HAL open science

Différentiabilité et intégrabilité en Coq. Application à la formule de d'Alembert

Catherine Lelay, Guillaume Melquiond

To cite this version:

Catherine Lelay, Guillaume Melquiond. Différentiabilité et intégrabilité en Coq. Application à la formule de d'Alembert. 2011. hal-00642206v1

HAL Id: hal-00642206

https://inria.hal.science/hal-00642206v1

Preprint submitted on 17 Nov 2011 (v1), last revised 2 Dec 2011 (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.

Différentiabilité et intégrabilité en Coq Application à la formule de d'Alembert*

C. Lelay[1] & G. Melquiond[2]

INRIA Saclay Île-de-France,
91405 Orsay cedex, France
1 : catherine.lelay@inria.fr
2 : guillaume.melquiond@inria.fr

Résumé

La bibliothèque standard de Coq contient de nombreuses définitions et lemmes permettant de faire de l'analyse, mais ceux-ci se limitent à l'étude de fonctions totales à une seule variable. Même si l'étude des dérivées partielles de fonctions à deux variables peut le plus souvent réutiliser les lemmes existants, l'étude de la formule de d'Alembert comme solution de l'équation des ondes en dimension 1 montre les limites de cette approche. La formalisation en Coq de cette étude a ainsi nécessité de développer une théorie sur la dérivation des intégrales à paramètre et des fonctions à deux variables. De plus, démontrer en Coq la dérivabilité de la formule de d'Alembert s'est révélé être excessivement long. Nous avons donc construit une tactique par réflexion permettant de résumer l'expression « par application des théorèmes généraux de dérivation » souvent utilisée dans les démonstrations sur papier.

1. Introduction

Le système Coq est un assistant de preuve interactif s'appuyant sur la théorie des types et le calcul des constructions inductives [1]. Il a été conçu pour permettre la preuve de théorèmes mathématiques et de nombreux développements ont montré son utilité [7]. Nous nous sommes plus particulièrement intéressés à son utilisation pour vérifier des preuves venant du domaine de l'analyse. En effet, Coq est accompagné d'une bibliothèque standard qui contient entre autres une formalisation axiomatique des nombres réels et de nombreux théorèmes d'analyse [10]. Pour évaluer la richesse de cette bibliothèque, nous l'avons confronté à un problème relativement simple : vérifier que la formule de d'Alembert est une solution de l'équation des ondes en dimension 1. La formalisation Coq résultante et les outils associés sont disponibles à l'adresse

`http://www.lri.fr/~lelay/autoderive.html`

1.1. Équation des ondes et formule de d'Alembert

L'équation des ondes acoustiques dans un milieu homogène est l'une des modélisations les plus simples du phénomène physique de propagation d'onde. Son expression mathématique est un système d'équations aux dérivées partielles, c'est-à-dire des équations portant sur les dérivées par rapport aux coordonnées spatiales et temporelle d'une certaine valeur, ici l'amplitude d'une onde. La résolution de ce système a de nombreuses applications dans le domaine industriel, de l'acoustique d'une salle de concert aux tests non destructifs sur les matériaux.

*. Ce travail est financé par F∮ST (ANR-08-BLAN-0246-01) et Coquelicot (DIM Digiteo).

En dimension 1, le système d'équations s'obtient par l'étude de la position $(x, u(x,t))$ d'un point sur une corde finie ou infinie :

$$\forall (x,t) \in \mathbb{R}^2, \quad \begin{cases} \dfrac{\partial^2 u}{\partial t^2}(x,t) - c^2 \dfrac{\partial^2 u}{\partial x^2}(x,t) = f(x,t) \\ u(x,0) = u_0(x) \\ \dfrac{\partial u}{\partial t}(x,0) = u_1(x) \end{cases} \tag{1}$$

Ce système et sa résolution par un schéma numérique à trois points ont fait l'objet d'une formalisation en Coq [2]. Cette formalisation a cependant suivi une approche plutôt algébrique et n'a donc mis à contribution qu'une petite partie de la bibliothèque standard de Coq sur l'analyse. En particulier, elle est construite sur l'hypothèse que l'équation des ondes admet une solution suffisamment régulière.

Il s'avère que, en dimension 1, cette solution a une formule explicite. Il s'agit de la formule de d'Alembert [9, 13] :

$$u(x,t) = \underbrace{\frac{1}{2}\left(u_0(x+ct) + u_0(x-ct)\right)}_{\alpha(x,t)} + \underbrace{\frac{1}{2c}\int_{x-ct}^{x+ct} u_1(\xi)\, d\xi}_{\beta(x,t)} + \underbrace{\frac{1}{2c}\int_0^t \int_{x-c(t-\tau)}^{x+c(t-\tau)} f(\xi,\tau)\, d\xi\, d\tau}_{\gamma(x,t)} \tag{2}$$

Cette fonction va constituer notre cas d'étude, l'objectif étant de vérifier formellement qu'elle est solution du système (1). Le principe de la preuve est simple : il suffit de substituer u par son expression (2) dans les membres gauches de (1) et de vérifier que les expressions obtenues sont bien égales aux membres droits. Plus précisément, tout le travail va consister à définir et vérifier les dérivées secondes par rapport à x et t de chacun des trois morceaux α, β et γ de u.

Pour cela, nous avons besoin entre autres des théorèmes de dérivation pour les opérations de base (addition, multiplication, composition, ...), de la dérivation d'une primitive et d'une intégrale à paramètre. Notez que la présence de l'intégrale à paramètre dans γ rend la preuve assez technique à réaliser, même sur le papier.

1.2. Contributions

Notre travail a porté sur deux points. Tout d'abord, nous avons vérifié formellement que la formule de d'Alembert est solution. La section 2 présente les différentes notions permettant de vérifier la dérivabilité des fonctions α, β et γ, les théorèmes actuellement présents dans la bibliothèque standard de Coq, ainsi que ceux que nous avons démontrés afin de mener à bien la vérification. Parmi ces notions, il y a les dérivées partielles et les dérivées secondes (sections 2.1 et 2.2) et une variante de la primitive (section 2.3). La majorité des théorèmes d'analyse que nous avons dû prouver en Coq concerne les intégrales à paramètre et leur différentiabilité (section 2.4).

Il s'est avéré que, même en ajoutant tous les théorèmes qui manquaient à la bibliothèque standard, la vérification de la formule de d'Alembert n'en restait pas moins fastidieuse (section 3.1). Nous avons donc introduit un peu d'automatisation sous la forme d'une tactique par réflexion (section 3.2). La présence de termes de preuve dans l'expression de la dérivation et de l'intégration en Coq nous a conduit à généraliser les propriétés usuelles d'analyse aux fonctions partielles (section 3.3). Ce sont ces fonctions partielles que notre tactique utilise en interne pour prouver les équations fournies par l'utilisateur et générer les contraintes de définition (section 3.5).

La section 4 présente quelques travaux portant sur des sujets analogues et s'appuyant sur la bibliothèque standard de Coq. Finalement, la section 5 fait le point sur le travail effectué et détaille le gain apporté par notre tactique et ses limitations. Elle est aussi l'occasion de partager quelques enseignements de ce travail sur la bonne façon de formaliser l'analyse.

2. Vérification de la formule de d'Alembert

2.1. Exprimer l'équation des ondes

Pour exprimer le système d'équations (1) dans Coq, nous avons besoin de définir les notions de dérivées partielles d'une fonction à deux variables, ainsi que les notions de dérivées secondes correspondantes.

Avec les dérivées définies dans la bibliothèque standard de Coq, une propriété P sur le nombre dérivé de f en x s'exprime par :

```
forall pr : derivable_pt f x, P (derive_pt f x pr)
```

où `pr` est une preuve que f est dérivable en x et `derive_pt f x pr` est la valeur de la dérivée en ce point. Notez que la preuve de dérivabilité apparaît dans la valeur de la dérivée. Cette preuve `pr` (et la valeur de la dérivée) est obtenue en dépliant l'expression de la fonction f et de son nombre dérivé ℓ par application de lemmes démontrés dans la bibliothèque standard.

Il est parfois possible de se passer du terme de preuve `pr` pour exprimer l'énoncé ci-dessus en l'exprimant sous la forme :

```
forall l, derivable_pt_lim f x l -> P l
```

La proposition `derivable_pt_lim f x l` exprimant le fait que f est dérivable en x et admet ℓ pour dérivée en ce point.

Même s'il est possible d'exprimer l'équation des ondes à partir de ces définitions, l'écriture devient rapidement très lourde. Par exemple, un énoncé sur la dérivée seconde d'une fonction à deux variables par rapport à sa première variable aurait la forme :

```
forall (pr1 : derivable (fun x => f x y))
  (pr2 : derivable_pt (derive (fun x => f x y) pr1) x),
    P (derive_pt (derive (fun x => f x y) pr1) x pr2)
```

Cela revient à garder deux termes de preuve pour chaque dérivée seconde, et donc quatre termes de preuves pour exprimer la première condition de l'équation des ondes en Coq. Pour simplifier les notations, il est préférable d'avoir une définition spécifique pour les dérivées secondes :

Définition 2.1. Soient $f : \mathbb{R} \to \mathbb{R}$ et $x, \ell \in \mathbb{R}$.
On dit que f *est dérivable deux fois en* x *et admet pour dérivée seconde* ℓ si
- f est dérivable sur $\mathbb{R}$;
- la dérivée de f est dérivable en x et admet pour dérivée ℓ.

Remarque. Cette définition suppose que la fonction est dérivable sur tout $\mathbb{R}$. En fait, la dérivabilité sur un voisinage suffit mais elle n'est pas facile à manipuler avec les fonctions totales. Ce choix permet de pouvoir utiliser la bibliothèque standard avec les nouvelles définitions.

Ces définitions s'étendent au cas des dérivées partielles. Cela nous permet d'exprimer l'énoncé ci-dessus sous la forme :

```
forall (pr : derivable20_pt f x y), P (derive20_pt f x y pr)
```

Le suffixe "20" indique que l'on considère la dérivée seconde par rapport à la première variable ; le suffixe "01" désignerait la dérivée première par rapport à la seconde variable. Une variante permettant, comme pour la dérivée, de s'affranchir du terme de preuve est la suivante :

```
forall l, derivable20_pt_lim f x y l -> P l
```

Une fois ces notations en place, il devient facile d'exprimer le système d'équations différentielles.

2.2. Dériver Alpha

$$\forall (x,t) \in \mathbb{R}^2, \quad \alpha(x,t) = \frac{1}{2}\left(u_0(x+ct) + u_0(x-ct)\right)$$

On vérifie les différentes dérivées de α par application des théorèmes simples sur la dérivation des opérations de base ($+$, $\times$, ...) et de la composition. Ces théorèmes sont présents dans la bibliothèque standard.

La façon usuelle d'aborder les dérivées successives est de commencer par dériver une fois, puis de dériver l'expression obtenue, et ainsi de suite. La bibliothèque standard ne facilite pas ce travail cependant. Elle fournit un théorème prouvant que la dérivée ne dépend pas de la preuve de dérivabilité mais pas que la dérivée ne dépend pas de l'expression de la fonction. L'ajout du théorème suivant s'est donc révélé indispensable pour manipuler des dérivées multiples.

Théorème 2.2. *Soient $f, g : \mathbb{R} \to \mathbb{R}$ deux fonctions et $x, l \in \mathbb{R}$.*

$$\textit{Si } \forall y \in \mathbb{R},\ f(y) = g(y), \textit{ alors } f'(x) = l \Rightarrow g'(x) = l.$$

2.3. Dériver Beta

$$\forall (x,t) \in \mathbb{R}^2, \quad \beta(x,t) = \frac{1}{2c}\int_{x-ct}^{x+ct} u_1(\xi)\, d\xi$$

La relation de Chasles permet de réécrire β sous la forme

$$\frac{1}{2c}\left(\int_0^{x+ct} u_1(\xi)\, d\xi - \int_0^{x-ct} u_1(\xi)\, d\xi\right).$$

Les dérivées premières de β s'obtiennent alors à l'aide du théorème de dérivation d'une primitive. Les dérivées premières ne contiennent plus que des opérations de base et les dérivées secondes s'obtiennent donc par application des théorèmes généraux.

Comme pour la dérivée, l'expression d'une intégrale en Coq fait appelle à un terme de preuve, ainsi une propriété P sur l'intégrale de f entre a et b s'exprime en Coq par :

```
forall pr : Riemann_integrable f a b, P (RiemannInt pr).
```

La bibliothèque standard contient des lemmes permettant d'étudier et de dériver des intégrales de fonctions à une seule variable, mais leur utilisation n'est pas aisée. Par exemple, la primitive fournie par la bibliothèque standard de Coq n'est utilisable que sur un intervalle borné. Cette définition la rend particulièrement lourde à utiliser dans le cas où l'on s'intéresse à la primitive sur $\mathbb{R}$ entier. Nous avons donc prouvé un théorème de dérivation pour une primitive à la définition plus usuelle : $x \mapsto \int_a^x f(t)\, dt$. Notez que l'hypothèse est un peu plus forte que strictement nécessaire : nous avons supposé que la fonction f est continue sur $\mathbb{R}$.

Théorème 2.3. *Soient $f : \mathbb{R} \mapsto \mathbb{R}$ une fonction continue, $a \in \mathbb{R}$. Alors la fonction $F_a : x \mapsto \displaystyle\int_a^x f(t)\, dt$ est dérivable et*

$$\forall x \in \mathbb{R}, \quad F_a'(x) = f(x)$$

La définition précédente nécessite toujours d'employer la relation de Chasles pour séparer β en la somme de deux intégrales par rapport à un point fixé à l'avance. Pour simplifier les notations et l'étude de la fonction β, nous avons choisi d'introduire la définition suivante :

Définition 2.4. Soit f une fonction localement intégrable sur $\mathbb{R}$.[1] On définit la fonction à deux variables

$$\texttt{Rint}(f) : (a,b) \mapsto \int_a^b f(t)\,dt.$$

Un autre intérêt de cette définition apparaît lors de l'automatisation de la dérivabilité : elle permet de traiter les calculs de façon uniforme. En particulier, en utilisant le théorème 2.8 liant la différentielle et les dérivées partielles exposé dans la partie suivante, on obtient ce corollaire du théorème 2.3 :

Corollaire 2.5. *Soient f une fonction continue et $(a,b) \in \mathbb{R}^2$. La fonction $\texttt{Rint}(f)$ est différentiable en (a,b) de différentielle*

$$d\,\texttt{Rint}(f)(a,b) = (-f(a), f(b)).$$

Comme pour la dérivée de la primitive, l'hypothèse "f est continue" est très forte. Un ensemble plus faible d'hypothèses sera utilisé pour le théorème 3.8 de dérivation de la fonction intégrale dans le cadre de l'automatisation.

2.4. Dériver Gamma

$$\forall (x,t) \in \mathbb{R}^2, \quad \gamma(x,t) = \frac{1}{2c} \int_0^t \int_{x-c(t-\tau)}^{x+c(t-\tau)} f(\xi,\tau)\,d\xi\,d\tau$$

Les dérivées par rapport à x se calculent par le théorème de dérivation d'une intégrale à paramètre. Le traitement des dérivées par rapport à t est un peu plus compliqué puisque t apparaît à la fois comme borne de l'intégrale extérieure et à l'intérieur de celle-ci. Il faut donc passer par la fonction suivante et ses dérivées partielles.

$$(t_1,t_2) \mapsto \frac{1}{2c} \int_0^{t_1} \int_{x-c(t_2-\tau)}^{x+c(t_2-\tau)} f(\xi,\tau)\,d\xi\,d\tau.$$

La bibliothèque standard de Coq n'est d'aucune utilité pour traiter la fonction γ. Elle ne s'intéresse ni aux intégrales à paramètre ni aux fonctions à plusieurs variables et leur différentiabilité. Nous avons donc démontré en Coq les théorèmes suivants.

Dérivation d'une intégrale à paramètre

Théorème 2.6. *Soient $a,b \in \mathbb{R}$, $a < b$ et $f : (x,t) \mapsto f(x,t)$ une fonction telle que :*
- *f est intégrable par rapport à t sur l'intervalle $[a;b]$;*
- *f est dérivable par rapport à x ;*
- *$\dfrac{\partial f}{\partial x}$ est continue sur $\mathbb{R} \times [a;b]$.*

Alors, la fonction $F : x \mapsto \displaystyle\int_a^b f(x,t)\,dt$ est dérivable de dérivée $F'(x) = \displaystyle\int_a^b \frac{\partial f}{\partial x}(x,t)\,dt$.

Ce théorème s'appuie sur le fait qu'une fonction de $\mathbb{R}^2 \to \mathbb{R}$ continue est uniformément continue sur un compact. Ce fait a été admis pour l'instant ; c'est la seule hypothèse de ce travail.

1. Une fonction f est dite localement intégrable sur S si, pour tout $a,b \in S$, f est Riemann-intégrable entre a et b.

Différentiabilité

Définition 2.7. Soient $f : \mathbb{R}^2 \to \mathbb{R}$ une fonction, $x, l \in \mathbb{R}^2$. On dit que f *est différentiable au point* x *de différentielle* l si

$$\forall \varepsilon \in \mathbb{R}_+^*,\ \exists \delta \in \mathbb{R}_+^*, \quad \forall k \in \mathbb{R}^2,\ \|k\|_\infty < \delta \Rightarrow |f(x+k) - f(x) - \langle l; k\rangle| \leq \varepsilon \|k\|_\infty.$$

Les dérivées partielles s'expriment facilement comme projections de la différentielle. Le théorème suivant permet de faire le trajet en sens inverse, ce qui est moins trivial.

Théorème 2.8. *Soient* $f : \mathbb{R}^2 \to \mathbb{R}$ *une fonction et* $(x, y) \in \mathbb{R}^2$.
Si f *admet des dérivées partielles* f_1' *et* f_2' *continues au point* $(x, y) \in \mathbb{R}^2$*, alors* f *est différentiable en* (x, y) *et admet pour différentielle* $(f_1'(x, y), f_2'(x, y))$.

L'hypothèse sur la continuité des dérivées partielles dans ce dernier théorème provient de l'utilisation du théorème de la valeur moyenne lors de la démonstration. Celui-ci nous fournit une différence entre deux valeurs des dérivées partielles qui doit être suffisamment petite.

Dérivée d'une fonction composée

Pour finir, le théorème important de cette partie permet la dérivation de γ par rapport à t.

Théorème 2.9. *Soient* $f : \mathbb{R}^2 \to \mathbb{R}$ *et* $u_1, u_2 : \mathbb{R} \to \mathbb{R}$ *des fonctions. Soit* $x \in \mathbb{R}$.
Si
- u_1 *et* u_2 *sont dérivables en* x,
- f *est différentiable en* $(u_1(x), u_2(x))$,

alors la fonction $\Phi : x \mapsto f(u_1(x), u_2(x))$ *est dérivable en* x *et*

$$\Phi'(x) = u_1'(x) \cdot f_1'(u_1(x), u_2(x)) + u_2'(x) \cdot f_2'(u_1(x), u_2(x))$$

Ce théorème se démontre facilement en Coq en le regardant comme un corollaire du lemme suivant.

Lemme 2.10. *Soient* $f, u_1, u_2 : \mathbb{R}^2 \to \mathbb{R}$ *des fonctions. Soit* $x \in \mathbb{R}$.
Si
- u_1 *et* u_2 *sont différentiables en* x,
- f *est différentiable en* $(u_1(x), u_2(x))$,

alors la fonction $\Phi : x \mapsto f(u_1(x), u_2(x))$ *est dérivable en* x *et*

$$\begin{aligned}\Phi'(x) = (\quad & \pi_1\left(u_1'(x)\right) \cdot f_1'(u_1(x), u_2(x)) + \pi_1\left(u_2'(x)\right) \cdot f_2'(u_1(x), u_2(x)) \quad ; \\ & \pi_2\left(u_1'(x)\right) \cdot f_1'(u_1(x), u_2(x)) + \pi_2\left(u_2'(x)\right) \cdot f_2'(u_1(x), u_2(x)) \quad)\end{aligned}$$

où π_1 *et* π_2 *sont les première et deuxième projections.*

La démonstration de ce lemme est assez complexe du fait du nombre important de paramètres à extraire des différentes hypothèses de continuité et différentiabilité.

Les différents théorèmes présentés jusqu'ici nous ont permis de démontrer formellement que la formule de d'Alembert est une solution de l'équation des ondes en dimension 1.

3. Automatisation de la preuve

3.1. Un exemple détaillé

Les démonstrations que nous avons présentées pour α, β et γ sont essentiellement constituées par l'application de théorèmes de dérivation dont les seules hypothèses sont la dérivabilité d'autres

Code 1 Un script Coq que l'utilisateur n'aurait pas dû avoir à écrire : le calcul de la fonction β_2'.

```
Lemma beta01_lim : forall x t,
  derivable01_pt_lim beta x t (1/2 * (u1 (x+c*t) + u1 (x-c*t))).
Proof.
  intros.
  (* étape 1 : faire apparaître tous les termes ignorés dans l'expression de la dérivée *)
  assert (1/2 * (u1 (x+c*t) + u1 (x-c*t)) =
    1/(2*c) * (fst (- u1 (x-c*t),u1 (x+c*t)) * (0-c*1)
    + snd (- u1 (x-c*t),u1 (x+c*t)) * (0+c*1))).
    simpl ; field.
    apply Hc.
    rewrite H ; clear H.
  (* étape 2 : prouver un lemme intermédiaire *)
  assert (Cf : continuity u1).
    apply derivable_continuous.
    apply d1_u1.
  (* étape 3 : appliquer mécaniquement les théorèmes de dérivation *)
  apply derivable_pt_lim_scal.
  apply (derivable_pt_lim_comp2 (Rint u1 RInt_u1)).
  apply differentiable_pt_lim_Rint, Cf.
  apply derivable_pt_lim_minus.
  apply derivable_pt_lim_const.
  apply (derivable_pt_lim_expr (fun t => c*t)).
    intros ; reflexivity.
  apply derivable_pt_lim_scal.
  apply derivable_pt_lim_id.
  apply derivable_pt_lim_plus.
  apply derivable_pt_lim_const.
  apply (derivable_pt_lim_expr (fun t => c*t)).
    intros ; reflexivity.
  apply derivable_pt_lim_scal.
  apply derivable_pt_lim_id.
Qed.
```

fonctions. Le code Coq 1 donne l'exemple de la démonstration que β est dérivable par rapport à t et admet pour dérivée $\beta_2'(x,t) = \frac{1}{2}\left(u_1(x+ct) - u_1(x-ct)\right)$.

Dans une démonstration sur papier, on dirait que "β peut s'écrire $\frac{1}{2c}(\int_0^{x+ct} u_1 - \int_0^{x-ct} u_1)$ et u_1 est continue, donc par application des théorèmes généraux, β est dérivable par rapport à t de dérivée $\frac{1}{2}\left(u_1(x+ct) - u_1(x-ct)\right)$". La notion de fonction intégrale (corollaire 2.5) permet de se passer de l'étape de réécriture qui est assez lourde en Coq. Mais d'autres étapes importantes de cette démonstration restent à la charge de l'utilisateur :

1. Réécrire la dérivée pour faire apparaître des opérations comme "$+0$" ou "$\times 1$" que l'on ignore généralement mais qui sont indispensables à l'application des théorèmes de dérivabilité ;
2. Justifier que $(x,y) \mapsto \int_x^y u_1(\xi)\,d\xi$ est différentiable en montrant que u_1 est continue ;
3. Appliquer les théorèmes de dérivation, en précisant éventuellement leurs paramètres quand Coq ne les infère pas.

L'objectif est donc de construire un outil qui se charge d'étudier la dérivabilité à partir d'un but de la forme `derivable_pt_lim f x g` en demandant (éventuellement) à l'utilisateur de démontrer des faits (par exemple la continuité de u_1) qu'il n'aurait pas réussi à inférer à partir du contexte.

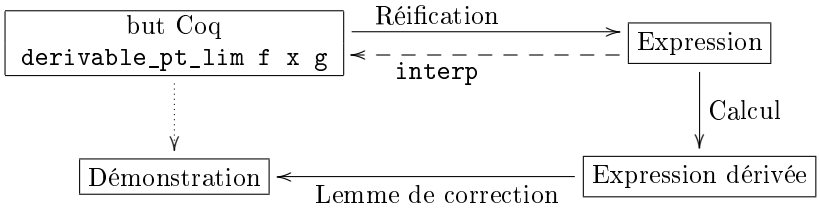
3.2. Preuve par réflexion

Cet outil prend la forme d'une tactique qui analyse le but et génère le terme de preuve correspondant, potentiellement en appelant d'autres tactiques pour certains sous-termes. Un tel outil

fonctionne généralement dans la méta-logique par application successive de théorèmes et de règles de réécriture en fonction de la forme du but. Coq a un langage dédié appelé Ltac pour exprimer de tels algorithmes [4]. Cependant, une autre approche se rencontre en Coq. Elle consiste à encoder le but dans le logique et à calculer sur cette forme. Il n'y a alors qu'un seul théorème à appliquer, celui qui prouve la correction de ce calcul. Il s'agit de la preuve par réflexion [3].

Dans notre cas, le calcul est celui de la dérivée et sa comparaison avec celle de l'utilisateur. Le calcul d'une dérivée ne demandant pas un grand nombre d'opérations, la réflexion ne va pas apporter ici de gain notable d'efficacité. Il n'y aura pas non plus de réduction importante sur la taille du terme de preuve généré. Il y a néanmoins un avantage à l'approche par réflexion : calculer une dérivée est une opération simple, reconnaissable au premier coup d'œil et que l'on retrouve dans nombre de systèmes de calcul formel. Il est donc souhaitable d'utiliser cette représentation traditionnelle de la dérivée.

Le principe de réflexion est résumé par la figure suivante :



Les composants nécessaires pour effectuer une telle preuve par réflexion sont :
- un type inductif pour représenter les fonctions à étudier (définition 3.1),
- une dérivée formelle travaillant sur les expressions représentées par ce type (définition 3.2),
- un preuve que la dérivée ainsi construite est compatible avec la dérivée traditionnelle des fonctions (section 3.5),
- un outil de réification capable de produire une expression inductive représentant une fonction donnée.

Définition 3.1. Une expression e reconnue par la tactique est de la forme

$$e ::= x \mid cst \mid e + e \mid e \times e \mid -e \mid \int_e^e f(t)\,dt \mid f(e) \mid f(e,e)$$

où x est la variable de dérivation, cst est une expression réelle ne contenant pas x (c'est-à-dire une constante du point de vue de la dérivation) et f est une "fonction" à un ou deux arguments dont la définition sera explicitée plus bas.

Définition 3.2. La dérivée formelle d'une expression est définie de manière classique par :

$$\begin{aligned}
D(x) &= 1 \\
D(cst) &= 0 \\
D(e_1 + e_2) &= D(e_1) + D(e_2) \\
D(e_1 \times e_2) &= e_1 \times D(e_2) + e_2 \times D(e_1) \\
D(-e) &= -D(e) \\
D\left(\int_{e_1}^{e_2} f(t)\,dt\right) &= f(e_2) + (-f(e_2)) \\
D(f(e)) &= D(e) \times f'(e) \\
D(f(e_1, e_2)) &= D(e_1) \times f'_1(e_1, e_2) + D(e_2) \times f'_2(e_1, e_2)
\end{aligned}$$

La version implantée en Coq effectue des tests supplémentaires pour optimiser quelques termes. Par exemple, elle détecte si l'un des membres d'un produit est une constante et elle évite dans ce cas de produire des multiplications par 0.

La question de la forme à donner aux "fonctions" dans les expressions se pose au moment d'exprimer les dérivées de fonctions comme $D(f(e)) = D(e) \times f'(e)$. En effet, l'expression d'une dérivée se fait en Coq à l'aide d'un terme de preuve que l'on ne souhaite pas devoir fournir avant ou pendant le fonctionnement de la tactique.

Nous avons donc décidé d'utiliser des fonctions partielles pour représenter ces termes. La fonction dérivée de f devient ainsi la fonction définie sur l'ensemble où f est dérivable et qui a pour valeur la fonction dérivée de f, la justification de la dérivabilité de f étant laissée à l'utilisateur.

3.3. Fonctions partielles

Les fonctions partielles s'appuient sur le type `partial_val` qui représente des valeurs partielles, c'est-à-dire des réels auxquels il faut fournir un terme de preuve avant de pouvoir utiliser leur valeur. La définition formelle est la suivante.

Définition 3.3. Une valeur partielle est la donnée de :
- un domaine `pdom : Type` ;
- une application `pval : pdom -> R` qui à une preuve dans le domaine associe la valeur ;
- une preuve `pHeq : forall d1 d2 : pdom, pval d1 = pval d2` qui dit que la valeur réelle ne dépend pas du terme de preuve.

Les fonctions partielles à une ou deux variables sont alors de type (`R ->`) `R -> partial_val`. Pour une fonction partielle f, on notera par la suite $x \in f_{dom}$ pour "`d : pdom (f x)`" et $f_{val}(x)$ pour "`pval (f x) d`" afin de rendre les théorèmes plus lisibles.

Notez que `Type` a été choisi comme type des domaines `pdom` au lieu de `Prop` qui semblerait plus naturel. En effet, on veut pouvoir extraire certaines informations comme la dérivabilité (de type `Set`) et la Riemann-intégrabilité (de type `Type`) du domaine afin de construire la valeur. Par exemple, la valeur partielle représentant l'intégrale $\int_a^b f$ demande pour domaine une preuve que f est intégrable entre a et b et donne comme valeur l'intégrale extraite de cette preuve. [2]

Les notions de continuité et de dérivabilité ont été adaptées au cas des fonctions partielles.

Définition 3.4. Soient f une fonction partielle et $x \in f_{dom}$.
f est continue en x si

$$\forall \varepsilon \in \mathbb{R}_+^*,\ \exists \delta \in \mathbb{R}_+^*, \quad \forall y \in \mathbb{R},\ |y - x| < \delta \Rightarrow y \in f_{dom} \wedge |f_{val}(y) - f_{val}(x)| < \varepsilon$$

Définition 3.5. Soient f une fonction partielle, $x \in f_{dom}$ et $l \in \mathbb{R}$.
f est dérivable en x et admet pour dérivée l si

$$\begin{gathered}\forall \varepsilon \in \mathbb{R}_+^*,\ \exists \delta \in \mathbb{R}_+^*, \quad \forall y \in \mathbb{R},\ |y - x| < \delta \\ \Rightarrow y \in f_{dom} \wedge |f_{val}(y) - f_{val}(x) - l \times (y - x)| \leq \varepsilon \cdot |y - x|\end{gathered}$$

Ces définitions s'étendent au cas de fonctions à deux variables. Pour la différentiabilité, le produit $l \times (y - x)$ est alors remplacé par un produit scalaire.

Les valeurs partielles ressemblent à un type monadique, l'opérateur *return* (`Pval_val`) associant à un réel un domaine trivial (`True`) et l'opérateur *bind* (`Pfun_app`) appliquant une valeur partielle à une fonction partielle. Notez que ce n'est pas tout à fait une monade puisque les axiomes de monade ne sont vérifiés qu'à isomorphisme des types de domaine près. Il n'en reste pas moins facile de définir les opérations d'addition, multiplication, composition, etc, de fonctions partielles.

2. Une autre approche serait de définir une variante de l'intégrabilité qui vive dans `Prop` et une variante de l'intégrale qui n'ait pas besoin d'extraire de l'information de la preuve d'intégrabilité et de prouver l'équivalence avec les définitions de la bibliothèque standard. Bien que plus satisfaisante, cette approche n'aurait apporté aucune amélioration à la tactique et elle n'a donc pas été explorée plus avant au vu de son coût de formalisation.

Nous avons aussi démontré la compatibilité de la continuité et de la dérivabilité avec ces opérations partielles. La bibliothèque standard n'a été que de peu d'utilité dans ce travail puisque les domaines des valeurs partielles se cachent au plus profond des définitions de continuité et de dérivabilité et qu'il faut donc en revenir à manipuler des ε et des δ.

3.4. Intégrale de fonctions partielles

Même si l'intégrale de Riemann définie dans la bibliothèque standard de Coq n'utilise les valeurs de la fonction intégrée f que sur l'intervalle d'intégration $[a, b]$, cela ne transparaît pas dans son type. Pour pouvoir l'appliquer à une fonction partielle, il nous faut donc d'abord construire une fonction définie sur $\mathbb{R}$ entier.

Définition 3.6. Soient f une fonction partielle et $[a, b]$ [3] un intervalle inclus dans le domaine de f.

$$\forall x \in \mathbb{R},\ eval_{a,b}(f)(x) = \begin{cases} f_{val}(x) & \text{si } x \in [a; b], \\ f_{val}(a) & \text{si } x < a, \\ f_{val}(b) & \text{si } x > b. \end{cases}$$

Cette fonction conserve les propriétés de continuité et de dérivabilité sur l'intervalle $]a; b[$. De plus, si f est continue sur $[a, b]$, alors $eval_{a,b}(f)$ est continue sur $\mathbb{R}$.

Définition 3.7. Soient f une fonction partielle et $a, b \in \mathbb{R}$. On dit que f est Riemann-intégrable entre a et b si f est définie entre a et b et si la fonction $eval_{a,b}(f)$ est Riemann-intégrable entre a et b. L'intégrale de f entre a et b est alors définie comme l'intégrale de Riemann de $eval_{a,b}(f)$ entre a et b.

Comme pour les fonctions totales, il nous reste à définir une généralisation de la primitive permettant de dériver sans avoir à supposer que la borne inférieure est constante.

Théorème 3.8. *Soient f une fonction partielle et $(a, b) \in \mathbb{R}^2$.*
Si :
- *f est intégrable entre a et b,*
- *f est continue en a et en b,*
- *il existe $d \in \mathbb{R}_+^*$ tel que :*
  - *f est intégrable entre $a - d$ et $a + d$,*
  - *f est intégrable entre $b - d$ et $b + d$,*

alors la fonction partielle $\texttt{Pfun_Rint}(f) : (x, y) \mapsto \int_x^y f(t)\,dt$ *est differentiable au point (a, b) et admet pour différentielle* $d\texttt{Pfun_Rint}(f)(a, b) = (-f(a); f(b))$.

Remarque. Pour l'intégrale de Lebesgue, les deux dernières hypothèses sont des conséquences de l'hypothèse "f est continue en a et en b" car les fonctions majorées sur un compact sont intégrables sur celui-ci. Dans le cas de l'intégrale de Riemann qui est utilisée ici, cette hypothèse est indispensable pour dire que la fonction intégrale est définie sur un voisinage de (x, y).

3.5. Lemme de correction et tactique

Tout ce travail sur les fonctions partielles vise à pouvoir définir une fonction `interp` permettant de transformer une expression inductive en une fonction à valeur réelle. La fonction `interp` nous permet alors d'énoncer le lemme de correction de la dérivée formelle `D` définie plus haut.

```
Lemma D_correct : forall (e : expr) (x : R),
  let vd := interp (D e) x in
  forall d : pdom vd, Pfun_derivable_pt_lim (interp e) x (pval vd d).
```

3. Plus précisément, les bornes a et b ne sont pas ordonnées et l'intervalle est donc $[\min(a, b); \max(a, b)]$.

Ce lemme ne s'intéressant qu'aux fonctions partielles, c'est en fait un corollaire qui est utilisé dans la tactique.

```
Lemma AutoDerive_helper : forall e f x l,
  (forall x, let v := interp e x in forall d : pdom v, pval v d = f x) ->
  (let vd := interp (D e) x in { d : pdom vd | pval vd d = l }) ->
  derivable_pt_lim f x l.
```

Autrement dit, la tactique `AutoDerive` transforme un but de la forme `derivable_pt_lim f x l` en deux sous-buts. Le premier demande de vérifier que l'interprétation de l'expression e prend bien les mêmes valeurs que la fonction f sur le domaine de définition de e. Le second demande de vérifier que l'interprétation de $D(e)$ a un domaine non vide et que sa valeur est égale à ℓ.

Dans une approche par réflexion classique, le premier sous-but serait traité immédiatement par β-conversion, à supposer que la tactique `reify`[4] qui produit l'expression e à partir de la fonction f n'a pas fait d'erreur. Dans notre cas, ce n'est pas tout à fait vrai, à cause des termes de preuve présents dans les intégrales. En effet, les termes de preuve générés par `interp` ne sont pas ceux de l'utilisateur et il y a donc potentiellement une phase de réécriture pour unifier les deux fonctions. Quoi qu'il en soit, cette étape est complètement transparente et l'utilisateur ne verra jamais ce sous-but.

L'autre sous-but est plus intéressant. Si la dérivée formelle a produit une fonction u' dérivée d'une fonction u quelconque, alors montrer que le domaine de $D(e)$ est non vide demande de prouver que u est effectivement dérivable au point où u' est appliqué. Ce sont ces preuves que l'utilisateur aura à fournir. La tactique cherche cependant à en éliminer le plus possible en s'aidant du contexte[5]. L'autre partie de ce sous-but demande de prouver l'égalité entre la valeur de la dérivée et la dérivée utilisateur. Ce but est laissé à l'utilisateur, la tactique essayant seulement d'appliquer les tactiques `ring` et `field` au cas où les deux expressions seraient facilement identifiables, ce qui a toujours été le cas pour la vérification de la formule de d'Alembert.

Considérons à nouveau le code Coq 1. À l'aide de la tactique `AutoDerive`, le script se réduit à :

```
Lemma beta01_lim : forall x t, derivable01_pt_lim beta x t
  (1/2 * (u1 (x+c*t) + u1 (x-c*t))).
Proof.
  intros ; unfold beta.
  AutoDerive.
  apply Hc. (* c ≠ 0 *)
Qed.
```

La tactique a automatiquement déchargé les conditions d'existence de β_2' et a utilisé les lois de corps de $\mathbb{R}$ pour prouver que la dérivée proposée par l'utilisateur correspond bien à la dérivée calculée. Cette égalité nécessite cependant que c soit non nul (simplification d'un quotient) et c'est ce sous-but qui reste à la charge de l'utilisateur. Il représente une propriété à vérifier par les paramètres définis à l'extérieur du lemme ; il est donc normal de devoir le démontrer pour achever la preuve, ce qui se fait en appliquant l'hypothèse `Hc` qui dit que $c \neq 0$.

4. Variations autour de la dérivation

4.1. Autres bibliothèques et prouveurs

De nombreux autres assistants de preuve ont une formalisation de l'analyse qui aurait permis d'attaquer la démonstration de la formule de d'Alembert. Chacun a ses propres définitions de la dérivation et de l'intégration qui s'appuient généralement sur les particularités du système logique

4. L'étape de réification se fait nécessairement en Ltac.
5. Les domaines n'utilisant pas une représentation inductive, il y a ici aussi une étape de réification en Ltac.

pour en simplifier l'usage. La liste suivante est loin d'être exhaustive. Isabelle/HOL construit sa bibliothèque des réels sur l'analyse non-standard, ce qui simplifie les preuves de dérivation puisqu'il n'est plus nécessaire de manipuler ni ε ni δ [6]. La bibliothèque de PVS utilise des sous-types à profusion ; l'utilisateur n'a donc plus à fournir des termes de preuve au milieu des formules, le système l'interrogeant le cas échéant [5]. La bibliothèque de HOL (Light) utilise des définitions proches de celles que l'on trouve dans Coq, mais utilise une définition de la dérivabilité qui se ramène à la continuité du taux d'accroissement, ce qui simplifie certaines preuves [8].

Les paragraphes suivants rentrent plus en détail sur quelques développements s'appuyant sur les réels axiomatiques de Coq.

4.2. Bibliothèque standard de Coq

La bibliothèque standard contient en réalité deux définitions de la dérivabilité. La première définition, basée sur `derivable_pt_lim`, a été présentée et utilisée tout au long de cet article. La deuxième définition correspond au prédicat `D_in f f' dom x` qui affirme que la dérivée de f en x vaut $f'(x)$ [6], et cela en ne considérant que les points présents dans l'ensemble *dom*. Sur son principe, ce prédicat est donc très proche de notre définition 3.5 de la dérivée pour une fonction partielle. La différence vient du fait que, même si `D_in` ne s'intéresse qu'aux valeurs de f sur le domaine *dom*, la fonction f n'en reste pas moins totale. Il était donc impossible de l'utiliser au cœur de la tactique.

Il y a un autre point sur lequel la bibliothèque standard présente un air de ressemblance avec ce travail. Elle fournit en effet une tactique `reg` permettant de prouver automatiquement la continuité, la dérivabilité et le calcul des dérivées. Pour ce dernier point, la tactique fonctionne par réécritures successives : elle cherche un terme de la forme `derive_pt f x pr`, applique le lemme qui correspond à l'opérateur en tête de f, puis s'appelle récursivement sur les termes `derive_pt` créés par le lemme. On peut néanmoins relever plusieurs grandes différences avec le travail présenté dans cet article.

Tout d'abord, la tactique travaille sur des termes `derive_pt` (ce qui est indispensable pour une approche par réécriture) et l'utilisateur doit donc avoir sous la main la preuve *pr* de dérivabilité pour mettre le but sous la bonne forme. Ce n'est pas très limitant, d'autant que `reg` est capable de la fabriquer, mais cela revient donc à faire deux fois d'affilée la même preuve, ce que nous souhaitions éviter. Dans le même ordre d'idée, nous voulions construire une tactique réflexive qui colle au plus près de ce que l'on voudrait écrire pour un logiciel de calcul formel et la tactique `reg` ne satisfait pas ce critère.

Un avantage de la tactique `reg` est son exhaustivité : elle gère toutes les fonctions pour lesquelles la bibliothèque standard connaît des propriétés de dérivabilité, par exemple les fonctions trigonométriques. Malheureusement, elle ne traite pas la dérivation de primitives et d'intégrales et ne convient donc pas pour ce travail. Remarquez que, même pour les propriétés de continuité, la tactique `reg` n'est ici d'aucune utilité. En effet, la plupart de celles dont nous avons besoin portent sur $\mathbb{R}^2$ (théorèmes 2.6 et 2.8), ce qui est hors de portée de `reg`.

4.3. Différentiation automatique

En dehors de la bibliothèque standard, la preuve de correction de l'algorithme de différentiation automatique O∂yssée [11] présente des points communs avec notre travail. En particulier, l'opérateur de différentiation est une fonction inductive similaire à la nôtre. Les expressions considérées sont par contre purement polynomiales et il n'y a donc pas besoin de fonctions partielles. L'objectif de ce travail n'était pas non plus de fournir une tactique et il s'arrête donc à la fonction de correction.

Dans la catégorie différentiation automatique, il faut aussi mentionner la tactique `interval` qui

6. Bien que $f'(x)$ soit la seule valeur de f' à être utilisée, la définition nécessite que la dérivée soit passée comme une fonction, ce qui est un choix étrange.

permet la preuve d'inégalité sur des expressions à valeurs réelles [12]. Les expressions traitées, même si elles ne contiennent toujours pas d'intégrales, peuvent cette fois présenter des points de discontinuité ou de non-dérivabilité. Mais l'intervalle d'entrée ayant des bornes numériques explicites, l'algorithme de différentiation sait s'il y a des points à problème et n'a donc pas besoin de fonctions partielles.

5. Conclusion

5.1. La formule de d'Alembert

Après avoir défini et démontré 50 définitions et lemmes complémentaires pour l'arithmétique réelle, 50 pour les fonctions à une variables, et 135 pour les fonctions à deux variables, nous sommes parvenus à vérifier que la formule de d'Alembert est bien une solution de l'équation des ondes. En l'absence d'automatisation, cette formalisation Coq représente environ 2500 lignes de preuve et le seul détail qui n'a pas été prouvé formellement est la compacité sur $\mathbb{R}^2$ dans le théorème 2.6.

Même si la majorité des théorèmes permettant d'étudier les dérivées d'une fonction à une variable sont maintenant définis et démontrés, certains, comme le théorème 2.8, restent difficiles à utiliser du fait de leur énoncé compliqué en Coq. En particulier, l'usage des opérateurs `derive` utilisant des termes de preuve contraint l'utilisateur à réécrire le but afin d'appliquer certains théorèmes. Ce type d'énoncé est donc à éviter, quitte à introduire une fonction et à ajouter dans les hypothèses que cette fonction est bien la dérivée souhaitée.

5.2. La tactique

Ce travail a fait ressortir plusieurs situations où les preuves étaient répétitives et ne justifiaient pas une interaction avec l'utilisateur. C'est pourquoi nous avons écrit la tactique `AutoDerive` afin de décharger automatiquement les démonstrations de dérivabilité. Elle est capable de vérifier les dérivées de fonctions construites à partir de polynômes, d'intégrales de fonctions à une variable et de fonctions données par l'utilisateur. Dans le cas des polynômes, elle effectue le travail en totalité et l'utilisateur n'a besoin de prouver aucune autre condition. Dans les autres cas, elle peut produire des sous-buts demandant à l'utilisateur de prouver des propriétés supplémentaires. La définition des fonctions partielles ainsi que les lemmes pour les utiliser ont demandé environ 175 définitions et lemmes et 3000 lignes de preuve.

Concernant la vérification de la formule de d'Alembert, cette tactique permet de traiter entièrement les fonctions α et β. Pour l'étude de ces deux fonctions, cela fait passer les preuves d'environ 120 lignes à une vingtaine. Cela évite surtout toutes les réécritures décrites dans la section 3.1 et permet de faire ces preuves en quelques minutes seulement. Pour la fonction γ, la dérivée des intégrales à paramètre n'étant pas traitées par la tactique, seules les fonctions annexes sont aidées (c'est-à-dire la fonction $(x,t,\tau) \mapsto \int_{x-ct}^{x+ct} f(\xi,\tau)\,d\xi$ et ses dérivées). Le travail se fait donc en trois temps : l'utilisateur dérive d'abord sous l'intégrale, puis il dérive autour de l'intégrale en considérant l'intérieur opaque, et enfin il recolle les morceaux en réécrivant l'expression. La réduction des preuves pour les fonctions annexes de γ sont du même ordre que les réductions pour α et β. Compte tenu du nombre important de conditions nécessaires à l'application des théorèmes 2.8 et 2.6, la preuve de γ prend au final une à deux heures.

Il reste encore à compléter cette tactique avec les différentes fonctions et opérations non abordées pour le moment (fonctions trigonométriques, quotient, ...). Elles ne présentent pas de difficultés particulières : elles étaient juste inutiles pour démontrer la formule de d'Alembert. Il faudrait aussi une variante de la tactique capable de générer une hypothèse de la forme `derivable_pt_lim` pour une fonction donnée afin de simplifier un peu le calcul des dérivées multiples.

La plus grosse limitation de la tactique réside dans le fait qu'elle ne travaille pas sous le symbole

intégrale, ce qui conduit aux complications décrites ci-dessus pour la fonction γ. La difficulté n'est pas tant de traiter les lieurs mais plutôt le fait que dériver l'expression sous le symbole intégrale conduit à une expression partielle dont il faut ensuite assurer la définition et la continuité (voire juste l'intégrabilité). Ceci est un point que nous comptons approfondir dans des travaux futurs.

5.3. Formaliser l'analyse

Ce travail nous a appris de nombreuses choses sur la bonne façon de manipuler en Coq les notions de base de l'analyse. La plupart peuvent être considérées comme des détails : utilisation des termes de preuve, définition de la dérivabilité, définition de la primitive, etc. On pourrait même assimiler ça à des questions de goût et de couleur et supposer que d'autres utilisateurs auraient réagi autrement.

L'un des grands enseignements de ce travail réside donc plutôt dans la manipulation et la dérivation des fonctions à plusieurs variables. La bibliothèque standard a fait le choix de ne considérer que les fonctions de $\mathbb{R} \to \mathbb{R}$; une autre bibliothèque aurait pu s'intéresser au cas général des fonctions de $\mathbb{R}^n \to \mathbb{R}^m$. Il s'avère en fait que ces deux extrêmes sont des conséquences immédiates du cas $\mathbb{R}^2 \to \mathbb{R}$.

Tout d'abord, voyons les défauts d'une formalisation de $\mathbb{R} \to \mathbb{R}$. Le premier est évident : les théorèmes ne sont que de peu d'utilité pour passer à la dimension supérieure et l'utilisateur se retrouve très rapidement à devoir manipuler des ε et des δ. Le deuxième défaut est plus insidieux : les preuves des théorèmes sont bien plus compliquées que nécessaires. Par exemple, la bibliothèque standard propose un théorème pour $(u_1(x) \times u_2(x))' = u_1'(x) \times u_2(x) + u_1(x) \times u_2'(x)$ à la preuve relativement compliquée. Or il s'agit d'une conséquence immédiate du lemme 2.10 et de la différentiabilité de $x_1 \times x_2$. Évidemment, la complexité de la preuve originale se retrouve alors cachée dans celle du lemme 2.10. Mais cette complexité n'a plus besoin d'être répétée pour chacun des opérateurs binaires : la preuve est faite une fois pour toutes pour traiter la composition.

Voyons maintenant le cas $\mathbb{R}^n \to \mathbb{R}^m$. La valeur de m n'a pas vraiment d'importance : c'est juste l'affaire de projections et $m = 1$ suffit. Pour ce qui est de n, on pourrait croire que le passage de 2 à n est aussi compliqué que celui de 1 à 2 et que l'utilisateur n'est donc pas plus avancé. La preuve de γ dans la section 2.4 esquisse une solution. L'intégrale double peut en effet y être vue comme une fonction à trois variables quand on cherche à calculer les dérivées par rapport à t. Voyons ici le cas général où l'on cherche à dériver l'expression suivante par rapport à x :

$$f(x) = g(u_1(x), u_2(x), \ldots, u_n(x)).$$

Il semblerait naturel d'utiliser une notion de différentielle de fonctions à n variables pour traiter g. En fait, il suffit d'étudier la dérivabilité de la fonction à deux variables suivante :

$$F(x_1, x_2) = g(u_1(x_1), u_2(x_2), \ldots, u_n(x_2)).$$

Le théorème 2.8 nous permet alors de passer aux dérivées partielles et ainsi de se ramener au cas d'une fonction à $n-1$ arguments pour traiter la dérivée par rapport à x_2. Le cas $\mathbb{R}^n \to \mathbb{R}$ n'est donc qu'une induction simple sur le cas $\mathbb{R}^2 \to \mathbb{R}$. Cela n'empêche pas d'écrire les théorèmes correspondants pour simplifier la vie de l'utilisateur, mais cette approche permet au moins d'en simplifier les preuves.

Références

[1] Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development. Coq'Art : The Calculus of Inductive Constructions.* Texts in Theoretical Computer Science. Springer Verlag, 2004.

[2] Sylvie Boldo, François Clément, Jean-Christophe Filliâtre, Micaela Mayero, Guillaume Melquiond, and Pierre Weis. Formal proof of a wave equation resolution scheme : the method

error. In Matt Kaufmann and Lawrence C. Paulson, editors, *Proceedings of the 1st International Conference on Interactive Theorem Proving*, volume 6172 of *Lecture Notes in Computer Science*, pages 147–162, Edimbourg, Royaume-Uni, 2010.

[3] Samuel Boutin. Using reflection to build efficient and certified decision procedures. In Martín Abadi and Takayasu Ito, editors, *Proceedings of the 3th International Symposium on Theoretical Aspects of Computer Software TACS'97*, volume 1281 of *Lecture Notes in Computer Science*, pages 515–529, Sendai, Japon, 1997.

[4] David Delahaye. A tactic language for the system Coq. In Michel Parigot and Andrei Voronkov, editors, *Proceedings of the 7th International Conference on Logic for Programming and Automated Reasoning*, volume 1955 of *Lecture Notes in Computer Science*, pages 377–440, Île de la Réunion, France, 2000.

[5] Bruno Dutertre. Elements of mathematical analysis in PVS. In Joakim von Wright, Jim Grundy, and John Harrison, editors, *Proceedings of the 9th International Conference on Theorem Proving in Higher Order Logics TPHOL*, volume 1125 of *Lecture Notes in Computer Science*, pages 141–156, Turku, Finlande, 1996.

[6] Jacques D. Fleuriot. On the mechanization of real analysis in Isabelle/HOL. In Mark Aagaard and John Harrison, editors, *Proceedings of the 13th International Conference on Theorem Proving in Higher Order Logics*, volume 1869 of *Lecture Notes in Computer Science*, pages 145–161, Portland, OR, USA, 2000.

[7] Georges Gonthier. Formal proof – the four-color theorem. *Notices of the AMS*, pages 1382–1393, 2008.

[8] John Harrison. Constructing the real numbers in HOL. *Formal Methods in System Design*, 5(1-2) :35–59, 1994.

[9] J. le Rond D'Alembert. Recherches sur la courbe que forme une corde tendue mise en vibrations. In *Histoire de l'Académie Royale des Sciences et Belles Lettres (Année 1747)*, volume 3, pages 214–249. Haude et Spener, Berlin, 1749.

[10] Micaela Mayero. *Formalisation et automatisation de preuves en analyses réelle et numérique*. PhD thesis, Université Paris VI, 2001.

[11] Micaela Mayero. Using theorem proving for numerical analysis (correctness proof of an automatic differentiation algorithm). In Victor Carreño, César Muñoz, and Sofiène Tahar, editors, *Proceedings of the 15th International Conference on Theorem Proving and Higher-Order Logic*, volume 2410 of *Lecture Notes in Computer Science*, pages 246–262, Hampton, VA, USA, 2002.

[12] Guillaume Melquiond. Proving bounds on real-valued functions with computations. In Alessandro Armando, Peter Baumgartner, and Gilles Dowek, editors, *Proceedings of the 4th International Joint Conference on Automated Reasoning*, volume 5195 of *Lectures Notes in Artificial Intelligence*, pages 2–17, Sydney, Australie, 2008.

[13] Eric W. Weisstein. D'Alembert's Solution. From MathWorld–A Wolfram Web Resource `http://mathworld.wolfram.com/dAlembertsSolution.html`.