Yugoslav Journal of Operations Research
2 (1992), Number 1, 55–68

PARAMETER ESTIMATION IN LINEAR REGRESSION MODELS WITH STATIONARY ARMA(p,q)-ERRORS USING AUTOMATIC DIFFERENTIATION

Herbert FISCHER, Stefan SCHÄFFLER, Hubert WARSITZ

Technische Universität München
Institut für Angewandte Mathematik und Statistik
Arcisstr. 21, 8000 München 2, FRG

Abstract. The use of Automatic Differentiation for Time Series Analysis is considered. Especially we discuss the exact ML-estimation for linear regression models with stationary ARMA(p,q) residuals. The gradient and the Hessian matrix of the likelihood function, which has to be minimized, can be computed at fixed but arbitrary chosen points by Automatic Differentiation. The stationarity region for the ARMA(p,q) residuals is represented as a system of nonlinear inequalities. The special behavior of the likelihood function allows to use well-known methods for solving unconstrained nonlinear programming problems.

Key words and phrases: time series, linear regression, automatic differentiation

1. INTRODUCTION

One of the main problems in Time Series Analysis is the calculation of maximum-likelihood estimators. Especially concerning linear regression models with stationary ARMA(p,q) residuals, many investigations can be found in the literature, see e.g. Harvey and Phillips (1979) and the references cited there. The maximum-likelihood estimator can be characterized as the solution of some optimization problem. And to treat such a problem, gradient vector and Hessian matrix of the objective function are very useful. Our particular objective function is not easy to differentiate, because it involves implicitly defined functions. In the past, the exact treatment has been considered to be not manageable because of the complicated objective function and the difficult side conditions. In an approximate approach, derivatives mostly are replaced by quotients of differences. This common practice inevitably leads to the well-known predicament: a large discretization-stepsize yields inaccurate values and a small step size makes the computational process instable. But within the last decade it has become apparent that derivatives can be computed efficiently by Automatic Differentiation methods.

These ideas have been overseen or ignored for a long time, the breakthrough in Automatic Differentiation came with the work of Rall (1980, 1981, 1984, 1987), see

also Fischer (1987, 1988, 1989). The state of the art can be found in Griewank and Corliss (1991). Automatic Differentiation in statistics is e.g. used by Thacker (1990) and Sawyer (1984). Gradient vector and Hessian matrix of a fairly general function of several variables can be obtained "automatically" in an easy and straightforward manner. No discretization is involved, no manipulation of symbols is necessary. Furthermore, Automatic Differentiation makes it possible to treat functions which were out of reach previously, for instance: compute gradient vector and Hessian matrix of the determinant of a matrix, the entries of which are functions of several variables. This enables us to present a method for the exact ML-estimation. So, the intention of this paper is to demonstrate that Automatic Differentiation is a powerful device for Time Series problems.

In section 2 we introduce linear regression models with stationary ARMA(p,q) residuals and derive their likelihood functions, which have to be minimized. The next section provides appropriate optimization methods, a Newton method, a Broyden-Fletcher-Goldfarb-Shanno method, and theorems concerning the rate of convergence. Automatic Differentiation is described in section 4, where first we present the general idea and then focus on our particular Time Series problem.

2. THE MODEL AND THE LIKELIHOOD FUNCTION

Consider the linear model

$$y = X\beta + u. \tag{2.1}$$

$X \in \mathbb{R}^{T,r}$ is a given design matrix with $\operatorname{rank}(X) = r < T$, $y \in \mathbb{R}^T$ is a given vector, and the components $u_1, \ldots, u_T$ of u are ARMA(p,q), i.e.

$$\sum_{i=0}^{p} \Phi_i u_{t-i} = \sum_{j=0}^{q} \Theta_j \varepsilon_{t-j} \qquad \text{for } t = p+1, \ldots, T, \tag{2.2}$$

where $\Phi_0, \ldots, \Phi_p$ and $\Theta_0, \ldots, \Theta_p$ are real numbers, $\Phi_0 \equiv \Theta_0 \equiv 1$, and where the ε_τ are unobservable independent normal random variables with

$$E(\varepsilon_\tau) = 0, \quad E(\varepsilon_\tau^2) = \sigma^2, \quad E(\varepsilon_\tau, \varepsilon_\kappa) = 0 \quad \text{for } \tau, \kappa \in \mathbb{Z},\ \tau \neq \kappa. \tag{2.3}$$

For $\tau = \ldots, -1, 0, 1, \ldots$, (2.2) and (2.3) define a stochastic process that is stationary with u_t independent of $\varepsilon_{t+1}, \varepsilon_{t+2}, \ldots$, if and only if the Φ_i's are such that the principal minors $h_i(\Phi)$ of the matrix $Q(\Phi)$ with the components

$$q_{ij} = \sum_{k=0}^{\min(i,j)} (\Phi_{i-k}\Phi_{j-k} - \Phi_{p+k-i}\Phi_{p+k-j}) \qquad \text{for } i, j = 0, \ldots, p-1 \tag{2.4}$$

are positive, see Pagano (1973). Hence, we can state the stationarity condition as p nonlinear inequalities in $\Phi := (\Phi_1, \ldots, \Phi_p)^t$ by

$$h_i(\Phi) > 0 \qquad \text{for } i = 1, \ldots, p. \tag{2.5}$$

Let $\Theta := (\Theta_1, \dots, \Theta_q)^t$. Under stationarity of (2.2) we obtain the following likelihood function $L(\beta, \Phi, \Theta, \sigma)$

$$\begin{aligned} L(\beta, \Phi, \Theta, \sigma) &= (2\pi)^{-T/2} |\Sigma_T(\Phi, \Theta, \sigma)|^{-1/2} \times \\ &\qquad \times \exp\left\{-\tfrac{1}{2}(y - X\beta)^t \Sigma_T(\Phi, \Theta, \sigma)^{-1}(y - X\beta)\right\} \\ &= (2\pi)^{-T/2} (\sigma^2)^{-T/2} |V_T(\Phi, \Theta)|^{-1/2} \times \\ &\qquad \times \exp\left\{-\tfrac{1}{2\sigma^2}(y - X\beta)^t V_T(\Phi, \Theta)^{-1}(y - X\beta)\right\}, \end{aligned}$$

where $\Sigma_T(\Phi, \Theta, \sigma) = E(uu^t) \equiv \sigma^2 V_T(\Phi, \Theta)$ is given by a formula and positive definite.

The ML-estimator $(\hat{\beta}, \hat{\Phi}, \hat{\Theta}, \hat{\sigma})$ for $(\beta, \Phi, \Theta, \sigma)$ can be computed by solving the following minimization problem:

$$\min_{\beta, \Phi, \Theta, \sigma} \left\{ T \cdot \ln(\sigma^2) + \ln |V_T(\Phi, \Theta)| + \frac{(y - X\beta)^t V_T(\Phi, \Theta)^{-1}(y - X\beta)}{\sigma^2} \right\} \qquad (2.6)$$

subject to $h_i(\Phi) > 0$ for $i = 1, \dots, p$ and $\sigma > 0$.

The known matrix $V_T(\Phi, \Theta)$ comprises only T different entries $v_0(\Phi, \Theta), \dots, v_{T-1}(\Phi, \Theta)$ which are assumed to be twice continuously differentiable functions.

The optimization problem (2.6) is of the following form:

$$\min\{f(x) \mid h(x) > 0\}, \qquad f : \mathbb{R}^n \to \mathbb{R},\ f \in C^2,\ h : \mathbb{R}^n \to \mathbb{R}^{p+1},$$

where x denotes the four gathered variables β, Φ, Θ, σ. Therefore $n = r + p + q + 1$. The strict inequality constraints are such that they allow the application of minimization procedures for unconstrained problems with a special step size control.

3. OPTIMIZATION

In this chapter we describe two algorithms for solving the optimization problem (2.6). Most of the iterative algorithms for minimizing a given function F over a set $\Omega := \{x \in \mathbb{R}^n \mid h(x) > 0\}$ have the following structure:

Step 0) chose $x_0 \in \Omega$, set $j = 0$

Step 1) if $F'(x_j) = 0$ then stop

Step 2) chose a search direction s_j

Step 3) determine a step size τ_j such that with $x_{j+1} := x_j - \tau_j s_j$:

a) $x_{j+1} \in \Omega$

b) $F(x_{j+1}) < F(x_j)$

Step 4) $j = j + 1$, goto Step 1.

As seen in the general scheme, we have to specify the search direction and the step size.

From now on, we denote by g the gradient and by H the Hessian matrix of the objective function F, and we set $g_j = g(x_j) = F'(x_j)$ and $H_j = H(x_j) = F''(x_j)$.

First we propose a modified Newton algorithm. The search direction s_j is the solution of

$$H_j s_j = g_j \tag{3.1}$$

The step size τ_j will be chosen as follows:

Let $\delta \in (0, 1/2)$. Determine the smallest $\nu \in \{0, 1, 2, \dots\}$ such that with $\tau_j := (1/2)^\nu$ and with $x_{j+1} := x_j - \tau_j s_j$

$$\text{(a)} \quad h(x_{j+1}) > 0; \qquad \text{(b)} \quad F(x_{j+1}) \le F(x_j) - \delta \tau_j g_j^t s_j. \tag{3.2}$$

Condition (3.2)a ensures the positivity of the matrix $Q(\Phi)$. Hence, using the modified step size control (3.2) only an unconstrained minimization problem is to handle.

If (3.1) has no solution or more than one, there are a lot of possibilities to leave this critical situation with success. To describe these possibilities would exceed a reasonable size of this paper, details may be found in Dennis and Schnabel (1983), Fletcher (1983), Gill, Murray and Wright (1981).

The second proposed algorithm will not use the Hessian matrix of the objective function. The method approximates in a certain sense the inverse Hessian matrix of the objective function. This approximation is denoted by M_j. We start at $j = 0$ with the identity matrix as M_j and modify M_j in a special way to get M_{j+1}.

The search direction s_j will be computed by

$$s_j = M_j g_j,$$

and we shall get M_{j+1} with the following update:

$$M_{j+1} = M_j + \frac{d_j p_j + d_j^t M_j d_j}{(d_j^t p_j)^2} p_j p_j^t - \frac{p_j d_j^t M_j + M_j d_j p_j^t}{(d_j^t p_j)}. \tag{3.3}$$

In this formula d_j denotes $\|\tau_j s_j\|^{-1}(g_j - g_{j+1})$ and p_j denotes $\|s_j\|^{-1} s_j$. $\|\cdot\|$ denotes the Euclidean norm. This update is called the BFGS-update (Broyden-Fletcher-Goldfarb-Shanno).

The step size τ_j will be computed in a way slightly different from the above one:

Let $0 < \gamma_1 < \gamma_2 \le 1/2$. Determine the smallest $\nu \in \{0, 1, 2, \dots\}$ such that with $\tau_j := (1/2)^\nu$ and with $x_{j+1} := x_j - \tau_j s_j$.

$$\begin{aligned} &\text{(a)} \quad h(x_{j+1}) > 0 \\ &\text{(b)} \quad F(x_{j+1}) \le F(x_j) - \gamma_1 \tau_j g_j^t s_j \\ &\text{(c)} \quad g_{j+1}^t s_j \le \gamma_2 g_j^t s_j. \end{aligned} \tag{3.4}$$

Formula (3.4)a again ensures the positivity of $Q(\Phi)$.

To motivate this proposal we give two theorems. First, however, some definitions.

DEFINITION. A function $F : \mathbb{R}^n \to \mathbb{R}$ will be called *uniformly convex*, if F is twice continuously differentiable and there exist $\alpha, \zeta \in \mathbb{R}$ with $0 < \alpha < \zeta$ so that for all $x, y \in \mathbb{R}^n$ it holds

$$\alpha\|x\|^2 \leq x^t H(y)x \leq \zeta\|x\|^2.$$

Let the sequence $\{x_j\}$ be convergent to $\bar{x}$, then $\{x_j\}$ converges

$$\textit{superlinearly}, \quad \text{if} \quad \lim_{j\to\infty} \frac{\|x_{j+1} - \bar{x}\|}{\|x_j - \bar{x}\|} = 0,$$

$$\textit{quadratically}, \quad \text{if} \quad \overline{\lim_{j\to\infty}} \frac{\|x_{j+1} - \bar{x}\|}{\|x_j - \bar{x}\|^2} < \infty.$$

THEOREM 1. *Let F be uniformly convex, $\{x_j\}$ be generated with the Newton search direction and step size control (3.2) and $x_0 \in \Omega$ be an arbitrary point, then*

a) *the algorithm is practicable*

b) *$\{x_j\}$ converges superlinearly to the unique global minimizer $\bar{x}$ of F*

c) *if there exists $L \in \mathbb{R}$ with $\|H(x) - H(y)\|_M \leq L\|x - y\|$ for all $x, y \in \mathbb{R}^n$, then $\{x_j\}$ converges quadratically.*

Here

$$\|A\|_M = \sup_{x\in\mathbb{R}^n} \frac{\|Ax\|}{\|x\|}, \qquad A \in \mathbb{R}^{nn}.$$

PROOF. Ritter (1982).

The motivation for Theorem 1 is that if the Hessian matrix of a convex objective function at an isolated minimum $\bar{x}$ is positive definite, there exists a neighborhood of $\bar{x}$, where the objective function is uniformly convex.

THEOREM 2. *Let $x_0 \in \Omega$ and $\{x_j\}$ be generated by the BFGS-update and the step size control (3.4).*

a) *If F is continuously differentiable and convex, furthermore twice continuously differentiable on $S_0 := \{x \mid F(x) \leq F(x_0)\}$, and S_0 is bounded, then*

$\|g_j\| \to 0$ as $j \to \infty$ and every cluster point of $\{x_j\}$ will be a global minimizer.

b) *If additionally to a) $\{x_j\}$ converges to $\bar{x}$ with $g(\bar{x}) = 0$ and F is twice continuously differentiable on an open convex neighborhood U of $\bar{x}$ and $H(\bar{x})$ is positive definite, furthermore there exists $L \in \mathbb{R}$ with*

$$\|H(x) - H(y)\|_M \leq L\|x - y\| \qquad \textit{for all } x, y \in U,$$

then $\{x_j\}$ will converge superlinearly.

PROOF. Ritter (1982).

Finally we remark that, if in problem (2.6) the iteration sequence $\{x_j\}$ converges to the border of the feasible region, then the model established is not a stationary process. Hence, a new model has to be searched for the given data.

A great difficulty in applying the above algorithms is the provision of the first and second derivative of the objective function f. This difficulty will be avoided by using Automatic Differentiation. This technique delivers gradient and Hessian matrix of a function at arbitrary points exact up to rounding errors. No explicit formula for the gradient and the Hessian matrix will be needed. This method shall be lined out in the following.

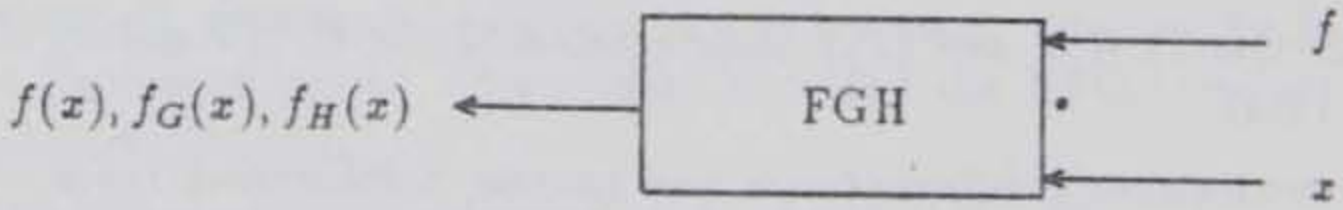
4. AUTOMATIC DIFFERENTIATION

In this section we first describe the basic ideas of Automatic Differentiation, and then we focus on our specific objective function. For the seek of simplification we use the forward mode, which additionally supports a parallel implementation in a natural way in opposite to the reverse mode, which would be applicable for our special problem, too. Let $f : D \subseteq \mathbb{R}^n \to \mathbb{R}$ be a twice differentiable function, let $f_G(x)$ and $f_H(x)$ denote the gradient resp. Hessian matrix of f at $x \in D$. Recall that

$$f_G(x) = \left[\frac{\partial f(x)}{\partial x_1} \quad \cdots \quad \frac{\partial f(x)}{\partial x_n}\right]^T \quad \text{is a vector in } \mathbb{R}^n,$$

$$f_H(x) = \begin{bmatrix} \frac{\partial^2 f(x)}{\partial x_1 \partial x_1} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \vdots & & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \cdots & \frac{\partial^2 f(x)}{\partial x_n \partial x_n} \end{bmatrix} \text{is a symmetric matrix in } \mathbb{R}^{nn}.$$

Define a black box FGH, which accepts the function f and the point x, and which produces the triple $f(x)$, $f_G(x)$, $f_H(x)$.



Our aim is to implement the black box FGH for fairly general f and arbitrary $x \in D$.

To begin with, we state trivial cases.

Let $r : D \to \mathbb{R}$ be a constant function, that means $r(x) = c$ for all $x \in D$. Then

$$r_G(x) = \text{zero-vector}, \qquad r_H(x) = \text{zero-matrix}.$$

Let $r : D \to \mathbb{R}$ be a projection, that means $r(x) = x_i = i$-th component of x. Then

$$r_G(x) = i\text{-th unit-vector}, \qquad r_H(x) = \text{zero-matrix}.$$

Hence, the black box FGH can be implemented for constant functions and projections. Though this is trivial, we need it as basis for the sequel.

Now we consider two familiar ways to build new functions from old ones: the rational composition and the library composition. Assume that the functions

$$a : D \subseteq \mathbb{R}^n \to \mathbb{R} \quad \text{and} \quad b : D \subseteq \mathbb{R}^n \to \mathbb{R}$$

are twice differentiable. Assume further that r is one of the functions $a+b$, $a-b$, $a \cdot b$, a/b with the provision $b(x) \neq 0$ in the case $r = a/b$. Then r is twice differentiable too. Let us mark the gradient and the Hessian matrix of a function by the subscript G resp. H. Table 1 shows formulas for r_G and r_H.

Table 1. Gradient and Hessian matrix of rational composition:

function	gradient	Hessian matrix
$r = a + b$	$r_G = a_G + b_G$	$r_H = a_H + b_H$
$r = a - b$	$r_G = a_G - b_G$	$r_H = a_H - b_H$
$r = a \cdot b$	$r_G = b \cdot a_G + a \cdot b_G$	$r_H = b \cdot a_H + a_G \cdot b_G^t + a \cdot b_H + b_G \cdot a_G^t$
$r = a/b$	$r_G = (a_G - r \cdot b_G)/b$	$r_H = (a_H - r_G \cdot b_G^t - r \cdot b_H - b_G \cdot r_G^t)/b$

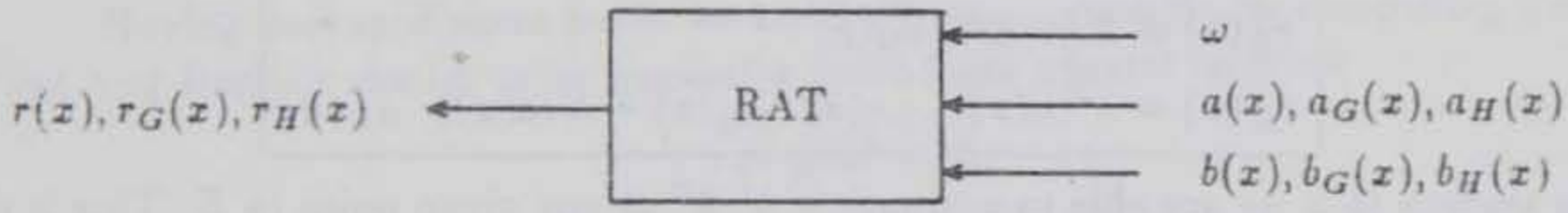
We strictly distinguish between functions and function values. So it should be clear that the table above shows equations of functions. Applying any of the functions r, r_G, r_H to some $x \in D$, we get an equation of function values. For instance in case of multiplication ($r = a \cdot b$), we obtain

$$r_H(x) = b(x) \cdot a_H(x) + a_G(x) \cdot b_G^t(x) + a(x) \cdot b_H(x) + b_G(x) \cdot a_G^t(x).$$

From the formulas in Table 1 we conclude:

For given $x \in D$, the triple $r(x)$, $r_G(x)$, $r_H(x)$ can be computed from the triples $a(x)$, $a_G(x)$, $a_H(x)$ and $b(x)$, $b_G(x)$, $b_H(x)$.

This observation allows to define a black box RAT, which accepts the type of $\omega \in \{+, -, \cdot, /\}$ and the triples $a(x)$, $a_G(x)$, $a_H(x)$ and $b(x)$, $b_G(x)$, $b_H(x)$, and which produces the triple $r(x)$, $r_G(x)$, $r_H(x)$.



Note that the triple $r(x)$, $r_G(x)$, $r_H(x)$ is not a triple of formulas, nor is it a triple of functions, it rather is an element of $\mathbb{R} \times \mathbb{R}^n \times \mathbb{R}^{nn}$. It is obvious that the black box RAT can easily be implemented as a procedure in PASCAL, as a subroutine in FORTRAN, or as a function in a more powerful programming language. As an alternative the black box RAT can be replaced by four black boxes, each of these designed for one particular type of ω.

A first very short example shall show the use of FGH and RAT. Consider the function

$$f : D \subseteq \mathbb{R}^3 - \mathbb{R} \quad \text{with} \quad f(x) = \frac{x_1 \cdot x_2}{x_3}.$$

Define the functions $f_1, f_2, f_3, f_4, f_5 : D \to \mathbb{R}$ by

$$\begin{aligned} f_1(x) &= x_1 \\ f_2(x) &= x_2 \\ f_3(x) &= x_3 \\ f_4(x) &= f_1(x) \cdot f_2(x) \\ f_5(x) &= f_4(x)/f_3(x). \end{aligned}$$

Of course, $f_5 = f$. For given $x \in D$ we compute

$$\begin{aligned} F_1 &\longleftarrow \mathrm{FGH}(f_1, x) \\ F_2 &\longleftarrow \mathrm{FGH}(f_2, x) \\ F_3 &\longleftarrow \mathrm{FGH}(f_3, x) \\ F_4 &\longleftarrow \mathrm{RAT}(\cdot, F_1, F_2) \\ F_5 &\longleftarrow \mathrm{RAT}(/, F_4, F_3) \end{aligned}$$

So we obtain the triple $F_5 = (f_5(x), f_{5G}(x), f_{5H}(x)) = (f(x), f_G(x), f_H(x))$. This example indicates that the black box FGH can be implemented for *any* explicitly given rational function.

Next we turn to library composition. Let Λ be a collection of real functions of one real variable. For brevity, these functions are called library functions. One may choose sin, ln, ... and the like as library functions.

Assume now that $r = \lambda \circ a$, where $\lambda : E \to \mathbb{R}$ is a library function with first derivative λ' and second derivative λ'' and $a : D \to \mathbb{R}$ is twice differentiable. Then r is twice differentiable, too. Table 2 shows formulas for r_G and r_H.

Table 2. Gradient and Hessian matrix of library composition.

$r(x) = \lambda(a(x))$
$r_G(x) = \lambda'(a(x)) \cdot a_G(x)$
$r_H(x) = \lambda''(a(x)) \cdot a_G(x) \cdot a_G^t(x) + \lambda'(a(x)) \cdot a_H(x)$

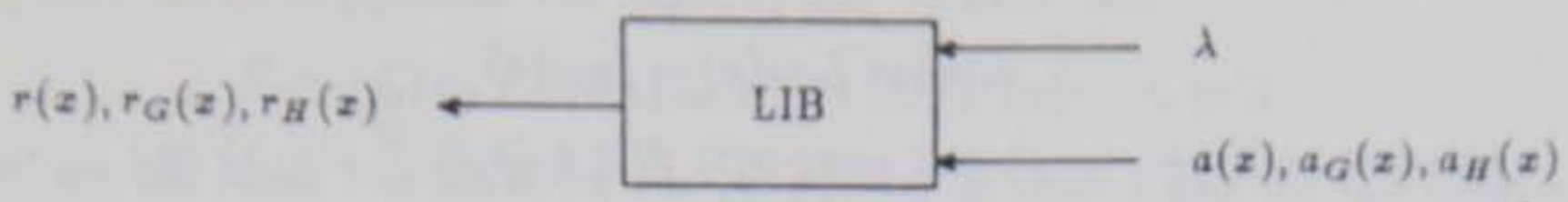
We assume that we are able to evaluate $\lambda, \lambda', \lambda''$ at any given point in E. This is no problem as long as λ is one of the commonly used library functions sin, ln, sqrt, ... and the like. The mechanism to get the triple $r(x)$, $r_G(x)$, $r_H(x)$ from the triple $a(x)$, $a_G(x)$, $a_H(x)$ does not depend on the particular x, it does not even depend on the values $a(x)$, $a_G(x)$, $a_H(x)$, it merely is a matter of the library function λ.

From the formulas in Table 2 we conclude:

For given $x \in D$, the triple $r(x)$, $r_G(x)$, $r_H(x)$ can be computed from the triple $a(x)$, $a_G(x)$, $a_H(x)$ using $\lambda, \lambda', \lambda''$.

This observation allows to define a black box LIB, which accepts the name of $\lambda \in \{\sin, \ln, \mathrm{sqrt}, \ldots\}$ and the triple $a(x)$, $a_G(x)$, $a_H(x)$, and which produces the

triple $r(x)$, $r_G(x)$, $r_H(x)$.



It is obvious that the black box LIB can easily be implemented as procedure in PASCAL, as subroutine in FORTRAN, or as function in a more powerful programming language. As an alternative the black box LIB can be replaced by a collection of black boxes, each of these designed for one particular library function.

A second very short example shall show the use of FGH, RAT and LIB. Consider the function

$$f : D \subseteq \mathbb{R}^2 \to \mathbb{R} \quad \text{with} \quad f(x) = \sin(x_1 + \ln(x_2)).$$

Define the functions $f_1, f_2, f_3, f_4, f_5 : D \to \mathbb{R}$ by

$$\begin{aligned} f_1(x) &= x_1 \\ f_2(x) &= x_2 \\ f_3(x) &= \ln(f_2(x)) \\ f_4(x) &= f_1(x) + f_3(x) \\ f_5(x) &= \sin(f_4(x)). \end{aligned}$$

Of course, $f_5 = f$. For given $x \in D$ we compute

$$\begin{aligned} F_1 &\longleftarrow \mathrm{FGH}(f_1, x) \\ F_2 &\longleftarrow \mathrm{FGH}(f_2, x) \\ F_3 &\longleftarrow \mathrm{LIB}(\ln, f_2) \\ F_4 &\longleftarrow \mathrm{RAT}(+, F_1, F_3) \\ F_5 &\longleftarrow \mathrm{LIB}(\sin, F_4). \end{aligned}$$

So we obtain the triple $F_5 = (f_5(x), f_{5G}(x), f_{5H}(x)) = (f(x), f_G(x), f_H(x))$.

Having prepared some tools, we are ready to concentrate on computing gradient and Hessian matrix of an explicitly given fairly general function

$$f : D \subseteq \mathbb{R}^n \to \mathbb{R}.$$

The phrase "explicitly given" shall mean that for $f(x)$ we have a formula which is composed of the components of x, some real constants, the rational operations $+, -, \cdot, /$, some library functions, and parentheses at proper places. Hence, we can set up a *characterizing sequence* $f_1, f_2, \ldots, f_s$ of functions $f_i : D \subseteq \mathbb{R}^n \to \mathbb{R}$ such that

(1) for $i = 1, \ldots, n$
$f_i(x) = x_i = i$-th component of x,

(2) for $i = n+1, \ldots, n+d$ with some $d \in \{0, 1, 2, \ldots\}$
$f_i(x) = c_i =$ real constant,

(3) for $i = n+d+1, \ldots, s$

$$f_i(x) = f_{\alpha(i)}(x)\, \omega_i\, f_{\beta(i)}(x)$$

with some $\omega_i \in \{+, -, \cdot, /\}$ and some $\alpha(i), \beta(i) \in \{1, 2, \ldots, i-1\}$ or

$$f_i(x) = \lambda_i(f_{\alpha(i)}(x))$$

with some $\lambda \in \Lambda$ and some $\alpha(i) \in \{1, 2, \ldots, i-1\}$,

(4) $f_s(x) = f(x)$.

A priori, the characterizing sequence $f_1, f_2, \ldots, f_s$ has nothing to do with differentiation, it merely describes how to compute $f(x)$ algorithmically. But in view of RAT and LIB it is a convenient means for computing $f_G(x)$ and $f_H(x)$.

We assume that all library functions used in the sequence $f_1, f_2, \ldots, f_s$ are twice differentiable. This guarantees that the given function f is twice differentiable. To the sequence $f_1, f_2, \ldots, f_s$ there correspond the sequences $f_{1G}, f_{2G}, \ldots, f_{sG}$ and $f_{1H}, f_{2H}, \ldots, f_{sH}$. Now we define

$$F_i := (f_i(x), f_{iG}(x), f_{iH}(x)) \qquad \text{for } i = 1, \ldots, s$$

and consider the sequence $F_1, F_2, \ldots, F_s$ of triples. For $i = 1, \ldots, n+d$ the triple F_i is obvious. And for $i = n+d+1, \ldots, s$ in this order we know the mechanism how to compute the triple F_i

— from two triples already available in case f_i is a rational composition,

— from one triple already available together with $\lambda_i, \lambda_i', \lambda_i''$ in case f_i is a library composition.

For given $x \in D$ the gradient $f_G(x)$ and the Hessian matrix $f_H(x)$ can be computed by the following Algorithm 1.

Algorithm 1. Computation of $f(x)$, $f_G(x)$, $f_H(x)$

Step 1: For $i = 1, \ldots, n$
$f_i(x)$ ⟵ x_i = i-th component of x, given
$f_{iG}(x)$ ⟵ i-th unit vector in $\mathbb{R}^n$
$f_{iH}(x)$ ⟵ zero-matrix in $\mathbb{R}^{nn}$
F_i ⟵ $(f_i(x), f_{iG}(x), f_{iH}(x))$.

Step 2: For $i = n+1, \ldots, n+d$
$f_i(x)$ ⟵ c_i = real constant, predefined
$f_{iG}(x)$ ⟵ zero-vector in $\mathbb{R}^n$
$f_{iH}(x)$ ⟵ zero-matrix in $\mathbb{R}^{nn}$
F_i ⟵ $(f_i(x), f_{iG}(x), f_{iH}(x))$.

Step 3: For $i = n+d+1, \ldots, s$
F_i ⟵ $\mathrm{RAT}(\omega_i, F_{\alpha(i)}, F_{\beta(i)})$
or
F_i ⟵ $\mathrm{LIB}(\lambda_i, F_{\alpha(i)})$.

This algorithm produces the triple F_s and we know that

$$F_s = (f_s(x), f_{sG}(x), f_{sH}(x)) = (f(x), f_G(x), f_H(x)).$$

If we use the black box FGH for the projections in Step 1 and for the constant functions in Step 2, we get a more concise form for Algorithm 1:

Algorithm 1. Computation of $f(x)$, $f_G(x)$, $f_H(x)$ — concise form

Step 1: For $i = 1, \ldots, n$
$F_i \longleftarrow \mathrm{FGH}(f_i, x)$.

Step 2: For $i = n+1, \ldots, n+d$
$F_i \longleftarrow \mathrm{FGH}(f_i, x)$.

Step 3: For $i = n+d+1, \ldots, s$
$F_i \longleftarrow \mathrm{RAT}(\omega_i, F_{\alpha(i)}, F_{\beta(i)})$
or
$F_i \longleftarrow \mathrm{LIB}(\lambda_i, F_{\alpha(i)})$.

A comparison of the characterizing sequence $f_1, f_2, \ldots, f_s$ and Algorithm 1 shows that a program for computing $f(x)$ can easily be transformed into a program for computing the triple $f(x)$, $f_G(x)$, $f_H(x)$. Loosely speaking we may say: If we have an algorithmic description for computing $f(x)$, and if we run through this description with triples instead of reals, with $\mathrm{RAT}(*, u, v)$ instead of $u * v$, and with $\mathrm{LIB}(\lambda, u)$ instead of $\lambda(u)$, we obtain $f(x)$, $f_G(x)$, $f_H(x)$ instead of $f(x)$. Furthermore, the close relation between the characterizing sequence $f_1, f_2, \ldots, f_s$ and the Algorithm 1 demonstrates that the black box FGH can be implemented for a fairly broad class of explicitly given functions.

Let us focus on our particular objective function $f : D \subseteq \mathbb{R}^n \to \mathbb{R}$ with

$$f(\beta, \Phi, \Theta, \sigma) = T \cdot \ln(\sigma^2) + \ln|V_T(\Phi, \Theta)| + \frac{(y - X\beta)^t V_T(\Phi, \Theta)^{-1}(y - X\beta)}{\sigma^2}. \quad (4.1)$$

The enterprise to compute gradient and Hessian matrix of f seems to be hopeless, for more than one reason. Firstly, how should we deal with the determinant $|V_T(\Phi, \Theta)|$? Secondly, we know the matrix $V_T(\Phi, \Theta)$, but what about its inverse? Anyway, there is no characterizing sequence for f at hand. Not yet. Of course, the determinant of a matrix $M \in \mathbb{R}^{TT}$ is a rational function of its entries,

$$|M| = \sum \pm M_{1k_1} \cdot M_{2k_2} \cdot \ldots \cdot M_{Tk_T}$$

where the sum has to be taken over all $T!$ permutations $(k_1, k_2, \ldots, k_T)$ of $(1, 2, \ldots, T)$, and the entries of M^{-1} are rational functions of the entries of M, by Cramer's rule. But this approach together with characterizing sequences for the entries of $V_T(\Phi, \Theta)$, would yield an astronomically long characterizing sequence for f. Fortunately, a characterizing sequence for a function is not unique, and we shall present a manageable one for f.

In order to get a more uniform notation we set $n = r + p + q + 1$ and

$$x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}, \quad \beta = \begin{bmatrix} x_1 \\ \vdots \\ x_r \end{bmatrix}, \quad \Phi = \begin{bmatrix} x_{r+1} \\ \vdots \\ x_{r+p} \end{bmatrix}, \quad \Theta = \begin{bmatrix} x_{r+p+1} \\ \vdots \\ x_{r+p+q} \end{bmatrix}, \quad \sigma = x_n, \tag{4.2}$$

and we define for $x \in D$

$$m(x) := V_T(\Phi, \Theta) \quad = \text{matrix in } \mathbb{R}^{TT}, \tag{4.3}$$

$$d(x) := |V_T(\Phi, \Theta)| = \text{real number}, \tag{4.4}$$

$$e(x) := y - X\beta \quad = \text{vector in } \mathbb{R}^T, \tag{4.5}$$

$$c(x) := \sigma^2 \quad = \text{real number}. \tag{4.6}$$

Then our objective function reads

$$f(x) = T \cdot \ln c(x) + \ln d(x) + \frac{e(x)^t m(x)^{-1} e(x)}{c(x)}. \tag{4.7}$$

We recall that the matrix $m(x)$ is positive definite. Hence, we may take advantage of the Cholesky-factorization. We are allowed to write $m(x)$ in the form

$$m(x) = l(x) \cdot l(x)^t \tag{4.8}$$

where $l(x)$ is a lower triangular matrix. This implies

$$m(x)^{-1} = (l(x)^{-1}) \cdot l(x)^{-1}. \tag{4.9}$$

Using the abbreviation

$$s(x) := l(x)^{-1} e(x) \tag{4.10}$$

the formula (4.7) for $f(x)$ can be rewritten as

$$f(x) = T \cdot \ln c(x) + \ln d(x) + \frac{s(x)^t s(x)}{c(x)}. \tag{4.11}$$

By the way, the previous problem with the determinant has vanished, because $d(x) = |m(x)| = |l(x)| \cdot |l(x)^t|$, and that yields

$$d(x) = \left(l_{11}(x) \cdot l_{22}(x) \cdot \ldots \cdot l_{TT}(x)\right)^2. \tag{4.12}$$

It remains to tell how $l(x)$ and $s(x)$ can be obtained algorithmically. The significant entries of the matrix $l(x)$ are

$$\begin{aligned} &\text{for } k = 1, \ldots, T, && l_{kk}(x) = \sqrt{m_{kk}(x) - \sum_{\rho=1}^{k-1} l_{k\rho}(x) \cdot l_{k\rho}(x)} \\ &\text{for } j = k+1, \ldots, T, && l_{jk}(x) = \left(m_{jk}(x) - \sum_{\rho=1}^{k-1} l_{j\rho}(x) \cdot l_{k\rho}(x)\right) / l_{kk}(x). \end{aligned} \tag{4.13}$$

Of course, the algorithmic description (4.13) defines a sequence of functions, where each one is a rational composition or a library composition. This sequence starts with l_{11} and ends with l_{TT}, we denote it by

$$l_{11}, \ldots, l_{TT}. \tag{4.14}$$

The components of $s(x)$ are

$$\text{for } j = 1, \ldots, T, \qquad s_j(x) = \Big(e_j(x) - \sum_{\nu=1}^{j-1} l_{j\nu}(x) \cdot s_\nu(x)\Big) / l_{jj}(x). \tag{4.15}$$

Of course, the algorithmic description (4.15) defines a sequence of functions, where each one is a rational composition. This sequence starts with s_1 and ends with s_T, we denote it by

$$s_1, \ldots, s_T. \tag{4.16}$$

Now we work bottom-up and glue parts together. We set $f_i(x) = x_i$ for $i = 1, \ldots, n$ and $f_{n+1}(x) = T$. The entries m_{jk} of m are known explicitly. We combine the characterizing sequences of all the m_{jk} in one sequence

$$f_1, \ldots, f_n, \ldots, m_{11}, \ldots, m_{TT}. \tag{4.17}$$

Then we continue with (4.14) and (4.15) and obtain

$$f_1, \ldots, f_n, \ldots, m_{11}, \ldots, m_{TT}, l_{11}, \ldots, l_{TT}, s_1, \ldots, s_T. \tag{4.18}$$

From the sequence (4.18) we proceed straightforward to d according to (4.12) and to f according to (4.11). The necessary rational compositions and library compositions are obvious, we denote these by $r_1, \ldots, r_w$. The resulting sequence

$$f_1, \ldots, f_n, \ldots, m_{11}, \ldots, m_{TT}, l_{11}, \ldots, l_{TT}, s_1, \ldots, s_T, r_1, \ldots, r_w. \tag{4.19}$$

is a characterizing sequence for our objective function f.

Recall that the sequence (4.19) merely is an algorithmic description for computing $f(x)$. But if we choose a point x and use the sequence (4.19) together with Algorithm 1, we obtain the triple $f(x)$, $f_G(x)$, $f_H(x)$.

5. CONCLUSION

Automatic Differentiation is a powerful means for computing derivatives of fairly general functions, including all rational functions. Even if some function f involves implicitly defined quantities, such as the determinant or the inverse of a matrix, the entries of which are functions of several variables, gradient and Hessian matrix of f at given points can be computed effectively.

We discussed the exact maximum-likelihood estimation for linear regression models with stationary ARMA(p,q) residuals. The ML-estimator is the solution of a nonlinear optimization problem, therefore we proposed to use well-established

optimization methods together with Automatic Differentiation. First numerical results are promising. We are planning to do more numerical experiments, especially in a parallel computing environment.

In a forthcoming paper we shall present a detailed implementation on a transputer-based machine and numerical results as well.

REFERENCES

J. E. Dennis jr and R. B. Schnabel (1983), *Numerical Methods for Nonlinear Equations and Unconstrained Optimization*, Prentice-Hall Inc., Englewood Cliffs, New Jersey.

H. Fischer (1987), *Automatic differentiation: How to compute the Hessian matrix*. Report No. 26, DFG-Schwerpunkt: Anwendungsbezogene Optimierung und Steureung, Technische Universität München.

H. Fischer (1988), *Automatic differentiation: Fast method to compute the quadratic form of Hessian matrix and given vector*, Facta Universitatis (Niš), Ser. Mat. Inf. 3, 51–59.

H. Fischer (1989a), *Fast method to compute the scalar product of gradient and given vector*, Computing 41, 261–265.

H. Fischer (1989b), *Automatic differentiation of characterizing sequences*, J. Computational and Applied Mathematics 28, 181–185.

R. Fletcher (1983), *Practical Methods of Optimization, Vol. 1, Unconstrained Optimization*. John Wiley & Sons, New York.

P. E. Gill, W. Murray and M. H. Wright (1981), *Practical Optimization*, Academic Press, London.

A. Griewank and G. F. Corliss (Eds.) (1991), *Automatic Differentiation of Algorithms*, SIAM, Philadelphia.

A. C. Harvey and G. D. A. Phillips (1979), *Maximum likelihood estimation of regression models with autoregressive — moving average disturbances*, Biometrika 66, 49–58.

M. Pagano (1973), *When is an autoregressive scheme stationary*, Communications in Statistics 1, 533–544.

L. B. Rall (1980), *Applications of software for automatic differentiation in numerical computation*, Computing, Supplementum 2, 141–156.

L. B. Rall (1981), *Automatic Differentiation: Techniques and Applications*, Lecture Notes in Computer Science 120, Springer-Verlag, Berlin.

L. B. Rall (1984), *Differentiation in PASCAL-SC: type GRADIENT*, ACM Trans. Math. Software 10, 161–184.

L. B. Rall (1987), *Optimal implementation of differentiation arithmetic*, In: *Computer Arithmetic, Scientific Computation and Programming Languages*, ed. by E. Kaucher, U. Kulisch, Ch. Ullrich, Teubner-Verlag, Stuttgart, 287–295.

K. Ritter (1982), *Numerical methods for nonlinear programming problems*, in: *Modern Applied Mathematics — Optimization and Operations Research*, ed. by B. Korte, North-Holland Publishing Company, Amsterdam, 227–264.

J. W. Sawyer jr (1984), *First partial differentiation by computer with an application to categorical data analysis*, The American Statistician 38, 300–308.

W. C. Thacker (1990), *Large least-squares problems and the need for automating the generation of adjoint codes*, Lectures in Applied Mathematics 26, American Math. Society, 645–677.