs (xs − Xn n) ≤ / 2 ^ (n − 1) ∗ mu0 ^ (2 ^ n − 1) ∗ B0 .
```

and establish the local stability of Newton's method:

```
Variable X0':R. (* the new initial approximation *)
Hypothesis Hmu01: 0 < mu0 < 1. (* hypotheses from Theorem 5 *)
Hypothesis Hmudisc: included (c_disc X0 (2 / mu0 ∗ B0)) (c_I a b).
Hypothesis Hdom: Rabs (X0' − X0) ≤ (1 − mu0)/(2 ∗ mu0) ∗ B0.

Theorem newton_stable: Un_cv (Xn X0' f f') xs.
```

For Newton's method with rounding at each step we introduce a new sequence:

```
Fixpoint Tn n := match n with
  |O ⇒ X0 |S n' ⇒ let tn := Tn n' in (rnd (tn − f tn / f' tn) n) end.
```

where rnd is a rounding function with the properties described in Theorem 6. The convergence of Newton with rounding is given by:

```
Lemma newton_rnd_conv : Un_cv Tn xs.
```

3.3 Formal proof in the multidimensional case

The formalization of multivariate analysis concepts presented above was done to cover the background mathematics necessary for Kantorovitch's theorem. The actual reasoning steps follow the paper proof (see section 3) in a similar manner to that described for the real case (see section 3.2).

The formalization done for the real case was a good guide in our development as the hypothesis and lemmas needed are just a generalization of the ones for the real case. This shows how proofs done on a simple model of a problem can help reason on a complicated one, provided the appropriate level of abstraction.

The work of generalization builds up to Kantorovitch's theorem. We give a sample of the hypotheses, definitions and statements of theorems in the multidimensional case.

```
Variables A0 B0 c : R.
Variable X0 : Rvec p.
Variable f : Rvec p → Rvec p.
Variable f': Rvec p → matrix R p.
Hypothesis pr: forall v i, dbl_pt f v i. (* once derivable*)
Hypothesis Hjac: forall v, f' v = Jac f v (pr v).
Hypothesis Hinv0 : \det (f' X0) <> 0.
Hypothesis Ha0: norm_m ((f' X0)^−1m) ≤ A0.
Hypothesis Hb0: norm (mult_mv ((f' X0)^−1m) (f X0)) ≤ B0.
Fixpoint Xn n := match n with
```

```
  |0 ⇒ X0 |S n ⇒ Xn n −^ mult_mv ((f' (Xn n))^−1m) (f (Xn n)) end .
(* ... *)
```

The statement of existence of the root of the function looks like its analog in one dimension:

```
Theorem newtonRp_exist:
  exists xs, conv Xn xs ∧ c_disc_Rp X0 (2∗B0) xs ∧ f xs = vect0 p.
```

4 Conclusion

The formal development described in this paper is available online at:
`http://www-sop.inria.fr/marelle/Ioana.Pasca/research.html`
This development represents a complete formal study of the theoretical properties of a numerical method. The paper describes the formalization and tries to point out relevant issues that arise. Though the development is made in Coq, these issues are common for most proof assistants. For example, most proof assistants do not have a multivariate analysis library, with the exception of HOL Light [16]. However, most of them do have a real analysis library, comparable to that of Coq and that can serve as a basis for multivariate analysis. For the interested reader we point out the following work on real analysis: in Isabelle [10], in PVS [8], in HOL [15], in ACL2 [11]. Once embarked on the formalization of multivariate analysis we need to carefully choose a representation of vectors well suited for the type system we work in. Derivatives are also delicate to handle in a proof assistant, because we will always need to prove that a function is derivable in a point before talking about the derivative at that point. Multivariate analysis cannot be approached without talking about matrices. Here also proof assistants provide more or less complex libraries, among which we mention [5] as a matrix formalization inside ACL2, [21] and [27] as standard Coq contributions on the topic, [3] as a formalization based on the SSReflect extension of Coq that we used in our work, [24] as a matrix theory in Isabelle/HOL and [16] in HOL Light.

Once we implemented all the necessary concepts in our systems, we were able to formalize well known results concerning Newton's method and also come up with new results. To the best of the author's knowledge, the result and proof of Theorem 6 are new, though the author is not an expert in numerical analysis. Using only a predetermined precision for our computation makes it that our formalization can be seen as an (imperfect) model of computation in multiple or arbitrary precision, thus validating Newton's method in such a context.

Since we are talking about a numerical method, it seems natural to have computations with Newton's method. At present proof assistants are not very well adapted for heavy computation. Real number computations can be performed inside exact real arithmetic libraries like [18, 25, 20]. Some results around exact computation with Newton's method are discussed by Julien and the author in [19], where Theorem 6 played a crucial role, as Newton's method with rounding at each step is a lot more efficient. In the future we could also investigate the use of Theorem 6 in the validation of computation with Newton's method on floating point numbers.

Acknowledgements. I thank Yves Bertot for his help and constructive suggestions.

References

[1] Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development, Coq'Art:the Calculus of Inductive Constructions*. Springer-Verlag, 2004.

[2] Yves Bertot, Georges Gonthier, Sidi Ould Biha, and Ioana Paşca. Canonical Big Operators. In *Proceedings of the 21st International Conference on Theorem Proving in Higher Order Logics (TPHOLs 2008)*, volume 5170 of *LNCS*, August 2008.

[3] Sidi Ould Biha. Formalisation des mathématiques : une preuve du théorème de Cayley-Hamilton. In *Journées Francophones des Langages Applicatifs*, pages 1–14, 2008.

[4] Coq development team. *The Coq Proof Assistant Reference Manual, version 8.2*, 2009.

[5] John Cowles, Ruben Gamboa, and Jeff Van Baalen. Using ACL2 Arrays to Formalize Matrix Algebra. In *ACL2 Workshop*, 2003.

[6] L. Cruz-Filipe, H. Geuvers, and F. Wiedijk. C-CoRN: The Constructive Coq Repository at Nijmegen. In A. Asperti, G. Bancerek, and A. Trybulec, editors, *Mathematical Knowledge Management, Third International Conference, MKM*, volume 3119 of *LNCS*, pages 88–103. Springer-Verlag, 2004.

[7] David Delahaye and Micaela Mayero. Quantifier Elimination over Algebraically Closed Fields in a Proof Assistant using a Computer Algebra System. In *Proceedings of Calculemus 2005*, volume 151(1) of *ENTCS*, pages 57–73, 2006.

[8] B. Duerte. Elements of Mathematical Analysis in PVS. In *Proceedings of the Ninth International Conference on Theorem Proving in Higher-Order Logics (TPHOL '96)*, 1996.

[9] B. Démidovitch et I. Maron. *Éléments de calcul numérique*. Mir - Moscou, 1979.

[10] Jacques D. Fleuriot. On the mechanization of real analysis in Isabelle/HOL. In J. Harrison and M. Aagaard, editors, *Theorem Proving in Higher Order Logics: 13th International Conference, TPHOLs 2000*, volume 1869 of *Lecture Notes in Computer Science*, pages 146–162. Springer-Verlag, 2000.

[11] R. Gamboa and M. Kaufmann. Nonstandard Analysis in ACL2. *Journal of automated reasoning*, 27(4):323–428, November 2001.

[12] François Garillot, Georges Gonthier, Assia Mahboubi, and Laurence Rideau. Packaging mathematical structures. In *TPHOLs*, pages 327–342, 2009.

[13] Georges Gonthier and Assia Mahboubi. A small scale reflection extension for the coq system. INRIA Technical report, available at `http://hal.inria.fr/inria-00258384`.

[14] Georges Gonthier, Assia Mahboubi, Laurence Rideau, Enrico Tassi, and Laurent Théry. A modular formalisation of finite group theory. In *Theorem Proving in Higher-Order Logics*, volume 4732 of *LNCS*, pages 86–101, 2007.

[15] John Harrison. *Theorem Proving with the Real Numbers*. Springer-Verlag, 1998.

[16] John Harrison. A HOL Theory of Euclidian Space. In Joe Hurd and Thomas F. Melham, editors, *TPHOLs*, volume 3603 of *LNCS*, pages 114–129. Springer, 2005.

[17] John Harrison and Laurent Théry. A Skeptic's Approach to Combining HOL and Maple. *J. Autom. Reasoning*, 21(3):279–294, 1998.

[18] Nicolas Julien. Certified exact real arithmetic using co-induction in arbitrary integer base. In *Functional and Logic Programming Symposium (FLOPS)*, LNCS. Springer, 2008.

[19] Nicolas Julien and Ioana Pasca. Formal Verification of Exact Computations Using Newton's Method. In *TPHOLs 2009*, volume 5674 of *LNCS*, pages 408–423, 2009.

[20] David R. Lester. Real Number Calculations and Theorem Proving. In Otmane Aït Mohamed, César Muñoz, and Sofiène Tahar, editors, *TPHOLs*, volume 5170 of *Lecture Notes in Computer Science*, pages 215–229. Springer, 2008.

[21] Nicolas Magaud. Ring properties for square matrices. `http://coq.inria.fr/contribs-eng.html`.

[22] Mihail Megan. *Analiza Matematica*. Mirton Timisoara, 1999.

[23] Mihail Megan. *Calcul Diferential si Integral in Rp*. Mirton Timisoara, 2000.

[24] Steven Obua. Proving bounds for real linear programs in isabelle/hol. In *Theorem Proving in Higher-Order Logics*, pages 227–244, 2005.

[25] Russell O'Connor. Certified Exact Transcendental Real Number Computation in Coq. In *Theorem Proving in Higher Order Logics, 21st International Conference, TPHOLs 2008, Montreal, Canada*, pages 246–261, 2008.

[26] Ioana Paşca. A Formal Verification for Kantorovitch's Theorem. In *Journées Francophones des Langages Applicatifs*, pages 15–29, 2008.

[27] J. Stein. Documentation for the formalization of Linerar Agebra. `http://www.cs.ru.nl/~jasper/`.

Contents

Centre de recherche INRIA Sophia Antipolis – Méditerranée
2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399