Available online at www.sciencedirect.com
ScienceDirect
C. R. Mecanique 336 (2008) 336–340
COMPTES RENDUS MECANIQUE
http://france.elsevier.com/direct/CRAS2B/

Différentiation automatique de la méthode asymptotique numérique typée : l'approche Diamant

Isabelle Charpentier *, Michel Potier-Ferry

Laboratoire de physique et mécanique des matériaux, UMR 7554, Île du Saulcy, 57045 Metz cedex 1, France

Reçu le 14 novembre 2007 ; accepté le 27 novembre 2007

Disponible sur Internet le 11 janvier 2008

Présenté par Évariste Sanchez-Palencia

Résumé

La Méthode Asymptotique Numérique (MAN) est une méthode de cheminement efficace basée sur des développements en série. Jusqu'à présent, ces développements étaient principalement effectués à la main. La Note discute l'automatisation des calculs de dérivées d'ordre élevé de la MAN et présente la bibliothèque de Différentiation Automatique (DA) Diamant. Comme tout outil de DA, Diamant est générique et peut être utilisée pour différentier le code de toute loi de comportement différentiable ce qui simplifie l'usage de la MAN de manière significative. Des performances numériques prouvent l'efficacité de l'approche Diamant. ***Pour citer cet article : I. Charpentier, M. Potier-Ferry, C. R. Mecanique 336 (2008).***

Abstract

Automatic differentiation of the asymptotic numerical method: the Diamant approach. The Asymptotic Numerical Method (ANM) is a competitive path following method based on truncated series. Until now, these Taylor expansions were mainly hand-written. The Note discusses the automation of high order derivative computations involved in the ANM and presents the Automatic Differentiation (AD) library Diamant. As any AD tool, the latter is generic and may be used to differentiate the code of any differentiable behaviour law what simplifies the implementation of the MAN in a significant manner. Numerical performances prove the efficiency of the Diamant approach. ***To cite this article: I. Charpentier, M. Potier-Ferry, C. R. Mecanique 336 (2008).***

Mots-clés : Informatique, algorithmique ; Continuation ; MAN ; Différentiation automatique

Keywords: Computer science; Continuation; ANM; Automatic differentiation

Abridged English version

Several numerical methods based on Taylor series have been discussed for the solution of various, sufficiently smooth, mechanical PDE problems. Among them, a path following method [1] and some optimal shape design technique [2] take advantage of truncated series for the computation of approximate solutions. Once the PDE problem

* Auteur correspondant.
Adresses e-mail : Isabelle.Charpentier@univ-metz.fr (I. Charpentier), michel.potierferry@univ-metz.fr (M. Potier-Ferry).

1631-0721/$ – see front matter
doi:10.1016/j.crme.2007.11.022

has been discretized and differentiated, computations are governed by following the same simple but efficient idea: the linear system to be solved is the same whatever the order of differentiation is, the right-hand side terms being different. In the case of the ANM, this system involves a tangent linear matrix. In both methods, the use of a direct solver is thus of peculiar interest since the CPU resource consumption may be shared over the different orders. The truncated series computed at a given point is then used to provide approximate solutions, in a vicinity of that point, without solving the linear system. These approximate solutions were successfully used either to follow the path in the continuation method [1] or within a gradient method when dealing with optimal shape design [2]. This Note is devoted to the automation of high order differentiation computations involved in the ANM.

The differentiation stage is not a restricting feature since related computations may be hidden to the user as performed in the MANLAB software [3]. The latter solves nonlinear problems written under the prescribed form $R(u) = L_0 + L(u) + Q(u, u)$ where functions L_0, L and Q are respectively constant, linear and quadratic functions provided by the user. Automatic Differentiation [4,2] (AD) is a more generic approach. The interested reader is referred to the AD community's website www.autodiff.org for a comprehensive overview of the available techniques, tools and usage. AD may be applied to general programs as it considers them as a sequence of elemental operations to be differentiated. High order AD tools rely on operator overloading as the vehicle of attaching high order derivative computations to the arithmetic operators and intrinsic functions provided by the programming language.

The ANM differentiation strategy appears somewhat different from the usual AD because high order derivative computations and solutions of the linear system are performed alternatively. In that context, the use of any classical AD tool, even it returns correct results, is not convenient since the differentiation is performed up to a maximal order fixed order K whatever the current order of differentiation is. The Automatic Differentiation library DIAMANT (as an acronym for *DIfférentiation Automatique de la Méthode Asymptotique Numérique Typée*) has been designed to be as efficient as the hand-coded ANM, to avoid the tedious and error-prone task of differentiation, and to hide the differentiation aspects to the user. As in any other AD tool, the last purpose is tackled by means of operator overloading capabilities provided by object-oriented languages such as `Fortran 90`, `C++` and `Matlab`. In the note, computer details are discussed within the `Fortran 90` framework. The same developments were adapted to `C++` and `MATLAB`. The first two goals are reached differentiating at order k only and considering full recurrence formulas (see Eq. (3)). This is a key-point since the ANM recurrence formulas may miss the term used in the construction of the tangent linear matrix. Implementation details may be observed on the files of the Diamant1 library.[1]

The Diamant series computations have been implemented following the rather simple algorithm presented in Table 1. Several data structures for the storage of the Taylor coefficients were used to measure the performances of Diamant with respect to the usual MAN [1] and the AD tool `rapsodia` [9]. As already noticed in [9], the data structure plays an important role in the efficiency of the derivative computations because of the memory access they require. Thus, in a fair comparison (Diamant2 vs MAN`Creal`), Table 2 proves the efficiency of the Diamant approach. The genericity of the Diamant approach makes easier the implementation of the ANM method which significantly compensates the small loss in efficiency noticeable in comparison with the user handcoded version of the ANM.

1. Introduction

Plusieurs méthodes numériques utilisant des développements en série de Taylor permettent de résoudre des problèmes de mécanique suffisamment réguliers. Parmi celles-ci, figurent la Méthode Asymptotique Numérique (MAN) [1] et une méthode d'optimisation de forme [2]. Dans ces deux cas, la différentiation du problème d'EDP discret, réalisée au moyen de formules de récurrence, fait apparaître un système linéaire dont la matrice est identique quel que soit l'ordre de différentiation. L'utilisation d'un solveur direct devient pertinente car le coût de décomposition de la matrice est « réparti » sur les différents ordres. Le développement en série évalué en un point est alors utilisé pour calculer, sans décomposition de la matrice, des solutions approchées au voisinage de ce point. Cette approche a prouvé son efficacité lors de la construction de chemins [1] et lors de la résolution de problèmes d'optimisation [2]. Malgré ces ressemblances, la phase de différentiation de la MAN est bien plus complexe que celle de son homologue car la MAN résout des problèmes d'EDP non linéaires réguliers ou régularisés, tandis que la technique d'optimisation s'applique à des problèmes d'EDP préalablement linéarisés.

[1] http://lpmm.sciences.univ-metz.fr/~charpentier/Diamant1.

Le développement en série est souvent délicat à exhiber pour un problème de mécanique concret. Comme montré dans cette Note, l'étape de différentiation peut être cachée à l'utilisateur. Tel est le cas dans le logiciel MANLAB [3] qui résout des problèmes non linéaires s'écrivant sous la forme $R(u) = L_0 + L(u) + Q(u,u)$ où L_0, L et Q sont respectivement des fonctions constante, linéaire et quadratique fournies par l'utilisateur. La technique de Différentiation Automatique (DA) [4,2] présentée dans cette note est plus générique. Les outils logiciels de DA, une liste exhaustive figure sur le site autodiff.org, permettent d'augmenter tout ou partie d'un code informatique pour lui faire calculer des dérivées, et notamment des dérivées de haut degré. Il est donc naturel de coupler DA et MAN. Cependant, un rapide calcul de complexité montre que les outils classiques de DA sont peu performants dans le cadre de la MAN car calculs de dérivées et résolutions du système linéaire sont effectués alternativement. De là est née l'idée de proposer la bibliothèque de différentiation automatique Diamant adaptée à la MAN et permettant de différentier des fonctions régulières quelconques.

La Note se présente comme suit. La Section 2 décrit les spécificités de la différentiation de la MAN, tandis que la Section 3 présente l'approche Diamant. Conclusions et perspectives figurent en Section 4.

2. Méthode asymptotique numérique

Considérons le problème générique (1) impliquant une matrice carrée inversible $A(p)$ et un second membre $b(p)$ dépendant de paramètres p décrivant des propriétés géométriques ou mécaniques, et une fonction non linéaire v :

$$A(p)u(a) + \lambda(a,p)v\big(u(a),p\big) = b(p) \tag{1}$$

Le vecteur inconnu $u(a,p)$ et le paramètre inconnu $\lambda(a,p)$ dépendent en sus du paramètre de longueur d'arc a usuellement choisi tel que :

$$a = \big\langle u(a) - u^n, u_1\big\rangle + \big(\lambda(a) - \lambda^n\big)\lambda_1 \tag{2}$$

où u_1 et λ_1 sont les dérivées premières de $u(a)$ et $\lambda(a)$. La dépendance en p est désormais omise.

La MAN propose de calculer la solution de (1)–(2) en quelques points (u^n, λ^n) calculés pour $a = 0$, puis de construire des solutions approchées entre ces points. La contrepartie du gain en « nombre de résolutions » est la différentiation de la fonction composée $V = v \circ u$. Soient $u_k^n = \frac{1}{k!}\frac{\partial^k u}{\partial a^k}(0)$, $\lambda_k^n = \frac{1}{k!}\frac{\partial^k \lambda}{\partial a^k}(0)$, $V_k = \frac{1}{k!}\frac{\partial^k (v\circ u)}{\partial a^k}(0)$ et $v_k = \frac{1}{k!}\frac{\partial^k v}{\partial u^k}(u^n)$ les $k^{\text{èmes}}$ coefficients de Taylor de u, λ, $v \circ u$ et v. Les équations satisfaites par a_k, u_k, λ_k et V_k sont obtenues en introduisant les séries de Taylor de a, λ, u et V dans (1)–(2). La différentiation en a est effectuée par le choix de $a_1 = 1$ et $a_k = 0$ $(k \neq 1)$. S'en déduisent les K systèmes d'équations suivants par identification des ordres de différentiation :

- système d'ordre 1 : $Au_1 + \sum_{l=0}^{1} \lambda_l V_{1-l} = 0$, $1 = \langle u_1, u_1\rangle + \lambda_1^2$,
- système d'ordre k : $Au_k + \sum_{l=0}^{k} \lambda_l V_{k-l} = 0$, $0 = \langle u_k, u_1\rangle + \lambda_k\lambda_1$, $\forall k = 2,\ldots,K$.

Ces systèmes sont résolus successivement en mettant en évidence, dans chacun d'eux, la matrice linéaire tangente K_T (également présente dans la méthode de Newton–Raphson) construite telle que $K_T u_1 = Au_1 + \lambda_0 v_1 u_1$. Elle peut être construite de manière automatique en choisissant pour u_1 les vecteurs de la base canonique. Du point de vue implémentation, une technique de coloration de graphe [4] permet de le faire à moindre coût. A l'ordre k, K_T apparaît en développant le terme V_k [5] puis en isolant la variable u_k. Les seconds membres sont construits en conséquence.

Pour illustrer cette Note, nous avons considéré le problème de Bratu [6] en choisissant $V(a) = v \circ u(a) = \exp(u(a))$ car ce cas test simple, distribué avec la bibliothèque Diamant, est précisément décrit dans [1]. En posant $\tilde{u}_k = ku_k$ et $\tilde{v}_k = kv_k$ (pour économiser des multiplications lors de l'implémentation), le coefficient de Taylor V_k peut s'écrire :

$$V_k = \sum_{l=1}^{k} \tilde{u}_l v_{k-l} = \sum_{l=1}^{k-1} \tilde{u}_l v_{k-l} + \tilde{u}_k v_0 \tag{3}$$

La partie $\tilde{u}_k v_0$ est, à un facteur multiplicatif près, celui pris en compte pour « isoler » K_T à l'ordre k. En conséquence, la formule de récurrence de l'exponentielle utilisée dans le cadre de la MAN (somme de droite) diffère d'un terme de la formule de récurrence de l'exponentielle usuellement codée en DA (somme de gauche). Ce détail est de prime

importance car l'automatisation de la différentiation n'est pas triviale à mettre en œuvre sur des formules de récurrence incomplètes.

L'objectif principal de la Note est de montrer que les développements en série de la MAN sont automatisables. Pour le faire, il suffit de remarquer que le choix de u_k et λ_k égaux à 0 permet de travailler avec des formules de récurrence complètes lors de la construction des seconds membres du système d'ordre k (les deux sommes de (3) sont égales lorsque $\tilde{u}_k v_0 = 0$). Cela est possible car les variables u_k et λ_k sont évaluées lors du « calcul du chemin » effectué après la résolution de ce système linéaire.

3. L'approche Diamant

L'approche Diamant (acronyme de *DIfférentiation Automatique de la Méthode Asymptotique Numérique Typée*) a été conçue pour occulter, à l'utilisateur, l'étape de différentiation de haut degré et être efficace dans ses calculs de séries. Par la suite, les détails d'implémentation sont proposés en langage `Fortran 90`. La même approche est possible en `C++`, `Matlab`, et plus généralement, dans tout autre langage de programmation permettant la surcharge d'opérateurs.

Le principe fondamental de la DA est que tout modèle différentiable ϕ peut être vu comme une séquence de fonctions élémentaires f. La différentiation de ϕ s'effectue alors en appliquant la régle de dérivation de la loi de composition des fonctions aux f la formant. Les dérivées d'ordre élevé se déduisent de formules de récurrence. Pour calculer les dérivées d'ordre élevé d'un code, les logiciels de DA disponibles sont en pratique `AD02` [7] pour `Fortran 90`, `AdolC` [8] pour `C` et `C++`, et `Rapsodia` [9] pour `Fortran 90` and `C++`. Tous reposent sur la technique de surcharge d'opérateurs pour attacher les calculs de dérivées aux opérateurs arithmétiques et aux fonctions mathématiques intrinsèques du langage de programmation.

L'organisation de la différentiation de haut degré de la MAN est différente de la DA usuelle. Les outils classiques peuvent effectuer ces calculs, mais ils le font avec une complexité algorithmique supérieure d'un ordre de grandeur à celle d'une différentiation manuelle. Contrairement aux autres outils de DA, Diamant se propose d'effectuer la différentiation à un ordre donné k en ne calculant ni les coefficients des ordres inférieurs ($1 \leqslant l \leqslant k-1$) (connus), ni ceux des ordres supérieurs ($k+1 \leqslant m \leqslant K$) (inutiles).

La bibliothèque Diamant utilise la programmation orientée-objet pour cacher la partie différentiation à ses utilisateurs. Elle utilise le type de données abstrait classiquement employé en DA, nommé `Creal` dans Diamant, pour stocker des coefficients de Taylor. Lorsqu'une variable `u` est de type `Creal`, ses composantes `u%d(k)` sont dédiées à la valeur de la variable u_0, et aux valeurs de ses coefficients de Taylor u_k ($k = 1, \ldots, K$). Dans l'algorithmique Diamant, toute variable ayant un rôle actif dans la différentiation doit être définie à l'aide de ce nouveau type par l'utilisateur. Formellement, procéder ainsi revient à « typer » la MAN. La fonctionnalité de « surcharge d'opérateurs » offerte par `Fortran 90` permet alors à une fonction (exp, log, ...) ou un opérateur ($+$, $*$, ...) de gérer ce type de données en implémentant les formules de récurrence correspondantes dans des modules. Regroupés, ces modules forment la bibliothèque Diamant. Le lecteur intéressé consultera les codes de la version libre de Diamant.[2]

L'algorithme Diamant du calcul en série de la MAN est présenté en Tableau 1 pour la fonction $W(u, v, \lambda) = \lambda v(u) = \lambda \exp(u)$ (de coefficients de Taylor w_k) dans laquelle la variable intermédiaire v sert à éviter le recalcul des coefficients de Taylor de $\exp(u)$. L'entier k figurant dans l'appel de $W(u, v, \lambda, k)$ est utilisé pour indiquer l'ordre de différentiation. A la première itération, le choix de $u_1 = 0$ et $\lambda_1 = 1$ permet d'obtenir le second membre w_1 en évaluant W. A l'itération k, l'évaluation de W s'effectue avec u_k et λ_k nulles pour construire les seconds membres. Après le « calcul du chemin », une seconde évaluation peut être nécessaire (version Diamant2) pour mettre à jour le $k^{\text{ème}}$ coefficient de Taylor de la variable intermédiaire v_k. Informatiquement, la routine évaluant W utilise les modules de la bibliothèque (instructions `USE <modules>`), définit les variables de `TYPE(Creal)` et contient les instructions de calcul `v=exp(u)` et `w=lambda*v`. La dérivation, implicitement présente au travers des modules et du type abstrait `Creal`, est mise en place à la compilation du code.

Les performances de différentiation présentées dans la Tableau 2 ont été obtenues en considérant plusieurs programmations. La version `MAN` [1] utilise tableaux et vecteurs traditionnels, tandis que `Rapsodia` stocke les coefficients à l'aide d'un type de données à composantes scalaires [9]. Les autres versions utilisent des variables `Creal`.

[2] http://lpmm.sciences.univ-metz.fr/~charpentier/Diamant1.

Tableau 1
Organisation par Diamant du calcul de séries de la MAN

```
INITIALISATION DES VARIABLES DE TYPE CREAL  u_l = 0,  λ_l = 0,  û est une variable réelle
CALCULS AUX ORDRES 0 ET 1 POUR CONSTRUCTION DE K_T :  initialisation de u_1 = 1,  λ_1 = 0
   évaluation de W(u, v, λ, 0) et W(u, v, λ, 1)
   construction et décomposition de la matrice linéaire tangente K_T
CALCULS À L'ORDRE k = 1 :  initialisation de u_1 = 0,  λ_1 = 1
   évaluation de W(u, v, λ, 1)  (avec u_1 = 0 pour retrait du terme pris en compte dans K_T)
   résolution du système linéaire :  K_T û = −w_1
   calcul du chemin :  λ_1 = 1/√(1 + ‖û‖²),  u_1 = λ_1 û
   évaluation de W(u, v, λ, 1)  (mise à jour de v_1 nécessaire dans la version Diamant2)
CALCULS AUX ORDRES SUPÉRIEURS :  k = k + 1 (récurrence)
   évaluation de W(u, v, λ, k)  (avec u_k = 0 pour retrait du terme pris en compte dans K_T)
   résolution du système linéaire :  K_T û = −w_k
   calcul du chemin :  λ_k = −λ_1(û^T . u_1),  u_k = −λ_k/λ_1 u_1 + û
   évaluation de W(u, v, λ, k)  (mise à jour de v_k nécessaire dans la version Diamant2)
```

Tableau 2
Performances normalisées des différentes approches de différentiation automatique

	MAN [1]	MAN`Creal`	Diamant1	Diamant2	Diamant0	Rapsodia
$K = 20$	1	6,5	13,7	5,0	31,5	22,7

Diamant0 est une DA classique, Diamant1 recalcule les coefficients de Taylor jusqu'à l'ordre k, Diamant2 ne différentie qu'à l'ordre courant. Seule Diamant2 nécessite la seconde évaluation de W. Ces résultats montrent la supériorité de Diamant2 sur la DA classique. On observe aussi que Rapsodia est plus rapide que Diamant0, ce qui met en évidence [9] l'influence du choix de la structure de données sur l'efficacité des accès mémoire. Diamant2 et MAN`Creal` ont des performances comparables. Le fait que MAN [1] soit la plus performante ne résulte donc que du choix des structures de données pour u et λ.

4. Conclusions

De par sa généricité, l'approche Diamant permet d'implémenter rapidement la MAN dans tout code éléments finis, ce qui compense largement la petite perte d'efficacité résultant du choix de la structure de données. Des bibliothèques Diamant écrites en C++ et Matlab sont en cours de validation.

Références

[1] B. Cochelin, N. Damil, M. Potier-Ferry, Méthode asymptotique numérique, Hermes Science Publications, 2007, p. 298.
[2] P. Guillaume, M. Masmoudi, Computation of high order derivatives in optimal shape design, Numer. Math. 67 (1994) 231–250.
[3] R. Arquier, Manlab : logiciel de continuation interactif (Manuel utilisateur), http://www.lma.cnrs-mrs.fr/~manlab/, 2005.
[4] A. Griewank, Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation, Frontiers in Appl. Math., vol. 19, SIAM, Philadelphia, 2000, p. 369.
[5] F. Faá di Bruno, Note sur un nouvelle formule de calcul différentiel, Quart. J. Math. 1 (1857) 359–360.
[6] E. Doedel, H. Keller, J.P. Kernevez, Numerical analysis and control of bifurcation problems (I). Bifurcation in finite dimensions, Internat. J. Bifurcation Chaos 1 (1991) 493–520.
[7] J. Pryce, J. Reid, AD01, a Fortran 90 code for automatic differentiation, Rutherford Appleton Laboratory, Chilton, Oxfordshire, England, RAL-TR-1998-057, 1998.
[8] A. Griewank, D. Juedes, J. Utke, ADOL-C, a package for the automatic differentiation of algorithms written in C/C++, ACM Trans. Math. Software 22 (1996) 131–167.
[9] I. Charpentier, J. Utke, Fast higher-order derivative tensors, soumis.