自动微分方法在 XIAMEN 软件优化中的应用*1)

陈晓宇[1] 程 强[1] 宋金帅[2] 迟学斌[1] 吴 玮[2]
(1. 中国科学院计算机网络信息中心超级计算中心, 北京 100190)
(2. 厦门大学化学系, 物理化学研究所, 固体表面物理化学国家重点实验室, 福建厦门 361005)

摘 要

比起有限差分方法来, 运用自动微分方法计算函数的梯度在计算时间和计算精度方面都具有明显的优势. 使用伴随模式计算函数的梯度, 在 XIAMEN 软件优化中得到了明显的加速效果. 使用 ADG 系统自动生成伴随模式, 大大降低了伴随模式的开发时间和难度. 重点讨论了伴随模式实现的几个关键难题, 并给出了几个典型应用的数值结果.

关键词: 自动微分; 梯度; 切线性模式; 伴随模式
MR (2000) 主题分类: 65D17

USING AUTOMATIC DIFFERENTIATION IN THE OPTIMIZATION OF XIAMEN SOFTWARE

[1]Chen Xiaoyu [1]Cheng Qiang [2]Song Jinshuan [1]Chi Xuebin [2]Wu Wei
(1. *Supercomputing Center of Chinese Academy of Science, Beijing 100190, China*)
(2. *Department of Chemistry, Institute of Physical Chemistry, key Laboratory of Solid Surface Physical Chemistry, Xiamen University, Xiamen 361005, Fujian, China*)

Abstract

Compared to the Finite Differecing method, the Automatic Differentiation method has significant advantage in terms of the running time and the computational precision. By using the adjoint model to claculate gradinents, the XIAMEN software is optimized with apperent speedup. The ADG sofware is used to automatically generate the adjoint model, which dramatically reducing the labors and the difficulty of the implementation process. Severel related techniques are discussed and several testing results are presented.

Keywords: automatic differentiation; gradient; tangent linear model; adjoint model
2000 Mathematics Subject Classification: 65D17

1. 引 言

XIAMEN 软件 [1] 是一个从头算的无自旋价键计算程序. 同国际上另外两个价键程序 TUR-TLE [2,3] 和 spin-couple VB [4] 相比, 它在计算时间、计算精度以及所处理体系的规模等

* 2007 年 7 月 19 日收到.
1) 本项研究工作得到国家自然科学基金项目 "自动微分方法研究及其实现"(60503031, 10871014) 和国家 973 项目 (2004CB418304) 和中国科学院重要方向项目 (KZCX3-SW-230) 的资助.

方面都具有明显的优势. XIAMEN 软件使用对不变式方法或行列式展开来计算 Hamiltonian 矩阵元和重叠矩阵元. 目前, XIAMEN 软件可用来计算价键态相关图和价键组态相互作用.

在 XIAMEN 软件中, 价键轨道是可以被优化的, 并使用变尺度算法 DFP [5] 和 DFP-BFS [6] 来进行. 优化算法主要使用了有限差分方法来计算能量函数的梯度. 当体系规模比较大时, 梯度的计算在整个程序运行时间中占了的绝大部分. 例如, 对应一个轨道系数数目为 205 的典型应用来说, 梯度的计算代价占了整个程序运行时间的 95% 以上. 因此, 如何降低梯度的计算代价, 对整个程序的优化效果甚为关键.

自动微分方法 [7−11] 是一系列基于链式求导法则的代码转换技术, 它通过各种预编译手段, 可以把一个数值程序代码转换成对应的计算微分的程序代码. 特别地, 在计算函数梯度方面, 自动微分方法和有限差分相比具有很大的优势. 无论采用切线性模式还是伴随模式 [12], 计算过程中都避免了截断误差. 其次, 应用伴随模式来计算函数梯度的时间代价仅有原程序运行时间代价的几倍而已, 并和独立变量的数目无关. 但是在很多情形下, 伴随模式需要大量的甚至无法承受的存储开销. 断点存储技术 [13] 通过增加少量浮点计算代价来大大降低存储开销, 从而提高伴随模式的实际性能.

伴随模式实现的另一个困难是分析复杂程序中各种对象之间的关系, 如变量的输入输出属性, 变量之间的依赖关系, 过程之间的相关性等. 本文通过使用伴随模式生成器 (ADG)[14,15], 大大降低了伴随模式的开发时间和困难, 使 XIAMEN 程序得到了明显的优化.

2. XIAMEN 程序的优化方法

在 XIAMEN 程序中, 轨道能量的计算主要包括 3 个步骤: 积分变换, 矩阵元的计算以及本征方程求解 [14]. 由于价键方法中轨道的非正交性, 使得体系能量难以直接写成单电子函数的简单表示式, 无法像分子轨道理论的 Hartree–Fock 方法那样将轨道优化问题化为解广义本征方程的问题. 因此, 目前程序中采用的优化方法为无约束优化方法.

在 XIAMEN 程序中, 轨道系数可通过求解以下最优化问题得到:

$$\min_{x\in R^n} E(x) \tag{2.1}$$

其中 $E(x)$ 为轨道的能量函数, x 为轨道系数, n 为轨道系数的数目. XIAMEN 软件使用拟牛顿法来求解上式, 做如下迭代:

1) 给定初值 x^0;

2) $x^{k+1} = x^k - H^k \cdot \nabla E(x^k)$.

其中 H^k 为海森矩阵逆的近似, 可由 DFP 算法计算得到. 在原程序中, 梯度 $\nabla E(x^k)$ 采用有限差分方法来计算, 计算代价在整个程序中占了较大比重. 因此, 本文重点讨论如何运用自动微分方法来降低梯度的计算代价.

2.1. 切线性模式计算梯度

切线性模式是微分代码的实现形式之一, 通过切线性模式计算函数的导数没有截断误差, 因而比有限差分方法具有更高的精度. 一个标准的数值程序对象通常可以用一个向量函数来

表示. 给定一个向量函数 $F : \mathbf{R^n} \to \mathbf{R^m}$, 假设其在给定点 (基态值)$X_0$ 处连续可微, 则其切线性模式可简单表示为

$$dY = \nabla_{X_0} F \cdot dX, \tag{2.2}$$

其中 $\nabla_{X_0} F$ 为 $m \times n$ 维雅可比矩阵, dX 和 dY 分别为输入和输出扰动向量. 如果令 $dX = e_i$, 则有 $dY = \partial F / \partial X_i$, 即雅可比矩阵的第 i 列.

运用切线性模式来计算梯度需要 $O(n)$ 的计算代价, 实际应用中大约是有限差分方法的两倍. 但在某些需要高精度计算雅可比矩阵 – 向量乘积的算法 (如 JFNK 方法 [13]) 应用中, 切线性模式简单易行, 具有明显的优势.

2.2. 伴随模式计算梯度

在上式中, 如果我们简单交换输入扰动和输出扰动向量的维数, 则可得到如下形式的伴随模式, 简单表示为

$$dX^* = \nabla^T_{X_0} F \cdot dY^*, \tag{2.3}$$

其中 $\nabla^T_{X_0} F$ 为雅可比矩阵的转置, dX^* 和 dY^* 分别为伴随模式的输出扰动和输入扰动. 特别地, 对于一个标量函数, 如果令 $dY^* = 1$, 则输出 $dX^* = \nabla_{X_0} F$ 即是梯度本身. 这一结论表明, 使用伴随模式来计算 (2.1) 式中能量函数的梯度具有理想的计算代价.

然而, 在大多数应用中, 开发伴随模式并不是一件容易的事. 我们知道, 非线性函数的导数计算依赖于其输入变量的值, 这里我们称为基态值. 在切线性模式中, 微分扰动的计算过程是沿着与原程序计算顺序一致的次序来进行, 因此任何当前位置的基态值都可通过上文的计算自然得到. 但在伴随模式中, 微分扰动的计算过程则是沿着与原程序计算顺序完全相反的次序来进行, 任何当前位置的基态值都可能被改写.

一般地, 要恢复当前位置的基态值通常有两种基本的方法, 即数据存储和重复计算 [14]. 事实上, 伴随模式的性能往往随这两种方法的不同组合而千差万别, 这种实现形式的多样性以及程序数据之间的复杂相关性给其开发带来了极大的困难.

从程序代码的规模来看, XIAMEN 程序的伴随模式有 7935 行, 正确性测试代码有 4356 行, 而原程序代码仅有 2773 行. 显然, 伴随模式开发的难度很大, 因此我们借助 ADG 系统来开发 XIAMEN 程序的伴随模式.

3. ADG 系统介绍

ADG 系统专门用于自动生成 Fortran77 程序的伴随模式代码, 它基于 YACC 和 C/C++ 开发而成. ADG 系统基于最小程序行为分解的模式伴随化方法实现, 在浮点计算和存储开销方面具有结构或划分深度依赖性的特点. 实践中, 由 ADG 系统生成的伴随模式一般用来计算函数的梯度, 比起有限差分方法来具有理想的计算代价.

3.1. 软件的功能

自动生成伴随模式. ADG 系统处理的基本程序对象为过程或函数, 它生成的伴随模式代码具有清晰的结构. 基本程序对象的接口为:

SUBROUTINE PROC(X_IN, X_OUT, X_IN_OUT, OTHERS)
FUNCTION FUNC(X_IN, X_OUT, X_IN_OUT, OTHERS)

其中 X_IN, X_OUT 和 X_IN_OUT 分别为实型输入变量、输出变量以及输入输出变量, OTHERS 为整型、字符型和逻辑型变量. 对应地, ADG 系统生成的伴随模式的接口为:

SUBROUTINE Adj_PROC(Diff_X_IN, X_IN, Diff_X_OUT, X_OUT,
Diff_X_IN_OUT, X_IN_OUT, OTHERS)
SUBROUTINE Adj_FUNC(Diff_X_IN, X_IN, Diff_X_OUT, X_OUT,
Diff_X_IN_OUT, X_IN_OUT, OTHERS, Diff_OUT, OUT)

这里需要说明的是, 在伴随模式中任何实型变量均随其微分变量成对出现, 其中 Diff_OUT 变量为一个函数对象的伴随模式的微分输入, 而 OUT 变量通常无意义.

自动生成测试程序. 对伴随模式在使用前必须做严格的测试, 通过 ADG 系统还可自动生成具有如下接口形式的测试程序:

SUBROUTINE Check_Adj_PROC(X_IN, X_OUT, X_IN_OUT, OTHERS)
FUNCTION Check_Adj_FUNC(X_IN, X_OUT, X_IN_OUT, OTHERS)

其中测试程序的参数表和原程序的参数表完全一致.

稀疏雅可比矩阵求解. 如果一个雅可比矩阵是稀疏的, 我们可以通过分析其稀疏结构来获得一个可行的输入矩阵 (即左乘或右乘输入矩阵) [11], 并通过反复使用切线性模式或伴随模式最终计算得到一个压缩的雅可比矩阵, 这大大减少了整个雅可比矩阵的计算代价. 对于复杂的情形, ADG 系统尚不能完全正确处理.

3.2. 软件的特色

基于最小程序行为分解的模式伴随化方法. 基于最小程序行为分解的模式伴随化方法 (LPBD) 是 ADG 系统的重要特色. LPBD 方法无论在软件实现、微分计算代价还是在算法通用性等方面都具有突出的优点. 首先, LPBD 方法具有很好的层次结构实现, 不同层次不同类型的程序对象具有相同或相似的伴随化实现过程, 并具有一致的微分代价. 其次, 按照 LPBD 方法实现的伴随模式在计算代价上具有结构依赖性. 伴随模式的浮点计算代价与具体划分方法和断点划分的数目无关, 仅与过程引用和划分的深度有关; 而空间存储开销也具有过程引用和划分深度依赖性, 断点数目与最大中间存储开销之间的折衷存在局部最优实现. 同时, 由于各划分模块相互共享中间存储空间, 这就大大降低了模式伴随化实现总的存储代价. 另外, LPBD 方法从程序语法结构出发来设计微分算法, 在自动微分领域具有普遍性, 得到的伴随模式具有结构严谨、代码简洁等优点.

静态数据相关分析. 准确分析各程序对象之间的相关性对于正确构造高效的伴随模式代码具有重要的意义. ADG 系统可处理多种形式的相关性, 包括数据输入输出 (IO) 相关分析、数据依赖相关分析、过程相关分析以及数据迭代相关分析等内容.

数据依赖相关与数据 IO 相关关系密切, 但又存在根本不同. 前者强调每个变量在数学关系上的依赖性; 而后者描述了一个数据对象的输入输出特性, 且具有相对性. 同时, 任何一个变量, 无论它是独立变量还是依赖变量, 在数学意义上都可等价为一个既是输入又是输出的变量来处理, 这一点对无法正确判断变量 IO 属性的伴随测试非常有用.

ADG 系统通过记录所有过程或函数哑元参数在所有局部计算中的 IO 属性, 并做深度递归计算, 来准确得到每个过程哑元参数的最终 IO 属性.

ADG 系统还通过对数据的初始相关矩阵做模二和迭代运算, 来最终完成数据的依赖相关分析, 该算法具有很好的对数收敛特性.

同时, ADG 系统通过全局过程相关分析, 可自动生成原程序中任意程序对象的相关引用树结构, 这有助于用户分析和测试一个复杂数值模式.

另外, ADG 系统还具有一定的分析数据迭代相关和函数迭代相关的能力.

结构化的微分实现. ADG 系统自动生成的伴随模式代码和伴随测试代码具有很强的结构化的特点. 通常, 伴随模式被分为变量定义、断点计算、中间扰动清零、伴随模式重建和数据复位五个部分.

ADG 系统采用标准化代码实现, 伴随程序中的扰动变量和基本态变量、微分计算语句按逆序实现, 具有清晰的程序结构. 微分代码保持了原模式本身的结构和风格 (如并行和向量特性、数据精度等), 即从语句到语句、结构到结构的微分实现. 在奇异点或不可导处, ADG 系统对微分扰动采取简单清零处理, 实践证明这对抑制扰动计算溢出具有积极意义, 但并不影响评价测试结果.

4. 伴随模式实现的关键技术

LPBD 算法和 IO 相关分析是 ADG 系统实现的两个关键技术. 前者能有效地降低伴随模式的存储开销, 后者有助于提高伴随模式代码的效率和确保测试程序的可靠性.

4.1. 最小程序行为分解算法

如前所述, 伴随模式的建立是从语句到语句、结构到结构的微分实现过程; 而原程序中每个非线性计算对象 (表达式、赋值语句、任意粒度的程序模块) 在做微分运算时都需要给出其对应位置的基态值. 数据存储方法通过事先运行原程序一次, 并依次保存原程序中所有非线性计算对象的基态值, 而在 “自下而上” 的伴随扰动计算过程中逐步恢复相应位置的基态值. 在最坏的情形下, 原程序中每个计算对象做伴随扰动计算之前都需事先进行存储, 即伴随模式中保存基态值所需的额外存储开销和原程序的浮点运算量成正比!

LPBD 算法将一个程序对象划分为一系列最小程序行为的顺序组合, 以最小程序行为做伴随模式实现的基本单位, 大大降低伴随模式的存储开销. 在文献 [12] 中证明, 采用 LPBD 算法得到的伴随模式在浮点计算量和存储开销方面同时具有结构或划分深度的依赖性.

图 1 表示了一个含 4 个基本计算单元 (EXP1∼EXP4) 的程序对象的伴随模式的实现过程, 其中左边方框为数据存储方法, 右边方框为最小程序行为分解方法. 一个程序对象按最小程序行为划分之后, 程序仅在几个断点处的基态值信息需要保存. 而在每个 LPB 内部, 则采取数据存储的方法来实现局部的伴随扰动计算. 事实上, 不同 LPB 之间的伴随扰动计算过程是相互独立的, 因而它们之间完成局部的伴随扰动计算的存储空间可以被共享, 这样采用最小程序行为分解方法所需的额外存储开销相当于所有 LPB 的基态值存储开销和一个 LPB 伴随实现的局部存储开销. 显然, 采用数据存储方法实现的伴随模式则需要更多的额外存储开销.

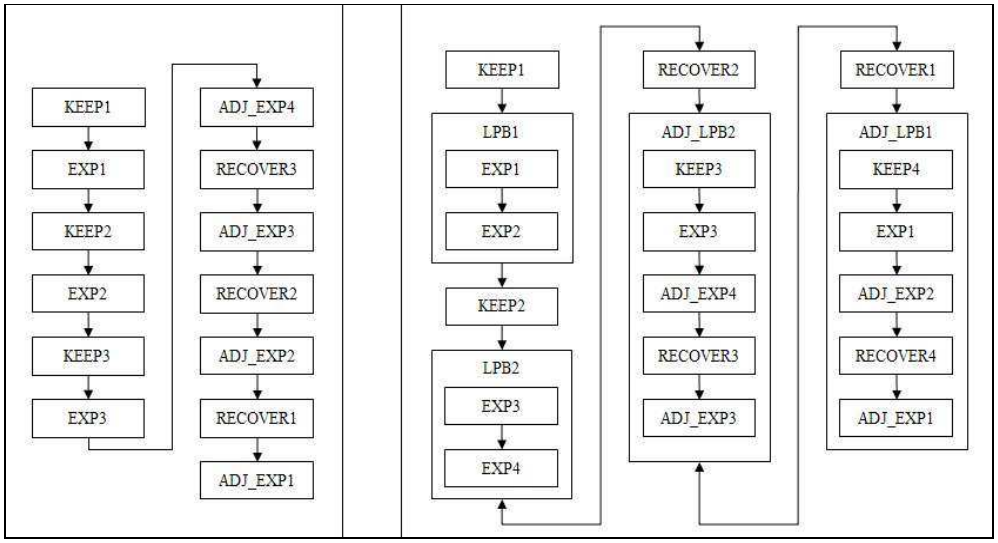

图 1 数据存储方法和 LPBD 算法的实现原理

4.2. IO 相关分析

事实上, 在 LPBD 算法的断点计算阶段, 一个 LPB 中只有这样的变量的基态值的保存是必要的, 即它具有输入属性, 同时在当前断点之后的下文中被重新赋值. 因此正确分析每一个 LPB 中每个变量局部的输入输出属性, 往往能够避免伴随模式中不必要的存储开销.

同时, 在伴随程序的测试过程中, 我们需要对具有不同 IO 属性的变量做不同的处理. 比如, 如果一个变量具有输入输出属性, 则需要在切线性模式执行之后伴随模式执行之前恢复该变量的基态值; 如果一个变量具有输入属性, 则须在伴随模式执行之前对其微分扰动清零, 等等. 实践中, 一个变量的 IO 属性分析的错误将可能直接影响到测试结果的可靠性. 在这种情形下, 我们可直接将该变量视为具有输入输出属性.

变量的 IO 属性可通过依次分析变量的引用和赋值过程来得到. 这里, 变量的 IO 属性包括输入属性 (IN)、输出属性 (OUT) 和输入输出属性 (IN_OUT). 特别地, 为了计算和软件实现的方便, 增加未知 IO 属性 (UNKNOWN). 一个变量的 IO 属性具有局部性, 即一个变量在一个程序段内的 IO 属性取决于该程序段内所有位置的 IO 属性, 并且可以通过依次计算得到. 具体地, 按照表一的运算规则, 从程序段起点到当前位置的 IO 属性可通过计算上文计算得到 IO 属性和当前位置的 IO 属性得到. 特别地, 如果一个变量在计算至当前位置的 IO 属性是输出或输入输出属性, 那么其最终 IO 属性就是输出或输入输出属性.

在一个实际的应用程序中, 一些特殊语法现象的运用可能使变量 IO 属性的分析变得相当复杂, 甚至根本不可能通过静态分析的方式得到. 在一个典型的 FORTRAN 程序中, 程序接口通常有三种表现方式, 即哑元参数表、公用块数据和文件读写. 通常, 哑元参数表的 IO 属性可以通过表 1 的运算规则简单运算得到. 对于文件读写的情形, 可以这样简单处理, 即当一个变量从一个文件读/写数据时, 其当前的 IO 属性就可视为输入/输出属性.

公用块变量在内存中具有单一的存储空间, 可以被一个或多个过程或函数中对应的公用块变量引用和改写. 程序中的公用变量的使用比较自由, 跟踪、记录和恢复其当前位置的基态值信息往往非常困难. 因此, 准确分析公用块变量的 IO 属性具有重要意义.

表 1 一个变量 IO 属性的基本运算规则

上文 IO 属性 \ 当前 IO 属性	UNKNOWN	IN	OUT	IN_OUT
UNKNOWN	UNKNOWN	IN	OUT	IN_OUT
IN	IN	IN	IN_OUT	IN_OUT
OUT	OUT	OUT	OUT	OUT
IN_OUT	IN_OUT	IN_OUT	IN_OUT	IN_OUT

如果一个公用变量在当前过程被用做局部 (或中间) 变量和输出变量, 该过程内任何当前位置的 IO 属性均可通过简单的局部计算得到. 但是, 如果一个公用变量在当前过程被用做输入变量, 而当前位置的数值无法分析出是在程序运行的 “何时” “何处” 被赋值的, 那么很难静态地分析出其当前位置的 IO 属性. 对此需另文专门讨论.

5. 数值试验结果

我们使用了三个体系来测试 XIAMEN 程序的优化效果, 分别为 CH_3F, GeI 和 F_2. 具体测试内容包括伴随模式的正确性、梯度的计算精度和时间代价. 这里, 我们首先给出 CH_3F 中伴随模式正确性的测试结果如下:

$$\begin{aligned} \langle \nabla F dX, \nabla F dX \rangle &= 3.5486430314196 \times 10^{-13} \\ \langle dX, \nabla^T F \nabla F dX \rangle &= 3.5486430314197 \times 10^{-13} \end{aligned}$$

表 2 通过三种方法计算轨道能量函数梯度

差分方法	切线性模式	伴随模式
-8.372068538811218E-4	-8.372501370376100E-4	-8.372501370376100E-4
1.921470537381989E-4	1.921725470010771E-4	1.921725469981574E-4
-2.681536177305398E-4	-2.680624679969122E-4	-2.680624679863531E-4
-6.897774261071955E-4	-6.896195883214099E-4	-6.896195883213525E-4
2.285359110402621E-5	2.270559742805618E-5	2.270559743318327E-5
3.462103247121377E-4	3.461801309012288E-4	3.461801309045601E-4
6.822754144720780E-5	6.842246780040918E-5	6.842246780481892E-5

接下来我们选择 GeI 体系来测试伴随模式的计算精度. 我们选取了第一次迭代中前 7 个轨道变量的导数值来比较, 其结果分别通过有限差分, 切线性模式以及伴随模式计算得到, 如表 2. 其中使用有限差分方法的结果是通过 20 次的重复计算, 期间不断改变扰动的大小, 并人工选取结果中最接近切线性模式计算结果的一个. 可以看出, 使用两种微分模式得到的导数基本相近. 而使用有限差分方法得到的结果虽然经过了多次反复计算, 最多也仅得到 3 位有效数字的精度.

表 3 计算梯度的加速效果

	轨道系数的数目	差分计算时间	伴随模式时间	梯度计算加速比
F_2	7	5.950	0.500	13.25
GeI	220	1602	23.50	68.16
CH_3F	205	1498	21.91	68.35

表 4 整个程序的优化效果

	轨道系数的数目	原程序时间	优化后时间	总加速比
F_2	7	6.370	0.870	7.349
GeI	220	1628	50.04	32.54
CH_3F	205	1518	42.04	36.10

最后我们给出了伴随模式计算梯度的加速比以及整个程序的优化效果. 运行环境为联想 6800 系统, 所有程序均为单 CPU. CPU 型号为安腾 64 位, 编译器是 Intel Fortran. 表 3 和表 4 分别为三个体系中计算梯度和整个程序运行的时间. 其中后两个应用由于轨道系数较多, 因此在梯度计算中得到了可观的优化效果. 而从整个程序的角度看加速比则有所下降, 但优化效果仍然显著.

6. 结 论

比起传统的有限差分方法来, 运用自动微分方法计算函数导数无论是在计算时间、计算精度还是在存储开销等方面都具有很大的优越性. ADG 系统的使用使得伴随模式开发的代价大大的降低. 伴随模式开发中我们使用了两个关键技术: LPBD 算法大大降低了伴随模式的存储开销; IO 属性分析则保证伴随模式的效率和正确性. 从数值测试结果来看, 使用自动微分方法优化后的 XIAMEN 程序得到了显著的优化效果.

参 考 文 献

[1] Wu W, Song L, Mo Y, et al. Xiamen-An ab initio spin-free valence bond (VB) program, Xiamen, Xiamen University, 1998.

[2] Dijkstra F. Valence Bond theory, implementation and use of analytical gradients. Universal, Holand, 1999.

[3] Verbeer J, Langenberg J H, Byrman C P, Dijkstra F and Lenthe J H van. TURTLE an Ab Initio VB/VBSCF program. Utrecht, 1998-1999.

[4] Haruyuki Nakano, Cooper D L, et al. J. Mole. Stru(Theochem). 1999, 461, 55.

[5] Powell M J D. On the convergence of the variable metric algorithms[J]. J. Inst. Math. Appl., 1971, 7: 21-36.

[6] Fletcher R. Practical Methods of Optimization. John Wiley and Sons, Inc., 2rd edition, 1987.

[7] Michael C. Bartholomew-Biggs, Steve Brown, Bruce Christianson, et. Automatic Differentiation of algorithms[J]. Journal of Computational and Applied Mathematics, 2000, 124: 171-190.

[8] Dimet Le, F. X and Talagrand O. Variational algorithms for analysis and assimilation of meteorological observations: theoretical aspects. Tellus, 1986, 38(A): 97-110.

[9] Mu Mu. Nonlinear singular vectors and nonlinear singular values. Science in China (D), 2000, 43: 375-385.

[10] Cacui D G. Sensitivity theory for nonlinear systems, II: Extension to additional classes of responses[J]. Journal of Mathematical Physics, 1981, 22(12): 2803-2812.

[11] Christian Bischof, Gordon Pusch and Ralf Knoesel. Sensitivity Analysis of the MM5 Weather Model using Automatic Differentiation. Computers in Physics, 1996, 0: 605-612.

[12] Andreas Griewank. On Automatic Differentiation, Mathematical Programming: Recent Developments and Applications (Amsterdam). M. Iri and K. Tanabe, eds., Kluwer Academic Publishers. 1989, 83-108.

[13] Cheng Qiang, Zhang Linbo and Wang Bin. Model Adjointisation and Its Costs. Science in China(F), 2004, 47(5): 587-611.

[14] Cheng Qiang, Wang Bin. ADG. Users' Guide, LASG, Institute of Atmospheric Physics (IAP), 2008.

[15] Cheng Qiang, Gao Jianwen, Wang Bin. Adjoint Code Generator. Science in China(F), accepted.