International Institute for Applied Systems Analysis
IIASA www.iiasa.ac.at

Automatic Differentiation and Uncertainty Analysis

Huiskes, M.

IIASA Interim Report
August 1998

IIASA

International Institute for Applied Systems Analysis • A-2361 Laxenburg • Austria
Tel: +43 2236 807 • Fax: +43 2236 71313 • E-mail: info@iiasa.ac.at • Web: www.iiasa.ac.at

INTERIM REPORT IR-98-083 / August 1998

Automatic Differentiation and Uncertainty Analysis

Mark Huiskes(huiskes@iiasa.ac.at)

Approved by
Arkadii Kryazhimskii (kryazhim@iiasa.ac.at, kryazhim@mi.ras.ru)
Senior Research Scholar, *Dynamic Systems Project*

Abstract

This paper aims to give an overview of the possibilities for using automatic differentiation for uncertainty analysis. It presents an introduction to the general theory of automatic differentiation. Following this an overview of sensitivity analysis and nonlinear regression is given to provide the reader with a clear understanding of both general concepts and their relation to automatic differentiation.

Special attention is paid to the effect of model nonlinearity on the quality of the obtained estimates and it is investigated how automatic differentiation can be used to improve the estimates. Further the new concept of standard error sensitivity is introduced and formulas for efficient computation are derived.

Finally the ***Oak*** system is discussed. This system is an implementation of the theory discussed in this paper using the ADOL-C library for automatic differentiation. To demonstrate the possibilities of this system several models used at the IIASA Sustainable Boreal Forests Project have been investigated.

About the Author

This paper was written at the International Institute for Applied Systems Analysis, Laxenburg, Austria, where the author participated in the Young Scientists Summer Program 1998. Mark Huiskes is a Ph.D. student at Wageningen Agricultural University at the department of Mathematics, Wageningen, The Netherlands.

Acknowledgements

I would like to thank Alexander Tarasiev and Arkadii Kryazhimskii for their advice and encouragement during my stay at the Dynamic Systems project at IIASA. Special thanks go to Anatoli Shvidenko of the Sustainable Boreal Forests Project who provided me with several models and data sets.

I would also like to thank my fellow YSSP students and the IIASA staff members in general for giving me a great summer.

M.J. Huiskes
Wageningen Agricultural University, department of Mathematics

Contents

Automatic Differentiation and Uncertainty Analysis

Mark Huiskes(huiskes@iiasa.ac.at)

1 Introduction

In this paper we investigate how automatic differentiation can be used for uncertainty analysis. Automatic differentiation, as it is now known, consists of a collection of techniques to obtain derivatives of functions in the form of computer code. These techniques produce, by varying means depending on their nature, an exact representation of the derivative. It differs however from symbolic differentiation in the fact that the result is not an analytic expression, but rather a sequence of elementary computing operations. In the next section we present an overview of the various forms of automatic differentiation and explain the general theory behind it.

In the second section we turn to sensitivity analysis. We investigate how automatic differentiation can be used to obtain sensitivity information of algorithms in the form of computer code. We consider how uncertainty in independent variables is propagated through a model to the uncertainty in the dependent variables. We also present the new concept of error sensitivity and expressions for its computation. As an illustration of how this theory is used in practice we consider a model that is used to estimate phytomass of Siberian forests.

Then, in section 4, we move on to the estimation of parameters in nonlinear models and its associated uncertainty analysis. We show how automatic differentiation is useful both for the actual parameter estimation itself and for the evaluation and extension of the reliability of the obtained estimates.

In section 5 we give a short description of ***Oak***, a system for performing uncertainty analysis by means of automatic differentiation. It consists of a graphical user interface connected to routines for optimization, nonlinear regression and sensitivity analysis.

At the end of the paper we have included a bibliography on automatic differentiation. This bibliography is divided into general theory, applications and implementations of automatic differentiation. It also contains literature on so-called adjoint modelling, which is closely related to automatic differentiation.

2 Automatic differentiation

The problem of the computation of first and higher order derivatives of functions with m components and n independent variables arises in various contexts, e.g. that of optimization, nonlinear equation solving, bifurcation studies, and as discussed in this paper, also in uncertainty analysis.

The most widely used method to obtain derivatives is that of approximation by finite differences. Another possibility is by means of symbolic differentiation. It has been shown

however that, for general functions, these methods are less efficient than automatic differentiation. See [Gri89] for a discussion. Automatic differentiation can be applied to large and complicated functions, is computationally more efficient, and yields exact results.

The collection of techniques referred to as automatic differentiation (AD) share the common property of operating on functions that are presented in the form of computer code. Such functions consist of a sequence of elementary arithmetic operations, e.g. addition, subtraction, multiplication and division, and standard univariate functions, e.g. the exponential and the sine function.

We use the following notation, as introduced in [Gri89], to denote a *computational decomposition algorithm* of a function $R : X \subset \mathbb{R}^n \to \mathbb{R}$, $y = R(x)$

$$\begin{aligned} &\text{For } i = n+1, n+2, \ldots, N \\ &\qquad x_i = f_i <x_j>_{j \in \mathcal{J}_i} \\ &y = x_N \end{aligned}$$

where

$$\mathcal{J}_i \subset \{1, 2, \ldots, i-1\} \quad \text{for} \quad i = n+1, n+2, \ldots, N.$$

The n independent variables, i.e. the variables on which R depends, are labelled x_1 through x_n. The algorithm consists of a sequence of computations of intermediate variables x_i by means of the functions $f_i, i = n+1, \ldots, N$. Each f_i may depend on variables, independent or intermediate, which have an index lower than i. The indices of the variables on which f_i depends are given by the set $\mathcal{J}_i$. The final quantity that is computed must be the function value $y = R(x) = x_N$, called the *dependent* variable. In [JM88] it is shown that almost all functions of practical interest can be represented as a composition of this form.

In practice it is not necessary to write the computer code explicitly in the form of such a representation. Current AD packages are able to transform code as it is usually written, e.g. including assignment statements consisting of a composition of elementary functions, and including for-loops and if-statements.

The computational decomposition can be visualized by means of a computational graph. See Fig. 1 for a computational graph of the function $y = \cos(x_1 + x_2) \cdot \exp(3x_3 + 2)$. Each vertex corresponds to a variable. There is a connection between x_i and x_j (for $i > j$)

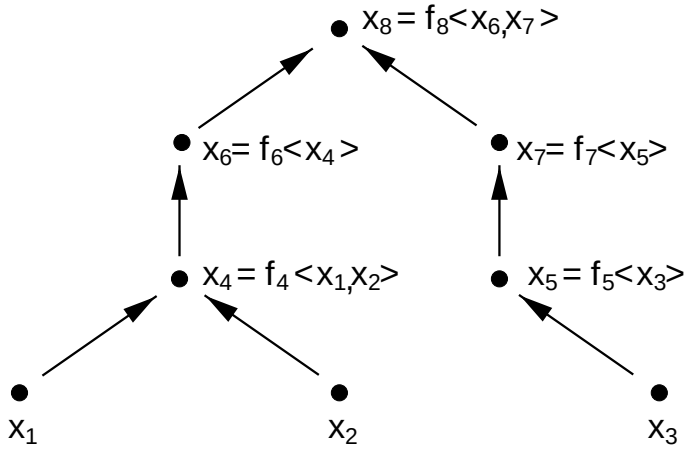

Figure 1: Computational graph of $y = \cos(x_1 + x_2) \cdot \exp(3x_3 + 2)$, where $f_4 <x_1, x_2>= x_1 + x_2$, $f_5 <x_3>= 3x_3 + 2$, $f_6 <x_4>= \cos(x_4)$, $f_7 <x_5>= \exp(x_5)$ and $f_8 <x_6, x_7>= x_6 x_7$

if $j \in \mathcal{J}_i$, i.e. if the function f_i depends on x_j. Since each x_i may only depend on variables

with smaller index the computational graph is acyclic.

Notice that both the computational decomposition and the corresponding computational graph are not unique.

Depending on how the derivative is computed in relation to the computational graph, we can discern the two main modes of automatic differentiation. These are the *forward* and *reverse* modes of automatic differentiation. For a detailed discussion of the relation between the computational graph and automatic differentiation, see [Iri84].

In the remaining part of this section, we first discuss the forward and reverse modes of automatic differentiation. Then, in section 2.3, we consider the computational complexity of these two modes. In section 2.4, we discuss the automatic evaluation of higher order derivatives. Finally, in section 2.5 we discuss the main implementation types of automatic differentiation.

2.1 The forward mode of automatic differentiation

The forward mode of AD is based on a straightforward application of the chain rule. With each vertex of the computational graph, i.e for each variable, we associate a derivative with respect to the n independent variables:

$$Dx_i = (\frac{\partial x_i}{\partial x_1}, \ldots, \frac{\partial x_i}{\partial x_n}), \quad \text{for} \quad i = 1, \ldots, N. \tag{1}$$

We have

$$Dx_i = e_i \quad \text{for} \quad i = 1, \ldots, n, \tag{2}$$

where e_i is the i-th Cartesian basis vector of $\mathbb{R}^n$. The derivatives associated with the other variables can be computed from previously computed derivatives by using

$$(Dx_i)_k = \frac{\partial x_i}{\partial x_k} = \sum_{j \in \mathcal{J}_i} \frac{\partial f_i}{\partial x_j} \frac{\partial x_j}{\partial x_k} = \sum_{j \in \mathcal{J}_i} \frac{\partial f_i}{\partial x_j} (Dx_j)_k. \tag{3}$$

We start from the derivatives associated with the independent variables and proceed through the computational tree until finally the derivative associated with the dependent variable is computed. This explains the term forward mode. The process can be summarized by the following algorithm for the simultaneous evaluation of a function R and its derivative DR

For $i = 1, 2, \ldots, n$
  $Dx_i = e_i$
For $i = n+1, n+2, \ldots, N$
  $x_i = f_i < x_j >_{j \in \mathcal{J}_i}$
  For $k = 1, 2, \ldots, n$
    $(Dx_i)_k = \sum_{j \in \mathcal{J}_i} \frac{\partial f_i}{\partial x_j} (Dx_j)_k$
$R = x_N$
$DR = Dx_N$

Algorithm 2.1: Forward mode of automatic differentiation.

2.2 The reverse mode of automatic differentiation

As will be shown later it is possible to obtain the derivative of a scalar function much more efficiently by proceeding through the computational graph in the reverse direction. To this end we now associate with each variable a scalar derivative

$$\bar{x}_i = \frac{\partial x_N}{\partial x_i} \quad \text{for} \quad i = 1, \ldots, N, \tag{4}$$

which is called the *adjoint* variable associated with x_i. Notice that we have $\bar{x}_N = 1$ and

$$\bar{x}_i = \frac{\partial R}{\partial x_i} \quad \text{for} \quad i = 1, \ldots, n. \tag{5}$$

This means that the derivative of $y = R(x)$ is equal to the adjoint variables associated to the independent variables.

Let

$$\mathcal{I}_j = \{i | j \in \mathcal{J}_i\} \quad \text{for} \quad j = 1, \ldots, N. \tag{6}$$

$\mathcal{I}_j$ is the set of indices of those variables that depend directly on x_j. For all $i \in \mathcal{I}_j$ we have $i > j$. Now consider x_N as a function of x_j, i.e.

$$x_N = x_N <x_i(x_j)>_{i \in \mathcal{I}_j}; \tag{7}$$

by the chain rule we then find

$$\bar{x}_j = \frac{\partial x_N}{\partial x_j} = \sum_{i \in \mathcal{I}_j} \frac{\partial x_N}{\partial x_i} \frac{\partial f_i}{\partial x_j} = \sum_{i \in \mathcal{I}_j} \frac{\partial f_i}{\partial x_j} \bar{x}_i. \tag{8}$$

It follows that the adjoint variable corresponding to x_j can be computed once all $\bar{x}_i$ with $i > j$ are known. The following algorithm for reverse automatic differentiation is based on this structure.

For $i = n+1, n+2, \ldots, N$
  $x_i = f_i <x_j>_{j \in \mathcal{J}_i}$
  $\bar{x}_i = 0$
$R = x_N$
$x_N = 1$
$<\bar{x}_i>_{i=1}^n = 0$
For $i = N, N-1, \ldots, n+1$
  $\bar{x}_j = \bar{x}_j + \frac{\partial f_i}{\partial x_j} \bar{x}_i$ for all $j \in \mathcal{J}_i$
$Dy = <\bar{x}_i>_{i=1}^n$

Algorithm 2.2: Reverse mode of automatic differentiation.

The functions f_i are visited in reverse order. For each $j \in \mathcal{J}_i$ it can be seen from (8) that the expression for $\bar{x}_j$ contains a term $\frac{\partial f_i}{\partial x_j} \bar{x}_i$. Notice that when the function f_i is visited, $\bar{x}_i$ has already been calculated. Other orders of visitation are also possible provided that this same requirement is met. (These could be favourable in particular implementations, for instance to reduce cache misses.)

2.3 The computational cost of automatic differentiation

We now consider the computational cost of the algorithms discussed above. First we assume that the cost of evaluating the response function R is equal to the sum of the costs of evaluating the functions of the computational decomposition, i.e.

$$\text{cost}(R) = \sum_{i=n+1}^{N} \text{cost}(f_i) \tag{9}$$

For the forward mode of automatic differentiation, we then have from Algorithm 2.1

$$\text{cost}(R, DR) = \sum_{i=n+1}^{N} (\text{cost}(f_i, Df_i) + nn_i \,\text{cost}(\text{mult} + \text{add})) \tag{10}$$

The *cost ratio* is defined by

$$\frac{\text{cost}(R, DR)}{\text{cost}(R)}, \tag{11}$$

i.e. the ratio of the cost of evaluating both R and its first order derivative DR and the cost of evaluating R itself.

If we assume that there exists a constant c for which the cost of evaluating a function f_i can be bounded for every i by the cost of cn_i arithmetic operations, then it follows that

$$\frac{\text{cost}(R, DR)}{\text{cost}(R)} \geq 1 + \frac{n}{c} \tag{12}$$

This linear growth of the computational cost of a derivative of forward automatic differentiation is of the same order as the cost for evaluating finite differences.

In [Gri89] it is shown that the cost ratio for reverse automatic differentiation is independent of the number of independent variables. From Algorithm 2.2, we find

$$\text{cost}(R, DR) = \sum_{i=n+1}^{N} (\text{cost}(f_i, Df_i) + n_i \,\text{cost}(\text{mult} + \text{add})) \tag{13}$$

and define

$$\omega = \max_{n<i\leq N} \frac{\text{cost}(f_i, Df_i) + n_i \,\text{cost}(\text{mult} + \text{add})}{\text{cost}(f_i)}. \tag{14}$$

We now find that the cost ratio satisfies

$$\frac{\text{cost}(R, DR)}{\text{cost}(R)} = \frac{\sum_{i=n+1}^{N} (\text{cost}(f_i, Df_i) + n_i \,\text{cost}(\text{mult} + \text{add}))}{\sum_{i=n+1}^{N} \text{cost}(f_i)} \leq \omega. \tag{15}$$

This means that the cost ratio is bounded by a ratio for one of the f_i. If we now suppose that the f_i are restricted to the elementary arithmetic operations and standard univariate functions, we can compute the worst case ratio of these functions. The highest ratio is a limit for the cost ratio. It is shown in [Gri89] that for the sine and cosine function the ratio lies just above two and for most other system functions is close to one. Multiplication turns out to be the most expensive arithmetic operation with a cost ratio just under 5. It thus follows that the cost of evaluating a function and its derivative by means of reverse

automatic differentiation is at most 5 times the cost of just evaluating the function itself. In practice one can expect this ratio to be even smaller.

For a thorough discussion of the costs of computing derivatives of functions with more than one component we refer to [Gri93]. The paper also goes into the memory requirements of the algorithms. In general it can be concluded that reverse automatic differentiation is preferable if the number of independent variables is higher or not far less than the number of dependent variables. The characteristics of the forward method become more favourable if there are far more dependent than independent variables. Mixed approaches are also possible, as for instance described in [Iri84].

In [Gri93] bounds are derived for the memory requirements of the forward and reverse method of automatic differentiation. The memory requirement of the forward mode is approximately linear in the number of independent variables compared to the memory requirement of the original code. Since for the reverse mode of automatic differentiation the entire structure computational graph must be stored, the memory requirement for the reverse mode is potentially much larger. In a standard implementation it may be proportional to the run time, T, of the original evaluation code. However in [Gri91] a scheme is described which limits the required memory to a fixed multiple of $\log(T)$.

2.4 Higher order derivatives by means of automatic differentiation

The forward mode can be readily extended to obtain higher order derivatives. The Hessian D^2x_N for instance can be obtained by updating Hessian matrices D^2x_i of all the variables with respect to the independent variables by using

$$D^2x_i = \sum_{j\in\mathcal{J}_i}\left(\frac{\partial f_i}{\partial x_j}D^2x_j + \sum_{k\in\mathcal{J}_i} Dx_j\frac{\partial^2 f_i}{\partial x_j\partial x_k}(Dx_k)^T\right) \tag{16}$$

where $Dx_i = e_i$ and $D^2x_i = 0$ for $i = 1,\ldots,n$. Similar chain rules are available for third and higher order derivatives. The evaluation of the Hessian matrix in the forward mode will usually be roughly n^2 times as expensive as evaluation of the function itself. See [Gri89].

In [BCG$^+$93] a method is described to extract partial derivatives of arbitrary order by interpolating a number of univariate Taylor expansions. Further in [Chr91] it is shown that univariate Taylor series can be propagated by a method very similar to reverse automatic differentiation. Since it is possible to obtain each partial derivative separately, the combination of these methods is ideally suited to exploit sparsity patterns of higher order derivatives.

The methods described above have been implemented in the ADOL-C package for automatic differentiation, see [GJSTar].

2.5 Implementations of automatic differentiation

In [Jue91] 29 software packages for automatic differentiation are compared which vary greatly in their possibilities. A first criterion for comparison is whether either forward or reverse mode is used, or both modes are available. In the paper a further five categories are discerned.

Somewhat more roughly we can observe two main implementation types. The first type makes use of a precompiler to obtain derivative code. The precompiler is presented with code for a function in some computer language such as Fortran. It generates code in the same language for the derivative and possibly higher derivatives of the function. Software

packages that use this approach are for instance JAKEF ([Hil82]), GRESS ([Hor91]) and PADRE2 ([Kub91]). These precompilers use Fortran as their source language.

For the second type no separate precompiler is required. Code is compiled by a regular compiler for a modern programming language, such as ADA or C++. The language constructs of operator overloading and polymorphic functions are required for this approach. These and a special data type for floating point numbers allow the computional graph to be constructed implicitly while evaluating the original function. Once the structure of the computational graph has been recorded, one can use first and higher order derivative library functions which make use of this structure. Two packages that use this approach are ADOL-C ([GJSTar]) and BC1([Chr92a]).

In Program 2.1 an example of C++ code that uses the ADOL-C library is shown. In this example the code for the calculation of a simple function is extended by some additional ADOL-C notation to be able to use the automatic differentiation routines of the library. The computation takes place between the statements `trace_on` and `trace_off`. The independent and dependent variables are indicated by means of the operators `<<=` and `>>=`, respectively. All variables that depend directly or indirectly on the independent variables are called *active* variables and have to be of type `adouble`.

The structure of the computational graph, called a tape, is stored sequentially in main memory and is automatically paged to disk when necessary. During subsequent derivative evaluations, tapes are always accessed sequentially, so they can be paged in and out to disk without significant run time penalties. After construction of the tape, the gradient function of the library may be called to evaluate the gradient of the dependent variable with respect to the indendent variables using the variable `tag` to indicate the required tape. The gradient may also be computed for different independent variable values using the same tape, provided that no new program branches are being taken due conditions on `adouble` values.

Similar functions are available for the evaluation of Jacobians, Hessians and higher order derivatives.

An important new area of application of automatic differentiation is that of obtaining accurate roundoff error estimates. See [Iri91] for a review. Automatic differentiation can also play an important part in the development of interval arithmetics. See [Kul96] for a discussion on how the arithmetic capability and reportoire of computers should be expanded to make optimal use of the recent advances.

3 Sensitivity analysis

3.1 Single response

Consider a scalar system response $R(x)$ depending on n independent variables $x = (x_1, \ldots, x_n)$. Higher dimensional responses are treated following this case. The sensitivity coefficients of the response are defined by

$$s_i(x) = \frac{\partial R}{\partial x_i}(x), \tag{17}$$

or scaled to nondimensional quantities

$$S_i(x) = \frac{\partial R}{\partial x_i}(x)\frac{x_i}{R}. \tag{18}$$

```
#include <adouble.h>
#include <adutils.h>

int main()
{
    const int n=10;
    double* x_in=new double[n];
    adouble* x=new adouble[n];
    double y_out;
    double* grad=new double[n];

    \\ Set values of independent variables.
    for (int i=0;i<n;i++) x_in=1.;

    \\ Start tracing of computational graph.
    short int tag=1;
    trace_on(tag);

    \\ Indicate independent variables.
    for (int i=0;i<n;i++) x[i]<<=x_in;

    \\ Perform computations using active variables.
    adouble int_res=(x[1]+x[2])*x[3]*cos(x[4]);
    adouble end_res=exp(int_res)*x[1]/x[5];

    \\ Indicate dependent variable(s).
    end_res>>=y_out;

    \\ End trace of computational graph.
    trace_off(tag);

    \\ Use function gradient from ADOL-C library.
    gradient(tag,n,x,grad);

    delete[] grad; delete[] x; delete[] x_in;
    return 0;
}
```

Program 2.1: Example of C++ code using the ADOL-C library.

Second order sensitivities are defined by

$$q_{ij}(x) = \frac{\partial^2 R}{\partial x_i \partial x_j}(x) \tag{19}$$

and

$$Q_{ij}(x) = \frac{\partial^2 R}{\partial x_i \partial x_j}(x)\frac{x_i x_j}{R}. \tag{20}$$

Sensitivities of order higher than two can be defined analogously.

If the independent variables x are perturbed by a vector δx, the sensitivity coefficients can be used to approximate the resulting change δR in the response. We have, using a Taylor expansion of order two, that

$$\delta R = R(x+\delta x) - R(x) = \sum_{i=1}^{n} s_i(x)\delta x_i + \frac{1}{2}\sum_{i,j=1}^{n} q_{ij}(x)\delta x_i \delta x_j + \mathcal{O}(|\delta x|^3). \tag{21}$$

Scaled sensitivity coefficients are used in dealing with relative perturbations, as can be seen from

$$\frac{\delta R}{R} = \sum_{i=1}^{n} S_i(x)\left(\frac{\delta x_i}{x_i}\right) + \frac{1}{2}\sum_{i,j=1}^{n} Q_{ij}(x)\left(\frac{\delta x_i}{x_i}\right)\left(\frac{\delta x_j}{x_j}\right) + \mathcal{O}(|\delta x|^3). \tag{22}$$

We will now assume that the independent variables are stochastic, thereby also making the response stochastic. We consider how uncertainty information about the independent variables can be transformed into uncertainty information about the response variable using the sensitivities. To this end we use a Taylor series expansion of the response around the expectation value of the independent variables:

$$\begin{aligned} R(x) &= R(\mathbf{E}[x]) + \sum_{i_1=1}^{n}\left(\frac{\partial R}{\partial x_{i_1}}\right)_{\mathbf{E}[x]}\delta x_{i_1} + \frac{1}{2}\sum_{i_1,i_2=1}^{n}\left(\frac{\partial^2 R}{\partial x_{i_1}\partial x_{i_2}}\right)_{\mathbf{E}[x]}\delta x_{i_1}\delta x_{i_2} \\ &+\frac{1}{3!}\sum_{i_1,i_2,i_3=1}^{n}\left(\frac{\partial^3 R}{\partial x_{i_1}\partial x_{i_2}\partial x_{i_3}}\right)_{\mathbf{E}[x]}\delta x_{i_1}\delta x_{i_2}\delta x_{i_3} + \ldots \\ &+\frac{1}{N!}\sum_{i_1,\ldots,i_N=1}^{n}\left(\frac{\partial^N R}{\partial x_{i_1}\ldots\partial x_{i_N}}\right)_{\mathbf{E}[x]}\delta x_{i_1}\delta x_{i_2}\ldots\delta x_{i_N} + \ldots \end{aligned} \tag{23}$$

in which $\delta x_i = x_i - \mathbf{E}[x_i]$.

The expansion is used to construct an Nth order approximation of the response variance

$$\mathrm{var}(R) = \mathbf{E}[(R - \mathbf{E}[R])^2], \tag{24}$$

in which, since $\mathbf{E}[\delta x_i] = 0$,

$$\begin{aligned} \mathbf{E}[R] &= R(\mathbf{E}[x]) + \frac{1}{2}\sum_{i_1,i_2=1}^{n}\left(\frac{\partial^2 R}{\partial x_{i_1}\partial x_{i_2}}\right)_{\mathbf{E}[x]}\mathbf{E}[\delta x_{i_1}\delta x_{i_2}] + \\ &\frac{1}{3!}\sum_{i_1,i_2,i_3=1}^{n}\left(\frac{\partial^3 R}{\partial x_{i_1}\partial x_{i_2}\partial x_{i_3}}\right)_{\mathbf{E}[x]}\mathbf{E}[\delta x_{i_1}\delta x_{i_2}\delta x_{i_3}] + \ldots + \\ &\frac{1}{N!}\sum_{i_1,\ldots,i_N=1}^{n}\left(\frac{\partial^N R}{\partial x_{i_1}\ldots\partial x_{i_N}}\right)_{\mathbf{E}[x]}\mathbf{E}[\delta x_{i_1}\ldots\delta x_{i_N}] + \ldots. \end{aligned} \tag{25}$$

Since we have

$$\mathbf{E}(\delta x_{i_1} \dots \delta x_{i_k} - \mathbf{E}[\delta x_{i_1} \dots x_{i_k}])(\delta x_{j_1} \dots \delta x_{j_l} - \mathbf{E}[\delta x_{j_1} \dots x_{j_l}]) =$$
$$(\mathbf{E}[\delta x_{i_1} \dots \delta x_{i_k} \delta x_{j_1} \dots \delta x_{j_l}] - \mathbf{E}[\delta x_{i_1} \dots \delta x_{i_k}]\mathbf{E}[\delta x_{j_1} \dots \delta x_{j_l}]), \quad (26)$$

where $i_1, \dots, i_k \in \{1, 2, \dots, n\}$ and $j_1, \dots, j_l \in \{1, 2, \dots, n\}$, it follows that the Nth order approximation of the response variance $\mathrm{var}_N(R)$ is given by

$$\mathrm{var}_N(R) = \sum_{k,l=1}^{N} \frac{1}{k!l!} \sum_{\substack{i_1,\dots,i_n=1 \\ j_1,\dots,j_n=1}}^{n} \left(\frac{\partial^k R}{\partial x_{i_1} \dots \partial x_{i_k}} \frac{\partial^l R}{\partial x_{j_1} \dots \partial x_{j_l}} \right)_{\mathbf{E}[x]}$$
$$(\mathbf{E}[\delta x_{i_1} \dots \delta x_{i_k} \delta x_{j_1} \dots \delta x_{j_l}] - \mathbf{E}[\delta x_{i_1} \dots \delta x_{i_k}]\mathbf{E}[\delta x_{j_1} \dots \delta x_{j_l}]) \quad (27)$$

Notice that some terms in this expression are zero, since $\mathbf{E}[\delta x] = 0$. For $N = 1$ we have

$$\mathrm{var}_1(R) = \sum_{i,j=1}^{n} \left(\frac{\partial R}{\partial x_i} \frac{\partial R}{\partial x_j} \right)_{\mathbf{E}[x]} \mathbf{E}[\delta x_i \delta x_j] \quad (28)$$

By noting that the $\frac{\partial R}{\partial x_i}$ are the first order sensitivities s_i and that $\mathbf{E}[\delta x_i \delta x_j]$ is the covariance $\mathrm{cov}(x_i, x_j)$ between x_i and x_j, we get

$$\mathrm{var}_1(R) = \sum_{i,j=1}^{n} s_i s_j \mathrm{cov}(x_i, x_j) = s\, \Sigma\, s^T \quad (29)$$

where $s = (s_1, \dots, s_n)$ is the sensitivity (row)vector and Σ the covariance matrix of the independent variables. This formula is known as the *sandwich rule*.

The second order approximation ($N = 2$) of the response variance (27) is given by

$$\mathrm{var}_2(R) = \sum_{i,j=1}^{n} \left(\frac{\partial R}{\partial x_i} \frac{\partial R}{\partial x_j} \right)_{\mathbf{E}[x]} \mathbf{E}[\delta x_i \delta x_j] +$$
$$\sum_{\substack{i_1,\, i_2 = 1 \\ j\ = 1}}^{n} \left(\frac{\partial^2 R}{\partial x_{i_1} \partial x_{i_2}} \frac{\partial R}{\partial x_j} \right)_{\mathbf{E}[x]} \mathbf{E}[\delta x_{i_1} \delta x_{i_2} \delta x_j] +$$
$$\frac{1}{4} \sum_{\substack{i_1,\, i_2 = 1 \\ j_1,\, j_2 = 1}}^{n} \left(\frac{\partial^2 R}{\partial x_{i_1} \partial x_{i_2}} \frac{\partial^2 R}{\partial x_{j_1} \partial x_{j_2}} \right)_{\mathbf{E}[x]} (\mathbf{E}[\delta x_{i_1} \delta x_{i_2} \delta x_{j_1} \delta x_{j_2}] - \mathbf{E}[\delta x_{i_1} \delta x_{i_2}]\mathbf{E}[\delta x_{j_1} \delta x_{j_2}]) \quad (30)$$

Notice that for a second order analysis third and fourth order central moments are required. If we assume however that the independent variables have a multivariate normal distribution, these higher order moments can be calculated from the second order moments, i.e. from the covariances, as is shown in [Ron88]. The joint density function of the independent variables for this case is given by

$$f(x) = \frac{1}{\sqrt{(2\pi)^n |\Sigma|}} \exp(-\frac{1}{2}(x - \mu)^T \Sigma^{-1} (x - \mu)) \quad (31)$$

where $\mu = \mathbf{E}[x]$ is the expectation vector of the independent variables and Σ the covariance matrix. We use

$$\Sigma_{ij} = \rho_{ij} \sigma_i \sigma_j \quad (32)$$

with σ_i the standard deviation of x_i and ρ_{ij} the correlation coefficient between x_i and x_j. Since the multivariate normal distribution is symmetric about its mean, all odd central moments are zero. One can derive from the characteristic function for the multivariate normal distribution

$$\phi(z) = \exp(-\frac{1}{2} z^T \Sigma z) \tag{33}$$

that for n even we have

$$\mathbf{E}[\delta x_{i_1} \dots \delta x_{i_n}] = \sum_{i_{k_l} < j_{k_l}} \rho_{i_{k_1} j_{k_1}} \rho_{i_{k_2} j_{k_2}} \dots \rho_{i_{k_{n/2}} j_{k_{n/2}}} \times \sigma_{i_1} \sigma_{i_2} \dots \sigma_{i_n} \tag{34}$$

Under the multivariate normal assumption, (30) then reduces to

$$\begin{aligned} \mathrm{var}_2(R) &= \sum_{i,j=1}^{n} s_i s_j \mathrm{cov}(x_i, x_j) + \\ & \frac{1}{4} \sum_{i,j,k,l=1}^{n} q_{ij} q_{kl} (\mathrm{cov}(x_i, x_k)\mathrm{cov}(x_j, x_l) + \mathrm{cov}(x_i, x_l)\mathrm{cov}(x_j, x_k)) \end{aligned} \tag{35}$$

3.2 Multivariate response

We now consider the case that the response $R(x)$ has several components:

$$R(x) = (R_1(x), \dots, R_m(x)). \tag{36}$$

Sensitivities are now defined for each component as in (17), (18), (19) and (20). Also the expressions for the approximations of the variance remain valid for each component separately. What remains to be investigated is the covariance between two components

$$\mathrm{cov}(R_p, R_q) = \mathbf{E}[(R_p - \mathbf{E}[R_p])(R_q - \mathbf{E}[R_q])] \tag{37}$$

Using the same approach as for the variance of a single response, we obtain for the Nth order approximation

$$\begin{aligned} \mathrm{cov}_N(R_p, R_q) &= \sum_{k,l=1}^{N} \frac{1}{k!l!} \sum_{\substack{i_1,\dots,i_n=1 \\ j_1,\dots,j_n=1}}^{n} \left(\frac{\partial^k R_p}{\partial x_{i_1} \dots \partial x_{i_k}} \frac{\partial^l R_q}{\partial x_{j_1} \dots \partial x_{j_l}} \right)_{\mathbf{E}[x]} \\ & (\mathbf{E}[\delta x_{i_1} \dots \delta x_{i_k} \delta x_{j_1} \dots \delta x_{j_l}] - \mathbf{E}[\delta x_{i_1} \dots \delta x_{i_k}] \mathbf{E}[\delta x_{j_1} \dots \delta x_{j_l}]) \end{aligned} \tag{38}$$

where again some terms are zero due to $\mathbf{E}[\delta x] = 0$. For $N = 1$ we have

$$\mathrm{cov}_1(R_p, R_q) = \sum_{i,j=1}^{n} s_i^{(p)} s_j^{(q)} \mathrm{cov}(x_i, x_j), \tag{39}$$

where $s_i^{(p)}$ denotes the sensitivity of the response component p with respect to the independent variable x_i. It follows that the covariance matrix of the response is given by

$$\mathrm{cov}_1(R) = S \Sigma S^T \tag{40}$$

in which $s_{ij} = s_j^{(i)}$. (Notice that S is equal to the derivative of the response with respect to the independent variables). The second order approximation of the covariance between

two response components is given by

$$\begin{aligned}\mathrm{cov}_2(R_p,R_q) &= \sum_{i,j=1}^{n}\left(\frac{\partial R_p}{\partial x_i}\frac{\partial R_q}{\partial x_j}\right)_{\mathbf{E}[x]}\mathbf{E}[\delta x_i\delta x_j]+\\ &\frac{1}{2}\sum_{\substack{i_1,\,i_2=1\\ j=1}}^{n}\left(\frac{\partial^2 R_p}{\partial x_{i_1}\partial x_{i_2}}\frac{\partial R_q}{\partial x_j}\right)_{\mathbf{E}[x]}\mathbf{E}[\delta x_{i_1}\delta x_{i_2}\delta x_j]+\\ &\frac{1}{2}\sum_{\substack{i=1\\ j_1,\,j_2=1}}^{n}\left(\frac{\partial R_p}{\partial x_i}\frac{\partial^2 R_q}{\partial x_{j_1}\partial x_{j_2}}\right)_{\mathbf{E}[x]}\mathbf{E}[\delta x_i\delta x_{j_1}\delta x_{j_2}]+\\ \frac{1}{4}\sum_{\substack{i_1,\,i_2=1\\ j_1,\,j_2=1}}^{n}&\left(\frac{\partial^2 R}{\partial x_{i_1}\partial x_{i_2}}\frac{\partial^2 R}{\partial x_{j_1}\partial x_{j_2}}\right)_{\mathbf{E}[x]}(\mathbf{E}[\delta x_{i_1}\delta x_{i_2}\delta x_{j_1}\delta x_{j_2}]-\mathbf{E}[\delta x_{i_1}\delta x_{i_2}]\mathbf{E}[\delta x_{j_1}\delta x_{j_2}]).\end{aligned}\tag{41}$$

Under the assumption that the independent variables have a multivariate normal distribution, we get using the same considerations as for (35)

$$\begin{aligned}\mathrm{cov}_2(R_p,R_q) &= \sum_{i,j=1}^{n}s_i^{(p)}s_j^{(q)}\mathrm{cov}(x_i,x_j)+\\ \frac{1}{4}\sum_{i,j,k,l=1}&q_{ij}^{(p)}q_{kl}^{(q)}(\mathrm{cov}(x_i,x_k)\mathrm{cov}(x_j,x_l)+\mathrm{cov}(x_i,x_l)\mathrm{cov}(x_j,x_k)).\end{aligned}\tag{42}$$

3.3 Standard error sensitivity

We now introduce a new type of sensitivity measure, which we shall refer to as *standard error sensitivity*. This sensitivity approximates the change in response standard error, given a change in the standard errors of the independent variables.

The standard errors are defined by

$$e_{R_p}=\frac{\sigma_{R_p}}{R_p},\quad p=1,\ldots,m\quad\text{and}\quad e_{x_i}=\frac{\sigma_{x_i}}{x_i},\quad i=1,\ldots,n.\tag{43}$$

We now define the (scaled) standard error sensitivity coefficients by

$$S_{e_i}^{p}=\frac{\partial e_{R_p}}{\partial e_{x_i}}\frac{e_{x_i}}{e_{R_p}}=\frac{\partial \sigma_{R_p}}{\partial \sigma_{x_i}}\frac{\sigma_{x_i}}{\sigma_{R_p}}.\tag{44}$$

Notice that just as for the response sensitivities of the previous sections these sensitivity coefficients depend on the independent variables (which are kept fixed during computation of the coefficients). The error sensitivity coefficients are used to estimate the effect δe_{R_p} of a change in the response standard errors δe_{x_i} caused by a change in the standard errors of the independent variables. This is achieved by means of the following Taylor expansion

$$\frac{\delta e_{R_p}}{e_{R_p}}=\sum_{i=1}^{n}S_{E_i}^{p}\left(\frac{\delta e_{x_i}}{e_{x_i}}\right)+\mathcal{O}(|\delta e|^2).\tag{45}$$

As in the case of response sensitivities these approximations can be improved by incorporating higher order sensitivities.

3.4 The role of automatic differentiation

Calculation of sensitivity coefficients amounts to computing a Taylor expansion of the response with respect to the independent variables. As explained in section 2 this can be done automatically and efficiently for a given piece of computer code representing the response using automatic differentiation.

Our AD uncertainty analysis tool using the automatic differentiation library ADOL-C (see [Gri89]) computes the sensitivity coefficients (as well as the scaled coefficients) for differentiable functions represented by C++ computer code. It offers the possibility to use these sensitivity coefficients to propogate uncertainties through the system by computing response covariances from covariance information of the independent variables and also allows computation of the standard error sensitivities.

Computation of the standard error sensitivities can be done by automatic differentiation of the covariance transformation code, but since the structure of the computation is rather simple it is more efficient to explicitly calculate the standard error sensitivities in terms of the response sensitivities.

We use the last equality of (44), i.e.

$$S_{e_i}^p = \frac{\partial \sigma_{R_p}}{\partial \sigma_{x_i}} \frac{\sigma_{x_i}}{\sigma_{R_p}}. \tag{46}$$

Using (39) and (32) we have as a first order approximation of $\sigma_{R_p}^2$

$$\sigma_{R_p}^2 = \sum_{i,j=1}^{n} s_i^{(p)} s_j^{(p)} \rho_{ij} \sigma_{x_i} \sigma_{x_j}. \tag{47}$$

By taking the derivative, we get

$$\frac{\partial \sigma_{R_p}}{\partial \sigma_{x_i}} = \frac{1}{2\sigma_{R_p}} \frac{\partial \sigma_{R_p}^2}{\partial \sigma_{x_i}} = \frac{1}{\sigma_{R_p}} \sum_{j=1}^{n} s_i^{(p)} s_j^{(p)} \rho_{ij} \sigma_j \tag{48}$$

Analogously we have as a second order approximation of $\sigma_{R_p}^2$

$$\sigma_{R_p}^2 = \sum_{i,j=1}^{n} s_i^{(p)} s_j^{(p)} \rho_{ij} \sigma_{x_i} \sigma_{x_j} + \sum_{i,j,k,l=1}^{n} q_{ij}^{(p)} q_{kl}^{(p)} (\rho_{ik}\rho_{jl} + \rho_{il}\rho_{jk}) \sigma_{x_i} \sigma_{x_j} \sigma_{x_k} \sigma_{x_l}, \tag{49}$$

giving the derivative

$$\frac{\partial \sigma_{R_p}}{\partial \sigma_{x_i}} = \frac{1}{\sigma_{R_p}} \left(\sum_{j=1}^{n} s_i^{(p)} s_j^{(p)} \rho_{ij} \sigma_j + 2 \sum_{j,k,l=1}^{n} q_{ij}^{(p)} q_{kl}^{(p)} (\rho_{ik}\rho_{jl} + \rho_{il}\rho_{jk}) \sigma_{x_j} \sigma_{x_k} \sigma_{x_l} \right). \tag{50}$$

Using this expression and (44) we can thus evaluate the standard error sensitivities in terms of the response sensitivities.

3.5 An example – phytomass (live biomass) of Siberian forests

In this section we consider a model as it is used in the Sustainable Boreal Forest Resources Project at IIASA to obtain estimates of the phytomass inventory of Siberian forests. The model basically transforms a database of information on the forest composition of so-called ecoregions into total phytomass estimates. The data for the ecoregions includes areas and growing stocks over dominant species, age, site indices and relative stocking.

In [SSN98a] and [SVN96] the details of the estimation process are described and it is shown how this model is applied to the assessment of carbon dynamics.

The stand phytomass is assumed to depend on the age A, the stock index SI and the relative stocking RS. Site indices are determined in Russia by average height at a certain age of stands. The relative stocking is defined as the ratio between the basal area of the investigated stand and the basal area of a fully stocked stand according to corresponding yield tables.

The total phytomass is divided into a number of groups, called fractions, where each of these fractions is assumed to have its own description in terms of the variables A, SI and GS. These dependencies have been estimated by means of nonlinear regression.

In this example we take pine as our species of consideration as it is generally found in mixed and deciduous forests or forest steppe. For pine five phytomass fractions are considered, cf. stem, bark, branches, needles and roots phytomass. The phytomass of each fraction is described by

$$M_{\mathrm{fr}} = R_{\mathrm{fr}} \cdot GS, \tag{51}$$

where the GS is (green) growing stock in m^3 and $\mathrm{R_{fr}}$ is a ratio in $\mathrm{Tg/m}^3$ expressing the relative density of the corresponding phytomass fraction. It is assumed that this ratio R_{fr} can be approximated by

$$R_{\mathrm{fr}} = c_0 SI^{c_1} A^{(c_2 + c_3 RS + c_4 RS^2)}, \tag{52}$$

where the coefficients c_0 through c_4 have been determined by regression as discussed in [SSN98b]. The coefficients that have been determined for the pine species under consideration are listed in Table 1.

Phytomass fraction	c_0	c_1	c_2	c_3	c_4
Stem	0.3172	0.0445	0.1338	-0.1824	0.0851
Bark	0.1335	0.7125	-0.4021	-0.1727	0.0458
Branches	0.2492	0.2122	-0.6300	0.5516	-0.3717
Needles	0.4146	0.7097	-0.8140	0.1107	-0.1257
Roots	0.6005	-0.0909	-1.0006	1.3113	-0.6323

Table 1: Regression coefficients in equation (52) for pine (mixed and deciduous forests and forest steppe)

It is further assumed that the growing stock is given by

$$GS = a_1(1 - \exp(-a_2 A))^{a_3}, \tag{53}$$

where each $a_i, i = 1, 2, 3$ is a quadratic polynomial in terms of the site index SI and the relative stocking RS, i.e.

$$a_i = \sum_{k_1=0}^{2} \sum_{k_2=0}^{2} a_{ik_1k_2} (SI)^{k_1} (RS)^{k_2}. \tag{54}$$

For this example we have (see [SVN96])

$$a_1 = 13.2 - 22.1 \cdot SI + 879 \cdot RS + 3.80 \cdot SI^2 - 121 \cdot SI \cdot RS + 93.6 \cdot RS^2 \tag{55}$$

$$a_2 = (2.92 - 0.25 \cdot SI - 0.2 \cdot RS + 0.016 \cdot SI^2 - 0.02 \cdot SI \cdot RS + 0.020 \cdot RS^2)/100 \tag{56}$$

$$a_3 = (209 - 7.20 \cdot SI - 7.71 \cdot RS + 1.33 \cdot SI^2 - 0.819 \cdot SI \cdot RS + 1.19 \cdot RS^2)/100 \tag{57}$$

By means of our uncertainty analysis tool we now investigate this model. We first consider a stand of age 50, with a site index SI of 3.0 and a relative stocking RS of 80

percent. The resulting sensitivity coefficients for the stem phytomass fraction are listed in Table 2. For the age variable we find a scaled sensitivity of 1.11. This means that if

Parameter	Sensitivity	Scaled Sensitivity
A	1.8089	1.1144
SI	−21.9966	−0.8131
RS	101.2832	0.9983
c_0	255.8697	1.0000
c_1	89.1654	0.0489
c_2	317.5071	0.5234
c_3	254.0057	−0.5708
c_4	203.2046	0.2131
$a_{1,0,0}$	0.1790	0.0292
$a_{1,0,1}$	0.1432	1.5506
$a_{1,0,2}$	0.1146	0.1322
$a_{1,1,0}$	0.5370	−0.1465
$a_{1,2,0}$	1.6111	0.0755
$a_{1,1,1}$	0.4296	−0.6409
$a_{2,0,0}$	3926.7868	1.4149
$a_{2,0,1}$	3141.4295	−0.0774
$a_{2,0,2}$	2513.1436	0.0063
$a_{2,1,0}$	11780.3605	−0.3629
$a_{2,2,0}$	35341.0815	0.0679
$a_{2,1,1}$	9424.2884	0.0232
$a_{3,0,0}$	−32.5342	−0.8389
$a_{3,0,1}$	−26.0273	0.0247
$a_{3,0,2}$	−20.8219	−0.0031
$a_{3,1,0}$	−97.6025	0.0865
$a_{3,2,0}$	−292.8076	−0.0480
$a_{3,1,1}$	−78.0820	−0.0079

Table 2: Sensitivity of stem phytomass with respect to model parameters.

we change the change the age by 5% then the resulting change in stem phytomass will be approximately $1.11 \times 5\% = 5.55\%$. From the table we can observe that the stem phytomass is most sensitive with respect to A, SI, RS, c_0, $a_{1,0,1}$, $a_{2,0,0}$ and $a_{3,0,0}$. A similar pattern is observed for the other phytomass fractions. In general we can conclude that the model is well behaved and there are no unstable parameters which might cause sudden changes in the dependent variables.

As an illustration of a transformation of covariance information on the independent variables to covariance of the dependent variables, in this case the phytomasses of the five fractions, we consider two cases. First we consider the case where all coefficients determined by regression, i.e. $c_i, i = 0, \ldots, 4$ and $a_{ik_1k_2}, i = 1, 2, 3; k_1 = 0, 1, 2; k_2 = 0, 1, 2$ are exact, i.e. can be determined with negligible variances. For the remaining variables A, SI and GS we assume that the standard errors are 5% and the cross-correlations given by 0.25. The resulting standard deviations and standard errors, and correlations of the phytomass fractions for a stand of age 50, site index 3.0 and relative stocking of 80% are listed in Table 3 and Table 4, respectively. Both first and second order approximations are provided. Notice that for this case we have very high correlations between the phytomass fractions.

Phytomass fraction	Phytomass	Standard Deviation	Standard Error
Stem	81.162	6.545	0.081
		(6.540)	(0.081)
Bark	8.169	0.440	0.054
		(0.440)	(0.054)
Branches	12.242	0.786	0.064
		(0.784)	(0.064)
Needles	7.990	0.373	0.047
		(0.373)	(0.047)
Roots	27.846	3.166	0.114
		(3.163)	(0.114)

Table 3: First and second order standard deviation and standard error approximations (first order approximations in brackets) for accurate regression coefficients.

Phytomass fraction	Stem	Bark	Branches	Needles	Roots
Stem	1.000	0.913	0.974	0.849	0.911
	(1.000)	(0.914)	(0.976)	(0.851)	(0.911)
Bark	0.913	1.000	0.930	0.968	0.930
	(0.913)	(1.000)	(0.931)	(0.969)	(0.931)
Branches	0.974	0.930	1.000	0.921	0.976
	(0.976)	(0.931)	(1.000)	(0.921)	(0.977)
Needles	0.849	0.968	0.921	1.000	0.965
	(0.851)	(0.969)	(0.921)	(1.000)	(0.967)
Roots	0.911	0.930	0.976	0.965	1.000
	(0.911)	(0.931)	(0.977)	(0.967)	(1.000)

Table 4: First and second order phytomass correlation approximations (first order approximations in brackets) for accurate regression coefficients.

Next we consider a more realistic case where the regression coefficients are not assumed to be known without error. As an example we assume that the regression coefficients have standard errors of 5% and random cross-correlations (also with A, SI and RS) which are uniformly distributed between -0.5 and 0.5. The covariance of A, SI and RS is the same as in the previous case. The resulting standard deviations and standard errors, and correlations of the phytomass fractions are listed in Table 5 and Table 6, respectively.

As can be expected due to the increase in uncertainty in the values of the regression coefficients, the standard deviations and standard errors are larger for this case. We also observe that the correlations between the phytomass fractions are now much smaller than in the previous case. We conclude that for reliable estimation of the response variable correlations, reliable information about the covariance of all of the independent variables, i.e. including the regression coefficients, is essential. In the following section on parameter estimation for nonlinear models it will be shown how reliable covariance information for regression coefficients can be obtained.

In Table 7 we have listed the computed standard error sensitivities for the two cases that we have considered. We observe that the standard error sensitivity of the age variable is equal to 0.5. This means, for example, that if we decrease the standard error of the age variable by 5% the resulting standard error of the stem phytomass will decrease by

Phytomass fraction	Phytomass	Standard Deviation	Standard Error
Stem	81.162	11.806	0.145
		(11.715)	(0.144)
Bark	8.169	1.117	0.137
		(1.109)	(0.136)
Branches	12.242	2.669	0.218
		(2.638)	(0.215)
Needles	7.990	0.918	0.115
		(0.913)	(0.114)
Roots	27.846	10.487	0.376
		(10.123)	(0.364)

Table 5: First and second order standard deviation and standard error approximations (first order approximations in brackets) for inaccurate regression coefficients.

Phytomass fraction	Stem	Bark	Branches	Needles	Roots
Stem	1.000	0.432	0.685	−0.174	0.607
	(1.000)	(0.432)	(0.691)	(−0.180)	(0.623)
Bark	0.432	1.000	0.686	−0.328	0.654
	(0.432)	(1.000)	(0.692)	(−0.338)	(0.671)
Branches	0.685	0.686	1.000	0.350	0.390
	(0.691)	(0.692)	(1.000)	(0.353)	(0.405)
Needles	−0.174	−0.328	0.350	1.000	0.392
	(−0.180)	(−0.338)	(0.353)	(1.000)	(0.405)
Roots	0.607	0.654	0.390	0.392	1.000
	(0.623)	(0.671)	(0.405)	(0.405)	(1.000)

Table 6: First and second order phytomass correlation approximations (first order approximations in brackets) for inaccurate regression coefficients.

approximately $0.5 \times 5\% = 2.5\%$. From the table it can be seen which variables are most important for the accuracy of the stem phytomass. For the case of inaccurate regression coefficients it can be concluded that an increase in accuracy will yield optimal results for the $a_{1,0,1}$ parameter. Other relatively error sensitive parameters are the three regressor variables and c_0. Just as the response covariance, the computation of the standard error sensitivity coefficients requires reliable estimates of the covariance of the independent variables.

Several other small models for growing stock and tree growth have been investigated using ***Oak***.

Parameter	Standard Error Sensitivity			
	Case 1		Case 2	
A	0.5030	(0.498)	0.1079	(0.1046)
SI	0.0921	(0.0892)	−0.1139	(−0.1130)
RS	0.4167	(0.4127)	0.1779	(0.1621)
c_0	0.0	(0.0)	0.1333	(0.1256)
c_1	0.0	(0.0)	−0.0091	(−0.0086)
c_2	0.0	(0.0)	0.1095	(0.1036)
c_3	0.0	(0.0)	0.0843	(0.0767)
c_4	0.0	(0.0)	0.0692	(0.0637)
a_{100}	0.0	(0.0)	−0.0063	(−0.0062)
a_{101}	0.0	(0.0)	0.3206	(0.3035)
a_{102}	0.0	(0.0)	0.0256	(0.0246)
a_{110}	0.0	(0.0)	0.0005	(0.0004)
a_{120}	0.0	(0.0)	0.0043	(0.0043)
a_{111}	0.0	(0.0)	0.0724	(0.0659)
a_{200}	0.0	(0.0)	0.1151	(0.0877)
a_{201}	0.0	(0.0)	0.0040	(0.0030)
a_{202}	0.0	(0.0)	−0.0011	(−0.0010)
a_{210}	0.0	(0.0)	−0.0059	(−0.0080)
a_{220}	0.0	(0.0)	−0.0035	(−0.0035)
a_{211}	0.0	(0.0)	0.0006	(0.0007)
a_{300}	0.0	(0.0)	0.0142	(0.0097)
a_{301}	0.0	(0.0)	0.0016	(0.0016)
a_{302}	0.0	(0.0)	0.0002	(−0.0002)
a_{310}	0.0	(0.0)	0.0011	(−0.0014)
a_{320}	0.0	(0.0)	0.0048	(0.0044)
a_{311}	0.0	(0.0)	0.0001	(0.0001)

Table 7: Standard error sensitivity of stem phytomass for both accurate (case 1) and inaccurate regression coefficients (case 2) (First and second order approximations with first order approximation between brackets).

4 Parameter estimation for nonlinear models

4.1 Overview

We start by giving a concise overview of parameter estimation for nonlinear models that should capture the essential ideas of the techniques in general use nowadays.

We consider a system response function R describing a nonlinear model

$$R(\alpha, \theta) : \mathbb{R}^q \times \Theta \to \mathbb{R}^m \tag{58}$$

where $\Theta \subset \mathbb{R}^p$. The response depends on

(i) $\alpha = (\alpha_1, \ldots \alpha_q)^T$, the regressor or explanatory variables, and

(ii) $\theta = (\theta_1, \ldots, \theta_p)^T$ the vector of unknown parameters to be estimated.

We consider the situation in which N multivariate observations have been made. These observations can be represented as a sequence of regressor variables vectors, each combined with a vector of corresponding response observations, i.e.

$$\{\alpha_i, y_i\} \tag{59}$$

where

$$\alpha_i = (\alpha_{i1}, \ldots, \alpha_{iq})^T \quad \text{and} \quad y_i = (y_{i1}, \ldots, y_{im})^T, \qquad i = 1, \ldots, N. \tag{60}$$

We then have

$$y_i = R(\alpha_i, \theta) + \varepsilon_i(\theta), \qquad i = 1, \ldots, N. \tag{61}$$

where $\varepsilon_i(\theta)$ is the difference between the vector of observed responses y_i and the model prediction by the system response function $R(\alpha_i, \theta)$.

It is assumed that there is one *true* parameter vector denoted by θ^* which is known to be in $\Theta \subset \mathbb{R}^n$. The *residuals* $\varepsilon_i(\theta^*)$ are assumed to be random vectors with $\mathbf{E}[\varepsilon_i] = 0$.

It will be useful to group the observations into one *observation vector*

$$\mathbf{y} = ((y_{11}, \ldots, y_{1m}), \ldots, (y_{N1}, \ldots, y_{Nm}))^T = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}. \tag{62}$$

Similarly we group the corresponding system responses in the *prediction map*

$$\mathbf{R} : \Theta \to \mathbb{R}^{N \times m} \tag{63}$$

defined by

$$\mathbf{R}(\theta) = ((R_1(\alpha_1, \theta), \ldots, R_m(\alpha_1, \theta)), \ldots, (R_1(\alpha_N, \theta), \ldots, R_m(\alpha_N, \theta))) = \begin{pmatrix} R(\alpha_1, \theta) \\ R(\alpha_2, \theta) \\ \vdots \\ R(\alpha_N, \theta) \end{pmatrix}, \tag{64}$$

and the $\varepsilon_i(\theta)$ into

$$\varepsilon(\theta) = \mathbf{y} - \mathbf{R}(\theta). \tag{65}$$

We shall repeatedly use the derivative of the prediction map with respect to θ. We use the following notation for the Jacobian matrix

$$J(\theta) = D_\theta \mathbf{R}(\theta) = \frac{d\mathbf{R}}{d\theta}(\theta) \quad \text{and} \quad J = J(\theta^*). \tag{66}$$

Now consider

$$\mathcal{P} = \{\mathbf{R}(\theta) | \theta \in \Theta\} \tag{67}$$

which is a manifold in $\mathbb{R}^{N \times m}$ which represents all possible predictions by the model for the available observations. It is usually referred to as the *expectation surface* or the *solution locus*. It seems reasonable from a geometrical point of view (ignoring probability information about the residuals for the moment) to choose the parameter vector θ in such a way that the corresponding prediction $\mathbf{R}(\theta)$ is the projection of the observation vector onto $\mathcal{P}$ (with respect to some inner product and induced norm $\|\cdot\|$). So we choose θ such that it is closest to the observation vector (and precisely for this reason it is sometimes the case that this estimator corresponds to the maximum likelihood estimator if the distribution of ε is taken into account). This estimator $\hat{\theta}$ is given by

$$\hat{\theta} = \min_{\theta \in \Theta} S(\theta) = \min_{\theta \in \Theta} \|\mathbf{y} - \mathbf{R}(\theta)\|^2 = \min_{\theta \in \Theta} \|\varepsilon\|^2 \tag{68}$$

It follows with $\hat{J} = J(\hat{\theta})$, that $\hat{\theta}$ must satisfy

$$\hat{J}^T(\mathbf{y} - \mathbf{R}(\hat{\theta})) = \hat{J}^T \varepsilon(\hat{\theta}) = 0. \tag{69}$$

This equation states that the difference between the observation vector and the prediction corresponding to $\hat{\theta}$ is orthogonal to the tangent plane of the expectation surface $\mathcal{P}$ at $\hat{\theta}$. If we use the projection matrix $\hat{P}_J = \hat{J}(\hat{J}^T\hat{J})^{-1}\hat{J}^T$, this can be expressed as

$$\hat{P}_J \, \varepsilon(\hat{\theta}) = 0. \tag{70}$$

These equations are called the *normal equations*, which state that the projection of $\varepsilon(\hat{\theta})$ has no component in the tangent plane of the expectation surface at $\hat{\theta}$. For most nonlinear models they cannot be solved analytically, so that iterative methods must be used to obtain $\hat{\theta}$. Notice that since a solution of (68), or equivalently (70), does not have to be unique, convergence of the iterative methods can, in general, not be guaranteed.

Once we have found an estimator $\hat{\theta}$ we want to know how reliable this estimator is. The results we present here rest on two important assumptions:

(i) the residual vector ε is normally distributed $\varepsilon \sim N(0, \sigma^2 I)$, i.e. we assume that all components are independent and normally distributed with equal variance σ^2. In appendix A we deal with the question what to do in the case that the assumption of equal variance is not valid. It is shown that if information about the covariance is available, this can be used to transform the parameters using a Cholesky decomposition.

(ii) $\hat{\theta}$ is sufficiently close to θ^* such that a linear approximation of the prediction map may be used. It can be shown that under certain regularity conditions, $\hat{\theta}$ is almost certain to be within a small neighborhood of θ^*, see [SW89, Chapter 12].

In the following we will refrain from using the order notation and will understand '$\approx$' to mean 'equal in first order approximation'. We first linearize the prediction map around the true parameter vector θ^*, i.e.

$$\mathbf{R}(\theta) \approx \mathbf{R}(\theta^*) + J\,(\theta - \theta^*). \tag{71}$$

To be able to use linear regression theory for the uncertainty analysis of $\hat{\theta}$ the key idea is now the following: if $\hat{\theta}$ is in a region where the linear approximation (71) is still acceptable, the projection of the residual vector on the tangent plane at θ^* will be close to the

projection on the expectation surface $\mathcal{P}$. Remember that the projection of the residual vector on the expectation surface gives the exact $\hat{\theta}$; projection on the tangent plane yields an approximation. We thus have

$$P_J(\mathbf{y} - \mathbf{R}(\theta^*)) = P_J\,\varepsilon \approx \mathbf{R}(\theta) - \mathbf{R}(\theta^*), \tag{72}$$

where

$$P_J = J(J^TJ)^{-1}J^T. \tag{73}$$

This also means that, since $\mathbf{R}(\hat{\theta}) - \mathbf{R}(\theta^*) \approx J(\hat{\theta} - \theta^*)$

$$\hat{\theta} - \theta^* \approx (J(\theta^*)^TJ(\theta^*))^{-1}J(\theta^*)^T\varepsilon, \tag{74}$$

which gives us a relationship between the deviation of the true parameter vector and its estimator in terms of the residual vector. This can be used to transform the stochastic properties of the residual vector to those of the deviation of the parameter estimate.

Notice that we also have

$$\mathbf{y} - \mathbf{R}(\hat{\theta}) \approx \varepsilon - P_J\varepsilon = (I - P_J)\varepsilon, \tag{75}$$

and evaluating the norm we get

$$(Nm - p)s^2 = S(\hat{\theta}) = \|\mathbf{y} - \mathbf{R}(\hat{\theta})\| \approx \varepsilon^T(I - P_J)\varepsilon \tag{76}$$

Here we have introduced

$$s^2 = \frac{S(\hat{\theta})}{Nm - p} \tag{77}$$

which is an estimator for the error variance σ^2.

Further it can easily be checked now that

$$\|\mathbf{R}(\hat{\theta}) - \mathbf{R}(\theta^*)\| \approx (\hat{\theta} - \theta^*)^TJ(\theta^*)^TJ(\theta^*)(\hat{\theta} - \theta^*) \approx \varepsilon^TP_J\varepsilon \approx S(\theta^*) - S(\hat{\theta}) \tag{78}$$

As a final important note we remark that in the above expressions $J(\theta^*)$ may be replaced by $J(\hat{\theta})$ without changing the results. This can be checked by approximating $J(\theta^*)$ by $J(\hat{\theta}) + H(\theta^* - \hat{\theta})$, where H is the second order derivative of $\mathbf{R}$ evaluated at $\hat{\theta}$.

From the distribution of the residual vector we now have the following theorem. For the regularity conditions and details of the proof, we refer to [SW89].

Theorem 4.1 Given $\varepsilon \sim N(0, \sigma^2)$ and appropriate regularity conditions then, for large N, we have approximately:

(i) $\hat{\theta} - \theta^* \sim N(0, \sigma^2C^{-1})$, where $C = J(\theta^*)^TJ(\theta^*)$;

(ii) $S(\hat{\theta})/\sigma^2 \approx \varepsilon^T(I - P_J)\varepsilon/\sigma^2 \sim \chi^2_{Nm-p}$;

(iii) $\hat{\theta}$ is statistically independent of s^2; and

(iv) $\frac{(S(\theta^*)-S(\hat{\theta}))/p}{S(\hat{\theta})/(Nm-n)} \approx \frac{\varepsilon^TP_J\varepsilon}{\varepsilon^T(I-P_J)\varepsilon}\frac{Nm-p}{p} \sim \mathcal{F}_{p,Nm-p}$

Here χ^2_{Nm-p} is the Chi-square distribution with $Nm - p$ degrees of freedom and $\mathcal{F}_{p,Nm-p}$ the F-distribution with p and $Nm - p$ degrees of freedom. From (iv), (77) and (78) we have, approximately

$$\frac{(\hat{\theta} - \theta^*)^TJ^TJ(\hat{\theta} - \theta^*)}{ps^2} \sim \mathcal{F}_{p,Nm-p} \tag{79}$$

As mentioned we are considering a first order approximation, making that $J(\theta^*)$ may be replaced by $\hat{J} = J(\hat{\theta})$. An approximate $100(1-\alpha)\%$ confidence region is given by

$$\left\{\theta | (\hat{\theta} - \theta^*)^T \hat{J}^T \hat{J} (\hat{\theta} - \theta^*) \leq p s^2 \mathcal{F}_{p,Nm-p}\right\} \tag{80}$$

This confidence region can be investigated using the singular value decomposition of $\hat{J}$, see for instance [Sto98]

$$\hat{J} = U \Sigma V^T \tag{81}$$

The approximate $100(1-\alpha)\%$ confidence region now becomes

$$\left\{\theta | (\hat{\theta} - \theta^*)^T V \Sigma^2 V^T (\hat{\theta} - \theta^*) \leq p s^2 \mathcal{F}_{p,Nm-p}\right\} \tag{82}$$

This confidence region is an ellipsoid with the columns of V pointing in the direction of the ellipsoid's main axes. Projection of the ellipsoid on the parameter axes gives the *independent confidence intervals*:

$$\left[\hat{\theta}_i - \delta^I \theta_i, \hat{\theta}_i + \delta^I \theta_i\right] \tag{83}$$

where

$$\delta^I \theta_i = \sqrt{p s^2 \mathcal{F}_\alpha(p, Nm-p)(V \Sigma^{-2} V^T)_{ii}} \tag{84}$$

and where $\mathcal{F}_\alpha(p, Nm-p)$ denotes the upper α quantile for the F-distribution with p and $Nm-p$ degrees of freedom. The notation $(\cdot)_{ii}$ is used to denote the i-th diagonal element of a matrix. The intersection points of the ellipsoid with the parameter axes gives the *dependent confidence intervals*:

$$\left[\hat{\theta}_i - \delta^D \theta_i, \hat{\theta}_i + \delta^D \theta_i\right] \tag{85}$$

where

$$\delta^D \theta_i = \sqrt{\frac{p s^2 \mathcal{F}_\alpha(p, Nm-p)}{(V \Sigma^2 V^T)_{ii}}} \tag{86}$$

4.2 Assessing and extending linear inference theory

We have seen in the previous section how linear inference theory can be used to obtain estimates for the covariance and confidence intervals of the parameters. These estimates obtain their listed characteristics such as unbiasedness and normality only asymptotically, i.e. for large number of observations.

If the number of observations is moderate or small there may be substantial bias in both the estimate itself and in the parameter covariance. It is however possible to obtain first order approximations of these biases. These approximations make use of the curvature of the expectation surface and thus rely on being able to compute second order derivatives. An excellent overview of the relevant theory is presented in [SW89].

A first order approximation of the estimator bias is derived in [Box71]. It is shown that

$$E(\theta^* - \hat{\theta}) \approx -\frac{\sigma^2}{2} (\hat{J}^T \hat{J})^{-1} \hat{J}^T v \tag{87}$$

where v is an Nm vector, where the component with index $k = i\,m + j$, corresponding with response j of observation i, is given by

$$v_i = \mathrm{Tr}\left(H_k (\hat{J}^T \hat{J})^{-1}\right) \tag{88}$$

and H_k is the Hessian of $R_j(\alpha_i, \theta)$ with respect to θ evaluated at $\hat{\theta}$.

The relative bias is given by $\mathbf{E}(\theta^* - \hat{\theta})/\hat{\theta}$. In [Rat83] it is suggested that an absolute relative bias in excess of 1% is a good rule for indicating nonlinear behaviour in $\hat{\theta}$.

For normal data, empiral studies indicate that the approximation works very well, especially if the bias is large. See [Box71], [GR78] and [Rat83]. The bias vector can also help to indicate problem parameters. For further details on the properties of the bias approximation, see [CTW86].

In [Bea60] four measures of the nonlinearity of an estimation problem are proposed. Further insight in the nonlinearity of the problem can be acquired by computation of the *intrinsic* and *parameter-effects* curvature at the parameter estimate, as described in [BW80]. In the article it is shown that the bias estimate of Box can also be expressed in terms of these curvatures.

In [SW89, Chapter 4] higher order correction terms for the asymptotic covariance matrix are derived. These correction terms can be expressed in terms of the intrinsic and parameter-effects curvature. In [Cla80] an even higher order covariance correction is derived which involves the computation of third order derivatives.

The measures of nonlinearity and the covariance corrections will be implemented in ***Oak*** in the near future.

4.3 The role of automatic differentiation

Automatic differentiation should be the method of choice to obtain the derivatives for both the estimation of the parameters itself and for the subsequent uncertainty analysis.

For the estimation of the parameters an optimization problem must be solved. Especially for large numbers of parameters derivative information is then essential. Automatic differentiation is particularly efficient at obtaining the gradient of a function. As was shown in section 2.3 this can be achieved at a cost which is a small fixed multiple of the evaluation cost of the function to be optimized. If the numerical method of finite differences is used the multiple is linear in the number of independent variables. Since the gradient must be computed at each step of an iterative optimization procedure, such as the conjugate gradients or quasi-newton method, large gains in run time can be achieved by using automatic differentiation. The optimization algorithms can also benefit from the higher accuracy achieved by automatic differentiation compared to the approximation by finite differences.

For the uncertainty analysis of the parameter estimates we can first of all use automatic differentiation to obtain the Jacobian of the prediction map. This derivative can be used for the estimates that follow from linear inference theory for the asymptotic covariance matrix and asymptotic confidence intervals. Following that we can evaluate the second order derivative of the prediction map which is required for the computation of the bias approximation. The same derivative can be used to calculate intrinsic and parameter effects curvatures to assess the nonlinearity of the estimation problem, and to compute a nonlinearity correction for the parameter covariance matrix. This correction can be improved by computing the third order derivatives.

4.4 An example (continued) – phytomass of Siberian forests

We continue the example on Siberian forests phytomass that was described in Section 3.5. As an uncomplicated illustration of using the ***Oak*** system for nonlinear regression, we fit a growing stock relationship to a set of observations. The growing stock is assumed to be

given by

$$GS = a_1(1 - \exp(-a_2 A))^{a_3} \tag{89}$$

This equation is the same as (53). We do not take the dependency of $a_i, i = 1, 2, 3$ on the site index and relative stocking into account anymore, and consider the growing stock solely as a function of age A.

In Figure 2 the growing stock data is displayed together with the fit obtained using ***Oak***.

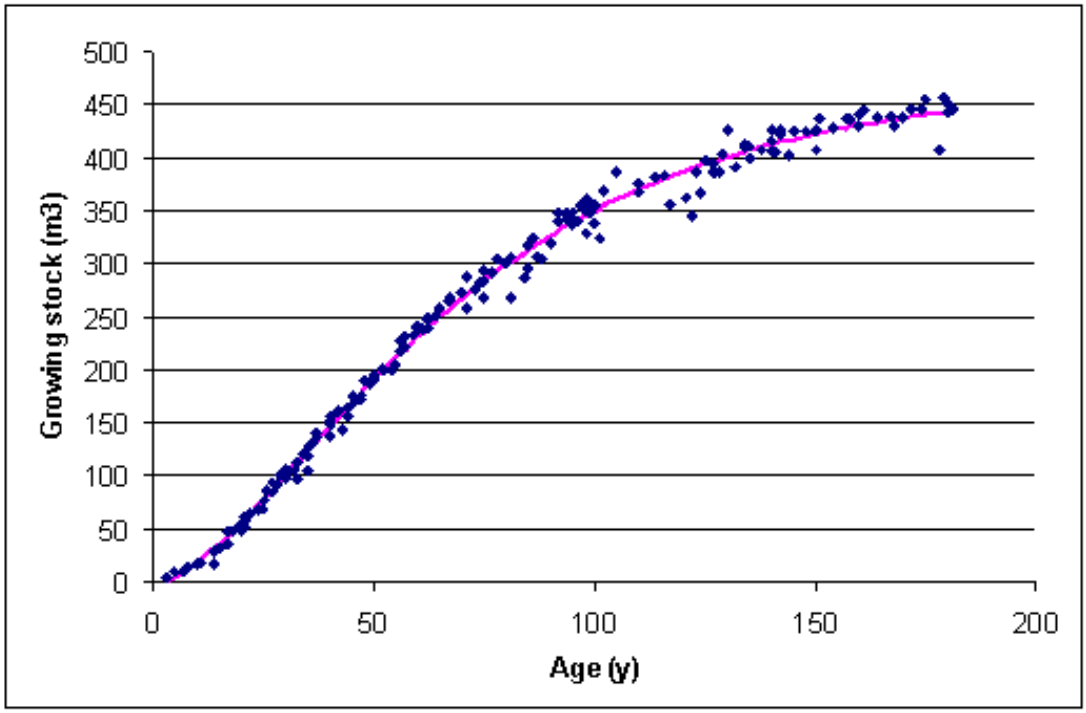

Figure 2: Growing stock volume vs. age with fit obtained by ***Oak***.

In Table 8 the parameter estimates and corresponding bias obtained by the ***Oak*** system are listed. As can be seen the bias is negligible compared to parameter values; we can conclude that the model is sufficiently linear to proceed by means of linear inference theory. In Table 9 the 90% independent and dependent confidence intervals for the parameters a_1, a_2 and a_3 are listed.

Parameter	Estimate	Bias
a_1	482.6	0.28
a_2	0.01824	$8.5e-06$
a_3	1.849	$7.6e-04$

Table 8: Results of nonlinear parameter estimation using the ***Oak*** system.

The estimates for the parameter standard deviation and standard error, and the correlations between the parameters obtained from the asymptotic covariance matrix are listed in Table 10 and Table 11, respectively.

Currently also a multiple response model for phytomass fractions is under investigation.

Parameter	Independent confidence interval		Dependent confidence interval	
a_1	456.7	508.5	474.3	490.9
a_2	0.01645	0.02003	0.01793	0.01855
a_3	1.755	1.942	1.823	1.875

Table 9: 90% independent and dependent confidence intervals for the parameters a_1, a_2 and a_3.

Parameter	Standard Deviation	Standard Error
a_1	10.3	0.0213
a_2	0.000712	0.0390
a_3	0.00139	0.0201

Table 10: Standard deviation and standard error approximations for the parameters a_1, a_2 and a_3.

Parameter	a_1	a_2	a_3
a_1	1.00	-0.889	-0.687
a_2	-0.889	1.00	0.918
a_3	-0.687	0.918	1.00

Table 11: Correlation coefficient approximations for the parameters a_1, a_2 and a_3.

5 *Oak* – A system for uncertainty analysis based on automatic differentiation

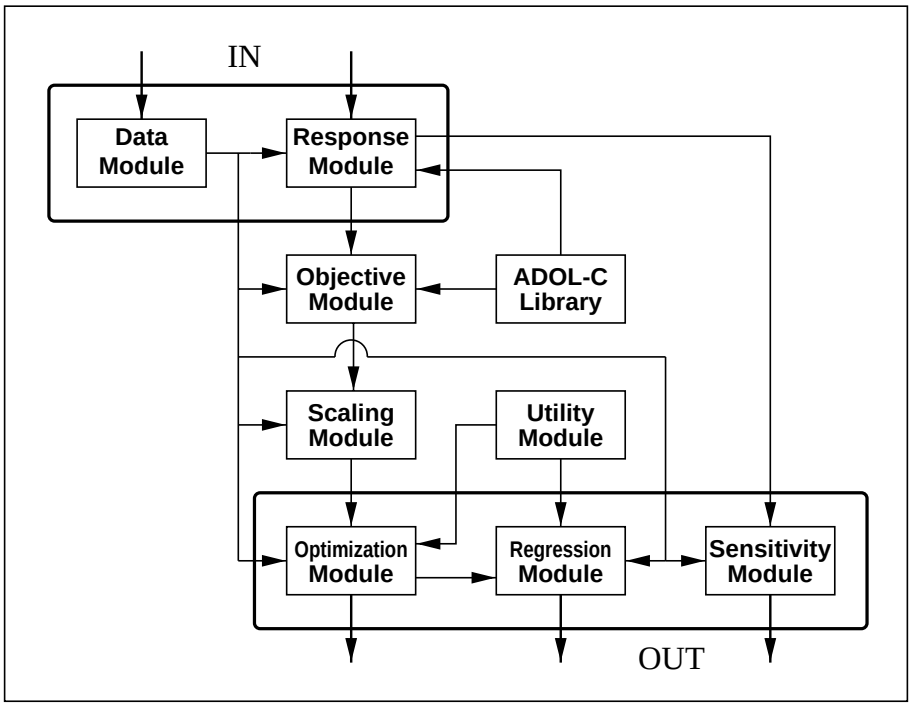

Figure 3: Schematic representation of Oak.

A system named ***Oak*** written in C++ has been developed which implements the methods described in the previous sections using the ADOL-C library for automatic differentiation. (see [GJSTar]). This section does not provide a user manual, but aims to present a brief overview of its structure and possibilities.

The user is expected to provide the (C++) code representing his/her model and add it to the ***Oak*** system. The system can then be used to perform a sensitivity analyis of the model, or if observations relating to the model are provided, be used for nonlinear parameter estimation. Both types of analysis are operated from a graphical user interface. The system is specifically designed for large scale parameter estimation.

Figure 3 shows the structure of the system. It consists of several modules handling the various tasks:

- **Data module** – Sets the number of parameters, regressor variables, responses. Reads (optional) observational data. Reads (optional) covariance information parameters and regressor variables.

- **Response module** – Contains the functions descibing the user's model.

- **Objective module** – Contains the objective function to be optimized for the parameter estimation problem. Can be changed by the user if necessary.

- **Scaling module** – Handles the scaling of the problem for efficient optimization.

- **Optimization module** – Contains the optimization routines. Can be extended by the user if necessary. Currently the optimization is performed by means of a conju-

gate gradients algorithm combined with a Brent line minimization algorithm. Also positive definiteness of the Hessian at the stationary point can be checked.

- **Utility module** – Contains various routines, e.g. for calculating eigenvalues of the Hessian matrix, computing α-percentiles of the χ^2 and F distributions and computing a singular value decomposition of the prediction map derivative.

- **Sensitivity module** – Computes sensitivity coefficients. Transforms covariance of independent variables into covariance of the dependent variables. Computes standard error sensitivity coefficients.

- **Regression module** – Computes covariance, confidence intervals and bias of estimated parameters. Performs an analysis of variance and computes the significance of the regression.

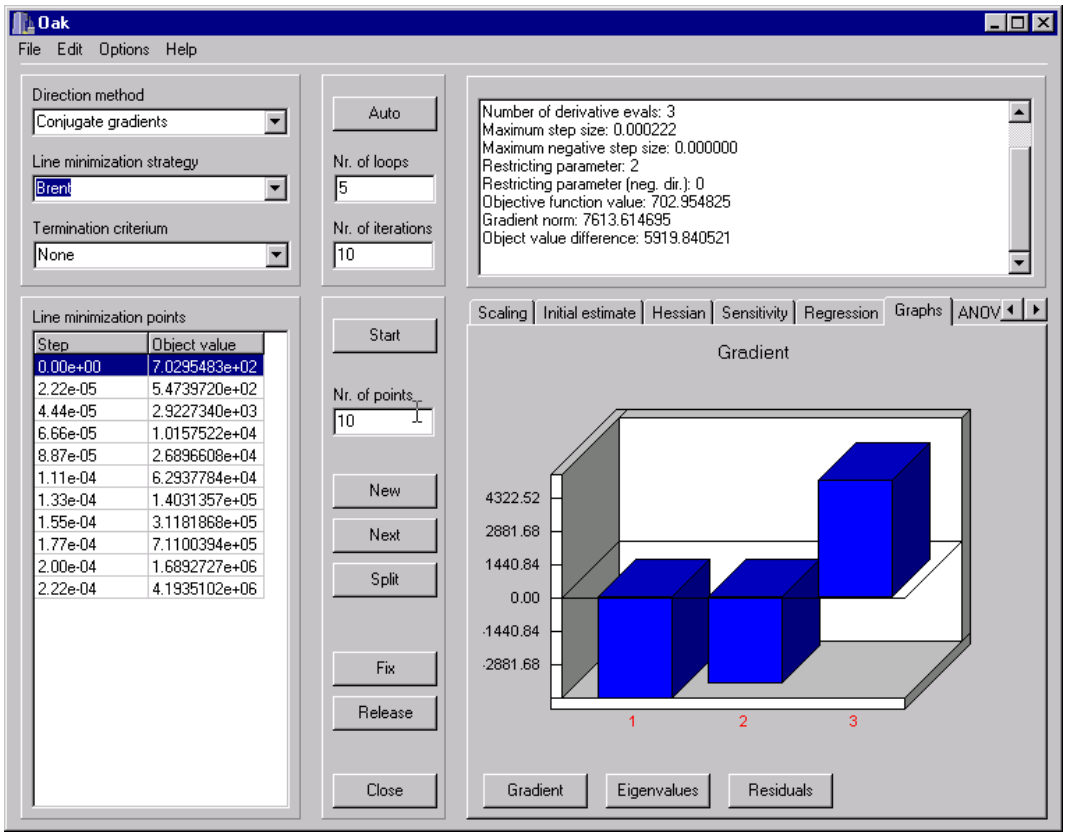

Figure 4: Screenshot Oak – Optimization and graphs

Figures 4, 5 and 6 show some screenshots of the ***Oak*** system.

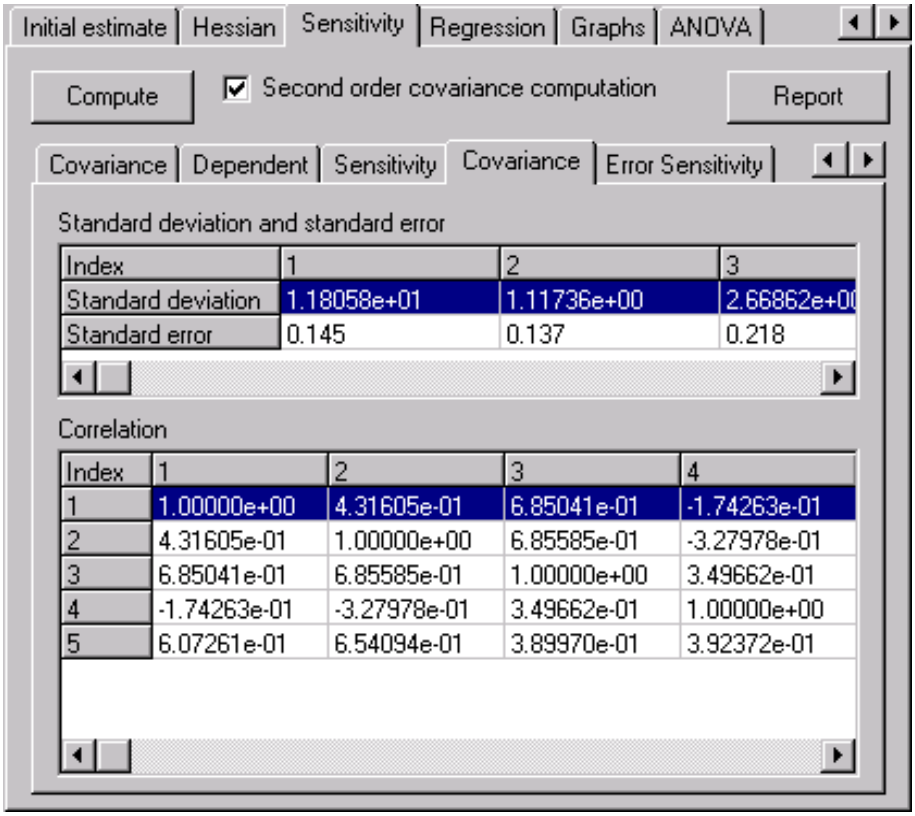

Figure 5: Screenshot Oak – Sensitivity methods: response covariance

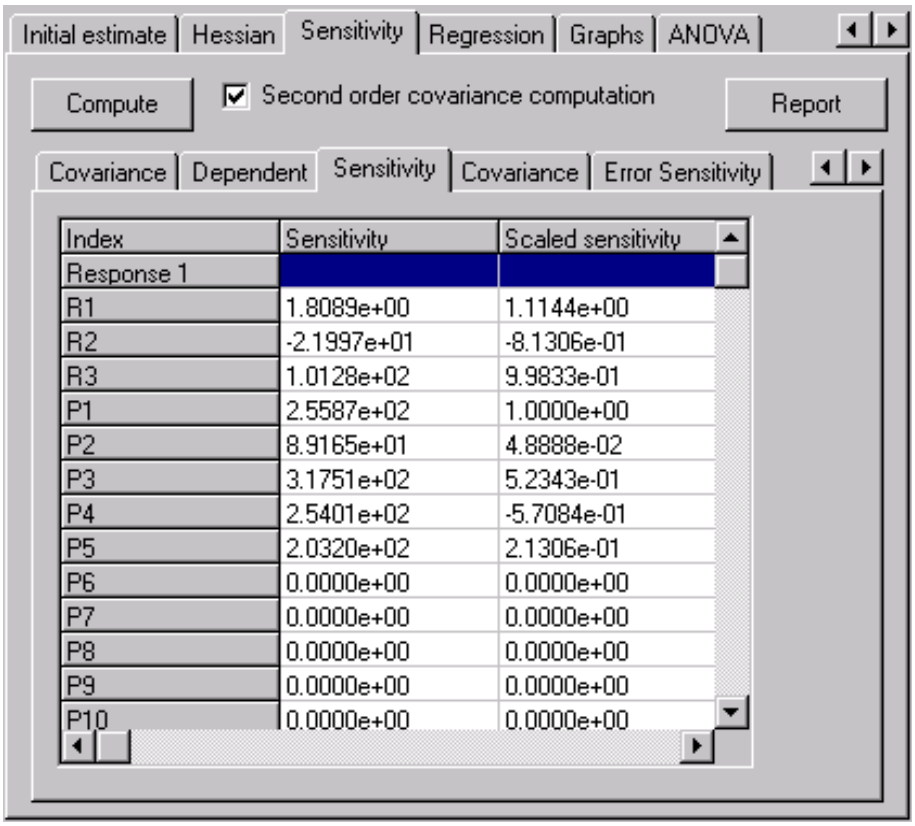

Figure 6: Screenshot Oak – Sensitivity methods: sensitivity coefficients

A Transforming estimation problems to guarantee i.i.d. residuals

Suppose we have

$$\tilde{y}_i = \tilde{R}(\alpha_i, \theta^*) + \tilde{\varepsilon}_i, \qquad i = 1, \ldots, N \tag{90}$$

with

$$\mathbf{E}[\tilde{\varepsilon}] = 0, \mathrm{cov}(\tilde{\varepsilon}, \tilde{\varepsilon}) = V\sigma^2 \quad \text{and} \quad \varepsilon \sim \mathcal{N}(0, V\sigma^2), \tag{91}$$

i.e. with arbitrary error covariance matrix $V\sigma^2$. Let

$$V = L^2 \tag{92}$$

be the Cholesky decomposition of V (which is symmetric and positive definite). By multiplication of the left and right sides of (90) by L^{-1} and setting

$$y_i = L^{-1}\tilde{y}_i, \quad R = L^{-1}\tilde{R} \quad \text{and} \quad \varepsilon = L^{-1}\tilde{\varepsilon} \tag{93}$$

we get

$$y_i = R(\alpha_i, \theta^*) + \varepsilon_i, \qquad i = 1, \ldots, N \tag{94}$$

where

$$\mathbf{E}[\varepsilon] = L^{-1}\tilde{\varepsilon} = 0 \quad \text{and} \quad \mathrm{cov}(\varepsilon, \varepsilon) = L^{-1}\mathrm{cov}(\tilde{\varepsilon}, \tilde{\varepsilon})L^{-1} = \sigma^2 I \tag{95}$$

References

[BCG93] Christian Bischof, George Corliss, and Andreas Griewank. Structured second- and higher order derivatives through univariate taylor series. *Optimization Methods and Software*, 2:211–232, 1993.

[Bea60] E.M.L. Beale. Confidence regions in nonlinear estimation. *Journal of Royal Statistical Society B*, 22:41–88, 1960.

[Box71] M.J. Box. Bias in nonlinear estimation. *Journal of Royal Statistical Society B*, 33:219–229, 1971.

[BW80] D.M. Bates and D.G. Watts. Relative curvature measures of nonlinearity. *Journal of the Royal Statistical Society B*, 42(1):1–25, 1980.

[BW88] D.M. Bates and D.G. Watts. *Nonlinear Regression Analysis and its Applications.* Wiley Series in Probability and Mathematical Statistics. John Wiley and Sons, Inc., New York, 1988.

[Chr91] D. Bruce Christianson. Reverse accumulation and accurate rounding error estimates for Taylor series coefficients. *Optimization Methods and Software*, 1(1):81–94, 1991. Also appeared as Technical Report No. NOC TR239, The Numerical Optimisation Centre, University of Hertfordshire, U.K., July 1991.

[Chr92] D. Bruce Christianson. Automatic Hessians by reverse accumulation. *IMA J of Numerical Analysis*, 12:135–150, 1992. Also appeared as Technical Report No. NOC TR228, The Numerical Optimisation Centre, University of Hertfordshire, U.K., April 1990.

[Cla80] G.P.Y. Clarke. Moments of the least squares estimators in a nonlinear regression model. *Journal of the Royal Statistical Society B.*, 42:227–237, 1980.

[CTW86] R.D. Cook, C.L. Tsai, and B.C. Wei. Bias in nonlinear regression. *Biometrika*, 73:615–623, 1986.

[DS81] N.R. Draper and H. Smith. *Applied Regression Analysis, second edition.* Wiley Series in Probability and Mathematical Statistics. John Wiley and Sons, Inc., New York, 1981.

[GJST92] Andreas Griewank, David Juedes, Jay Srinivasan, and Charles Tyner. ADOL-C, a package for the automatic differentiation of algorithms written in C/C++. *ACM Trans. Math. Software*, 1992. Also appeared as Preprint MCS–P180–1190, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., November 1990.

[GR78] P.R. Gillis and D.A. Ratkowsky. The behaviour of estimators of the parameters of various yield-density relationships. *Biometrics*, 34:191–198, 1978.

[Gri89] Andreas Griewank. On automatic differentiation. In M. Iri and K. Tanabe, editors, *Mathematical Programming: Recent Developments and Applications*, pages 83–108. Kluwer Academic Publishers, Dordrecht, 1989.

[Gri91] Andreas Griewank. Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. *Optimization Methods and Software*, 1991. Also appeared as Preprint MCS–P228–0491, Mathematics and

Computer Science Division, Argonne National Laboratory, Argonne, Ill., April 1991.

[Hil82] Kenneth E. Hillstrom. JAKEF – A portable symbolic differentiator of functions given by algorithms. Technical Report ANL–82–48, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., 1982.

[Hor91] Jim E. Horwedel. GRESS: A preprocessor for sensitivity studies on Fortran programs. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 243–250. SIAM, Philadelphia, Penn., 1991.

[Iri84] Masao Iri. Simultanious computation of functions, partial derivatives and estimates of rounding errors — Complexity and practicality. *Japan J. Applied Mathematics*, 1(2):223–252, 1984.

[Iri91] Masao Iri. History of automatic differentiation and rounding estimation. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 1–16. SIAM, Philadelphia, Penn., 1991.

[JM88] R.H.F. Jackson and G.P. McCormick. Second order sensitivity analysis in factorable programming: theory and applications. *Mathematical programming*, 41(1):1–28, 1988.

[Jue91] David Juedes. A taxonomy of automatic differentiation tools. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 315–329. SIAM, Philadelphia, Penn., 1991.

[Kub91] Koichi Kubota. PADRE2, a FORTRAN precompiler yielding error estimates and second derivatives. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 251–262. SIAM, Philadelphia, Penn., 1991.

[Kul96] Ulrich Kulisch. Memorandum ueber computer, arithmetik und numerik. Bericht 01–1996, Institut fuer Angewandte Mathematik, Universitaet Karlsruhe, D-76128 Karlsruhe, 1996.

[Rat83] D.A. Ratkowsky. *Nonlinear regression modeling.* Marcel Dekker, New York, 1983.

[Ron88] Yigal Ronen. Uncertainty analysis based on sensitivity analysis. In Yigal Ronen, editor, *Uncertainty analysis*, pages 41–70. CRC Press, Inc., Boca Ration, Florida, 1988.

[SSN98a] D. Shepashenko, A. Shvidenko, and S. Nilsson. Models for phytomass evaluation in stands of main forest forming species of siberia (in preparation). 1998.

[SSN98b] D. Shepashenko, A. Shvidenko, and S. Nilsson. Phytomass (live biomass) and carbon of siberian forests. *Biomass and Bioenergy*, 14(1):21–31, 1998.

[Sto98] Walter Stortelder. *Parameter estimation in nonlinear dynamical systems.* PhD thesis, Centrum voor Wiskunde en Informatica, Amsterdam, 1998.

[SVN96] Anatoly Shvidenko, Sergey Venevsky, and Sten Nilsson. Increment and mortality for major forest species of northern eurasia with variable growing stock. Working Paper WP–96–98, International Institute for Applied Systems Analysis, A-2361 Laxenburg, Austria, August 1996.

[SW89] G.A.F. Seber and C.J. Wild. *Nonlinear regresssion*. Wiley Series in Probability and Mathematical Statistics. John Wiley and Sons, Inc., New York, 1989.

Bibliography

Automatic Differentiation – General Theory

References

[Bau74] F. L. Bauer. Computational graphs and rounding errors. *SIAM J. on Numerical Analysis*, 11:87–96, 1974.

[BCG92] Christian Bischof, George Corliss, and Andreas Griewank. Structured second- and higher-order derivatives through univariate Taylor series. Preprint MCS–P296–0392, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., March 1992. ADIFOR Working Note # 6.

[BCG93] Christian Bischof, George Corliss, and Andreas Griewank. Structured second- and higher order derivatives through univariate taylor series. *Optimization Methods and Software*, 2:211–232, 1993.

[Bec94] Thomas Beck. Automatic differentiation of iterative processes. *Journal of Computational and Applied Mathematics*, 50:109–118, 1994.

[BF94] Thomas Beck and Herbert Fischer. The if-problem in automatic differentiation. *Journal of Computational and Applied Mathematics*, 50:119–131, 1994.

[BGJ91] Christian Bischof, Andreas Griewank, and David Juedes. Exploiting parallelism in automatic differentiation. In Elias Houstis and Yoichi Muraoka, editors, *Proceedings of the 1991 International Conference on Supercomputing*, pages 146–153. ACM Press, Baltimore, Md., 1991. Also appeared as Preprint MCS–P204–0191, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., January 1991.

[BH91] Christian Bischof and James Hu. Utilities for building and optimizing a computational graph for algorithmic decomposition. Technical Memorandum ANL/MCS–TM–148, Mathematics and Computer Sciences Division, Argonne National Laboratory, Argonne, Ill., April 1991.

[Bis91] Christian Bischof. Issues in parallel automatic differentiation. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 100–113. SIAM, Philadelphia, Penn., 1991.

[BS83] W. Baur and V. Strassen. The complexity of partial derivatives. *Theoretical Computer Science*, 22:317–330, 1983.

[Chr91] D. Bruce Christianson. Reverse accumulation and accurate rounding error estimates for Taylor series coefficients. *Optimization Methods and Software*, 1(1):81–94, 1991. Also appeared as Technical Report No. NOC TR239, The Numerical Optimisation Centre, University of Hertfordshire, U.K., July 1991.

[Chr92a] D. Bruce Christianson. Automatic Hessians by reverse accumulation. *IMA J of Numerical Analysis*, 12:135–150, 1992. Also appeared as Technical Report No. NOC TR228, The Numerical Optimisation Centre, University of Hertfordshire, U.K., April 1990.

[Chr92b] D. Bruce Christianson. Reverse accumulation and attractive fixed points. Technical Report NOC TR 258, The Numerical Optimisation Centre, University of Hertfordshire, Hatfield, UK, March 1992.

[Cor91a] George F. Corliss. Automatic differentiation bibliography. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 331–353. SIAM, Philadelphia, Penn., 1991.

[Cor91b] George F. Corliss. Overloading point and interval Taylor operators. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 139–146. SIAM, Philadelphia, Penn., 1991.

[Dix91] Lawrence C. W. Dixon. Use of automatic differentiation for calculating Hessians and Newton steps. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 114–125. SIAM, Philadelphia, Penn., 1991.

[DMM90] Lawrence C. W. Dixon, Z. Maany, and M. Mohseninia. Automatic differentiation of large sparse systems. *J. Economic Dynamics & Control*, 14(2), 1990. Presented at IFAC on Dynamic Modelling & Control of National Economies, Edinburgh, July, 1989. Also appeared as Technical Report NOC TR223, The Numerical Optimisation Center, Hatfield Polytechnic, Hatfield, U.K., July 1989.

[DP89] Lawrence C. W. Dixon and Richard C. Price. The truncated Newton method for sparse unconstrained optimisation using automatic differentiation. *J. Opt. Theory and Appl.*, 60(2):261 +, February 1989.

[DPS90] Paul H. Davis, John D. Pryce, and Bruce Stephens. Recent developments in automatic differentiation. In J. C. Mason and M. G. Cox, editors, *Scientific Software Systems*, pages 153–165. Chapman and Hall, 11 New Fetter Lane, London EC4P 4EE, 1990. Also appeared as Technical Report ACM–89–1, Royal Military College of Science at Shrivenham, Shrivenham, U.K.

[Fis90] Herbert Fischer. Automatic differentiation: Parallel computation of function, gradient and Hessian matrix. *Parallel Computing*, 13:101–110, 1990.

[Fis91] Herbert Fischer. Special problems in automatic differentiation. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 43–50. SIAM, Philadelphia, Penn., 1991.

[Fla91] Harley Flanders. Automatic differentiation of composite functions. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 95–99. SIAM, Philadelphia, Penn., 1991.

[Gay91] David M. Gay. Automatic differentiation of nonlinear AMPL models. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 61–73. SIAM, Philadelphia, Penn., 1991.

[GC91] Andreas Griewank and George F. Corliss, editors. *Automatic Differentiation of Algorithms: Theory, Implementation, and Application.* SIAM, Philadelphia, Penn., 1991.

[Gil92] J. Ch. Gilbert. Automatic differentiation and iterative processes. *Optimization Methods and Software*, 1:13–21, 1992. Also appeared as Preprint, INRIA, Le Chesnay, France, 1991.

[GR91] Andreas Griewank and Shawn Reese. On the calculation of Jacobian matrices by the Markowitz rule. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 126–135. SIAM, Philadelphia, Penn., 1991. Also appeared as Preprint MCS–P267–1091, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., January 1992.

[Gri89] Andreas Griewank. On automatic differentiation. In M. Iri and K. Tanabe, editors, *Mathematical Programming: Recent Developments and Applications*, pages 83–108. Kluwer Academic Publishers, Dordrecht, 1989.

[Gri90] Andreas Griewank. Direct calculation of Newton steps without accumulating Jacobians. In T. F. Coleman and Yuying Li, editors, *Large-Scale Numerical Optimization*, pages 115–137. SIAM, Philadelphia, Penn., 1990. Also appeared as Preprint MCS–P132–0290, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., February 1990.

[Gri91a] Andreas Griewank. Automatic evaluation of first- and higher-derivative vectors. In R. Seydel, F. W. Schneider, T. Küpper, and H. Troger, editors, *Proceedings of the Conference at Würzburg, Aug. 1990, Bifurcation and Chaos: Analysis, Algorithms, Applications*, volume 97, pages 135–148. Birkhäuser Verlag, Basel, Switzerland, 1991.

[Gri91b] Andreas Griewank. The chain rule revisited in scientific computing. *SIAM News*, 24, May & July 1991. no. 3, p. 20 & no. 4, p. 8.

[Gri91c] Andreas Griewank. Sequential evaluations of adjoints and higher derivative vectors by overloading and reverse accumulation. Preprint SC 91–3, Konrad-Zuse-Zentrum für Informationstechnik Berlin, July 1991.

[Gri93] Andreas Griewank. Some bounds on the complexity of gradients, jacobians and hessians. In P.M. Pardalos, editor, *Complexity in Nonlinear Optimization*, pages 128–161. World Scientific Publishers Co., 1993.

[Griar] Andreas Griewank. Achieving logarithmic growth of temporal and spatial complexity in reverse automatic differentiation. *Optimization Methods and Software*, to appear. Also appeared as Preprint MCS–P228–0491, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., April 1991.

[Hor92] Jim E. Horwedel. Reverse automatic differentiation of modular fortran programs. Technical Memorandum ORNL/TM 12050, Computing and Telecommunications Division, Oak Ridge National Laboratory, Oak Ridge, Tenn., March 1992.

[IK87] Masao Iri and K. Kubota. Methods of fast automatic differentiation and applications. Research Memorandum RMI 87 – 02, Department of Mathematical Engineering and Information Physics, Faculty of Engineering, University of Tokyo, 1987.

[Iri91] Masao Iri. History of automatic differentiation and rounding estimation. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 1–16. SIAM, Philadelphia, Penn., 1991.

[Jer90] M. Jerrell. Automatic differentiation using C++. *J. Object Oriented Programming*, pages 17–24, 1990.

[JG90] David Juedes and Andreas Griewank. Implementing automatic differentiation efficiently. Technical Memorandum ANL/MCS–TM–140, Mathematics and Computer Sciences Division, Argonne National Laboratory, Argonne, Ill., 1990.

[JM88] R.H.F. Jackson and G.P. McCormick. Second order sensitivity analysis in factorable programming: theory and applications. *Mathematical programming*, 41(1):1–28, 1988.

[Jue91] David Juedes. A taxonomy of automatic differentiation tools. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 315–329. SIAM, Philadelphia, Penn., 1991.

[Ked80] G. Kedem. Automatic differentiation of computer programs. *ACM Trans. Math. Software*, 6(2):150–165, June 1980.

[KI88] Koichi Kubota and Masao Iri. Formulation of fast automatic differentiation and the analysis of its complexity. *Transactions of Information Processing Society of Japan*, 29:551–560, 1988. (In Japanese).

[KT86] R. Kalaba and Leigh Tesfatsion. Automatic differentiation of functions of derivatives. *Computers and Mathematics with Applications*, 12A(11):1091–1103, November 1986.

[Law91] Charles L. Lawson. Automatic differentiation of inverse functions. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 87–94. SIAM, Philadelphia, Penn., 1991.

[MK89] K. Murota and Koichi Kubota. On elimination of intermediate variables in fast automatic differentiation. *Transactions of Information Processing Society of Japan*, 30:536–539, 1989. (In Japanese).

[Mor84] J. Morgenstern. How to compute fast a function and all its derivatives. A variation on the theorem of Baur-Strassen. Report No. 49, Laboratoire CNRS 168, Université de Nice, Nice, France, 1984.

[Par90] S. C. Parkhurst. The evaluation of exact numerical Jacobians using automatic differentiation. Technical Report NOC TR224, The Numerical Optimisation Center, Hatfield Polytechnic, Hatfield, U.K., December 1990.

[Ral91] Louis B. Rall. Point and interval differentiation arithmetics. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 17–24. SIAM, Philadelphia, Penn., 1991.

[Sou91] Edgar J. Soulié. User's experience with Fortran precompilers for least squares optimization problems. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 297–306. SIAM, Philadelphia, Penn., 1991.

[Spe80] B. Speelpenning. *Compiling Fast Partial Derivatives of Functions Given by Algorithms.* PhD thesis, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana-Champaign, Ill., January 1980.

[Tha91] William Carlisle Thacker. Automatic differentiation from an oceanographer's perspective. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 191–201. SIAM, Philadelphia, Penn., 1991.

[Wex87] Anthony S. Wexler. Automatic evaluation of derivatives. *Applied Mathematics and Computation*, 24:19–46, 1987.

[Wex88] Anthony S. Wexler. An algorithm for exact evaluation of multivariate functions and their derivatives to any order. *Computational Statistics and Data Analysis*, 6:1–6, 1988.

[Yos19] T. Yoshida. Derivation of a computational process for partial derivatives of functions using transformations of a graph. *Transactions of Information Processing Society of Japan*, 11:1112–1120, 19.

Automatic Differentiation – Applications

References

[BCG+93] Christian Bischof, George Corliss, Larry Green, Andreas Griewank, K. Haigler, and Perry Newman. Automatic differentiation of advanced CFD codes for multidisciplinary design. Preprint MCS–P339–1192, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., January 1993.

[Bernt] Martin Berz. Automatic differentiation as an application of nonarchimedean analysis. *IMACS Annals of Computing and Applied Mathematics*, in print.

[Car86] Bradley R. Carlile. Solution of nonlinear systems of equations on the FPS 64-bit family of scientific computers using automatic differentiation. In *Proceedings of the 1986 Array Conference (Portland, Oregon)*, pages 142–169, 1986.

[CGRW92] George Corliss, Andreas Griewank, Tom Robey, and Steve Wright. Automatic differentiation applied to unsaturated flow — ADOL–C case study. Technical Memorandum ANL/MCS–TM–162, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., April 1992.

[CMZ92] Stephen L. Campbell, Edward Moore, and Yangchun Zhong. Utilization of automatic differentiation in control algorithms. Preprint for presentation at the 31st IEEE Conference on Decision and Control to be held December 16–18, 1992, in Tucson., 1992.

[CMZ94] Stephen L. Campbell, Edward Moore, and Yangchun Zhong. Utilization of automatic differentiation in control algorithms. *IEEE Transactions on Automatic Control*, 39(5):1047–1052, 1994.

[Dix87] Lawrence C. W. Dixon. Automatic differentiation and parallel processing in optimisation. Technical Report No. 180, The Numerical Optimisation Center, Hatfield Polytechnic, Hatfield, U.K., 1987.

[DMM88] Lawrence C. W. Dixon, Z. Maany, and M. Mohseninia. Finite element optimization in Ada using automatic differentiation. Technical Report NOC TR205, The Numerical Optimisation Center, Hatfield Polytechnic, Hatfield, U.K., 1988.

[Evt91] Yuri G. Evtushenko. Automatic differentiation viewed from optimal control. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 25–30. SIAM, Philadelphia, Penn., 1991.

[Fisar] Herbert Fischer. Automatic differentiation of the vector that solves a parametric linear system. *J. Computational and Applied Mathematics*, 35, to appear.

[ITH88] Masao Iri, T. Tsuchiya, and M. Hoshi. Automatic computation of partial derivatives and rounding error estimates with applications to large-scale systems of nonlinear equations. *J. Computational and Applied Mathematics*, 24:365–392, 1988. Original Japanese version appeared in *J. Information Processing*, 26 (1985), pp. 1411–1420.

[KI91] Koichi Kubota and Masao Iri. Estimates of rounding errors with fast automatic differentiation and interval analysis. *Journal of Information Processing*, 14(4):508–515, 1991. English version of [Kubo89b].

[KL91] Dan Kalman and Robert Lindell. Automatic differentiation in astrodynamical modeling. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 228–243. SIAM, Philadelphia, Penn., 1991.

[KT84] R. Kalaba and A. Tischler. Automatic derivative evaluation in the optimization of nonlinear models. *The Review of Economics and Statistics*, 66:653–660, 1984.

[Lay91] J. Daniel Layne. Applying automatic differentiation and self-validating numerical methods in satellite simulations. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 211–217. SIAM, Philadelphia, Penn., 1991.

[Ral80] Louis B. Rall. Applications of software for automatic differentiation in numerical computation. In G. Alefeld and R. D. Grigorieff, editors, *Fundamentals of Numerical Computation (Computer Oriented Numerical Analysis)*, Computing Supplement No. 2, pages 141–156. Springer Verlag, Berlin, 1980.

[Ral81] Louis B. Rall. *Automatic Differentiation: Techniques and Applications*, volume 120 of *Lecture Notes in Computer Science*. Springer Verlag, Berlin, 1981.

[Ral85] Louis B. Rall. Global optimisation using automatic differentiation and interval arithmetic. Technical Summary Report No. 2832, Mathematics Research Center, University of Wisconsin - Madison, 1985.

[SBC91] Sirpa Saarinen, Randall Bramley, and George Cybenko. Neural networks, backpropagation, and automatic differentiation. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 31–42. SIAM, Philadelphia, Penn., 1991.

[Sha91] Piyush Shah. Application of adjoint equations to estimation of parameters in distributed dynamic systems. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 181–190. SIAM, Philadelphia, Penn., 1991.

[Tes91] Leigh Tesfatsion. Automatic evaluation of higher-order partial derivatives for nonlocal sensitivity analysis. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 157–165. SIAM, Philadelphia, Penn., 1991.

[Wor91] Brian Worley. Experience with the forward and reverse mode of GRESS in contaminant transport modeling and other applications. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 307–315. SIAM, Philadelphia, Penn., 1991.

Automatic Differentiation – Implementations

References

[BCC+91] Christian Bischof, Alan Carle, George Corliss, Andreas Griewank, and Paul Hovland. ADIFOR: Fortran source translation for efficient derivatives. Preprint MCS–P278–1291, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., December 1991. ADIFOR Working Note # 4.

[BCC+92a] Christian Bischof, Alan Carle, George Corliss, Moe El-Khadiri, Paul Hovland, and Andreas Griewank. Getting started with ADIFOR. Technical Memorandum ANL/MCS–TM–164, Mathematics and Computer Science Division,

Argonne National Laboratory, Argonne, Ill., 1992. ADIFOR Working Note # 9.

[BCC+92b] Christian Bischof, Alan Carle, George Corliss, Andreas Griewank, and Paul Hovland. ADIFOR – Generating derivative codes from Fortran programs. *Scientific Programming*, 1(1):11–29, 1992.

[BCCG92] Christian Bischof, Alan Carle, George Corliss, and Andreas Griewank. ADIFOR: Automatic differentiation in a source translation environment. Preprint MCS–P288–0192, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., January 1992. ADIFOR Working Note # 5. Accepted for the International Symposium on Symbolic and Algebraic Computation, July 27–29, 1992, Berkeley, Calif.

[BCG92a] Christian Bischof, George Corliss, and Andreas Griewank. ADIFOR exception handling. Technical Memorandum ANL/MCS–TM–159, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., January 1992. ADIFOR Working Note # 3.

[BCG92b] Christian Bischof, George Corliss, and Andreas Griewank. Hybrid evaluation of second derivatives in ADIFOR. Technical Memorandum ANL/MCS–TM–166, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., May 1992. ADIFOR Working Note # 8.

[BEK92] Christian Bischof and Moe El-Khadiri. On exploiting partial separability and extending the compile-time reverse mode in ADIFOR. Technical Memorandum ANL/MCS–TM–163, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., 1992. ADIFOR Working Note # 7.

[BH91] Christian Bischof and Paul Hovland. Using ADIFOR to compute dense and sparse Jacobians. Technical Memorandum ANL/MCS–TM–158, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., October 1991. ADIFOR Working Note # 2.

[CR84] George F. Corliss and Louis B. Rall. Automatic generation of Taylor series in Pascal-SC: Basic operations and applications to differential equations. In *Trans. of the First Army Conference on Applied Mathematics and Computing (Washington, D.C., 1983)*, pages 177–209. ARO Rep. 84-1, U. S. Army Res. Office, Research Triangle Park, N.C., 1984.

[DP87] Paul H. Davis and John D. Pryce. A new implementation of automatic differentiation for use with numerical software. Technical Report AM–87–11, School of Mathematics, University of Bristol, Bristol, U.K., 1987.

[Gar91] Oscar García. A system for the differentiation of Fortran code and an application to parameter estimation in forest growth models. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 273–286. SIAM, Philadelphia, Penn., 1991.

[GJSTar] Andreas Griewank, David Juedes, Jay Srinivasan, and Charles Tyner. ADOL-C, a package for the automatic differentiation of algorithms written in

C/C++. *ACM Trans. Math. Software*, to appear. Also appeared as Preprint MCS–P180–1190, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., November 1990.

[Hil82] Kenneth E. Hillstrom. JAKEF – A portable symbolic differentiator of functions given by algorithms. Technical Report ANL–82–48, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., 1982.

[Hil85] K. E. Hillstrom. Users guide for JAKEF. Technical Memorandum ANL/MCS–TM–16, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Ill., 1985.

[Hor91] Jim E. Horwedel. GRESS: A preprocessor for sensitivity studies on Fortran programs. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 243–250. SIAM, Philadelphia, Penn., 1991.

[HR92] David R. Hill and Lawrence C. Rich. Automatic differentiation in MATLAB. *Applied Numerical Mathematics*, 9:33–43, 1992.

[Hus90] R. Huss. An Ada library for automatic evaluation of derivatives. *Applied Mathematics and Computation*, 35:103–123, January 1990. Also appeared as Working Paper, Hughes Aircraft Company, February 1989.

[HWOP88] Jim E. Horwedel, Brian A. Worley, E. M. Oblow, and F. G. Pin. GRESS version 1.0 users manual. Technical Memorandum ORNL/TM 10835, Martin Marietta Energy Systems, Inc., Oak Ridge National Laboratory, Oak Ridge, Tenn., 1988.

[KI90] Koichi Kubota and Masao Iri. PADRE2, version 1 — User's manual. Research Memorandum RMI 90–01, Department of Mathematical Engineering and Information Physics, Faculty of Engineering, University of Tokyo, 1990.

[Kub88] Koichi Kubota. A preprocessor for fast automatic differentiation — Applications and difficulties on practical problems. In *RIMS Kokyuroku 648 "Fundamental Numerical Algorithms and their Software"*. Research Institute for Mathematical Sciences, Kyoto University, 1988. (In Japanese).

[Kub91] Koichi Kubota. PADRE2, a FORTRAN precompiler yielding error estimates and second derivatives. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 251–262. SIAM, Philadelphia, Penn., 1991.

[LS90] M. Liepel and K. Schittkowski. PCOMP: A FORTRAN code for automatic differentiation. Report No. 254, DFG Schwerpunktprogramm Anwendungsbezogene Optimierung und Optimale Steuerung, Mathematisches Institut, Universität Bayreuth, D-8580 Bayreuth, Germany, 1990.

[Maa89] Z. Maany. Ada automatic differentiation package for the optimization of functions of many variables. Technical Report NOC TR209, The Numerical Optimisation Center, Hatfield Polytechnic, Hatfield, U.K., July 1989.

[Maaar] Z. A. Maany. FORTRAN automatic differentiation package for the optimization of functions of many variables. to appear.

[Maz91] Vladimir Mazourik. Integration of automatic differentiation into a numerical library for PC's. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 286–293. SIAM, Philadelphia, Penn., 1991.

[Mic91] Leo Michelotti. MXYZPTLK: A C++ hacker's implementation of automatic differentiation. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 218–227. SIAM, Philadelphia, Penn., 1991.

[Obl85] E. M. Oblow. GRESS: Gradient-enhanced software system. Version D user's guide. Tech. Report, Oak Ridge National Laboratory, Oak Ridge, Tenn., 1985.

[PD87] John D. Pryce and Paul H. Davis. A new implementation of automatic differentiation for use with numerical software. Technical Report TR AM-87-11, Mathematics Department, Bristol University, 1987.

[Pfe87] F. W. Pfeiffer. Automatic differentiation in PROSE. *ACM SIGNUM Newsletter*, 22(1):1–8, 1987.

[PS91] John D. Pryce and Bruce R. Stephens. The DAPRE preprocessor users' guide. Technical Report ACM–91–3, Royal Military College of Science at Shrivenham, Shrivenham, U.K., 1991.

[Rei67] Alan Reiter. Automatic generation of Taylor coefficients (TAYLOR) for the CDC 1604. Technical Summary Report No. 830, Mathematics Research Center, University of Wisconsin - Madison, 1967.

[SP90] Bruce R. Stephens and John D. Pryce. *The DAPRE/UNIX Preprocessor Users' Guide v1.2*. Royal Military College of Science at Shrivenham, Shrivenham, U.K., 1990.

[SP91] Bruce R. Stephens and John D. Pryce. DAPRE: A differentiation arithmetic system for FORTRAN. Technical Report ACM–91–3, Royal Military College of Science, Shrivenham, U.K., 1991.

[Wen64] R. E. Wengert. A simple automatic derivative evaluation program. *Comm. ACM*, 7(8):463–464, 1964.

Adjoint modelling

References

[CC90] W. C. Chao and L. P. Chang. Status of the development of a variational data assimilation system using the adjoint method at Goddard Laboratory for Atmospheres. In *Proceedings of the International Symposium on Assimilation of Observations in Meteorology and Oceanography, World Meteorological Organization, Geneva, Switzerland*, pages 355–358, 1990.

[Cou85] P. Courtier. Experiments in data assimilation using the adjoint model technique. In *Proceedings of the Workshop on High-Resolution Analysis ECMWF (UK)*, June 1985.

[CT87] P. Courtier and Olivier Talagrand. Variational assimilation of meteorological observations with the adjoint equation – Part II. Numerical results. *Q. J. R. Meteorol. Soc.*, 113:1329–1347, 1987.

[CT90] P. Courtier and Olivier Talagrand. Variational assimilation of meteorological observations with the direct and adjoint shallow-water equations. *Tellus*, 42A:531–549, 1990.

[CTT90] P. Courtier, J. N. Thepaut, and Olivier Talagrand. 4-dimensional data assimilation using the adjoint of a primitive equation model. In *Proceedings of the International Symposium on Assimilation of Observations in Meteorology and Oceanography, World Meteorological Organization, Geneva, Switzerland*, pages 337–340, 1990.

[DT90] D. Douady and Olivier Talagrand. The impact of threshold processes on variational assimilation. In *Proceedings of the International Symposium on Assimilation of Observations in Meteorology and Oceanography, World Meteorological Organization, Geneva, Switzerland*, pages 486–487, 1990.

[Gri91] Andreas Griewank. Sequential evaluations of adjoints and higher derivative vectors by overloading and reverse accumulation. Preprint SC 91–3, Konrad-Zuse-Zentrum für Informationstechnik Berlin, July 1991.

[HCS82] M. C. G. Hall, D. G. Cacuci, and M. E. Schlesinger. Sensitivity analysis of a radiative-convective model by the adjoint method. *J. Atmos. Sci.*, 39:2038–2050, 1982.

[LD85] J. M. Lewis and J. C. Derber. The use of adjoint equations to solve a variational adjustment problem with advective constraints. *Tellus*, 37A:309–322, 1985.

[LN86] F.-X. Le Dimet and A. Nouailler. Assimilation of dynamic data in a limited-area model. In Y. K. Sasaki, editor, *Variational Methods in Geosciences*, pages 181–198. Elsevier, Amsterdam, 1986.

[LT86] F.-X. Le Dimet and Oliver Talagrand. Variational algorithms for analysis and assimilation of meteorological observations: Theoretical aspects. *Tellus*, 38A:97–110, 1986.

[NZ91] I. Michael Navon and Xiaolei Zou. Application of the adjoint model in meteorology. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 202–207. SIAM, Philadelphia, Penn., 1991.

[NZJ+90] I. Michael Navon, Xiaolei Zou, K. Johnson, J. Derber, and J. Sela. Variational data assimilation with an adiabatic version of the NMC spectral model. *Monthly Weather Review*, 1990.

[RN90] M. K. Ramamurthy and I. Michael Navon. Application of a conjugate-gradient method to variational assimilation of meteorological fields. In Oliver Talagrand and F.-X. Le Dimet, editors, *Proceedings of the International Symposium on Assimilation of Observations in Meteorology and Oceanography, World Meteorological Organization, Geneva, Switzerland*, pages 359–365, Geneva, 1990. World Meteorological Organization.

[Sha91] Piyush Shah. Application of adjoint equations to estimation of parameters in distributed dynamic systems. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 181–190. SIAM, Philadelphia, Penn., 1991.

[Tal91] Oliver Talagrand. The use of adjoint equations in numerical modelling of the atmospheric circulation. In Andreas Griewank and George F. Corliss, editors, *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, pages 169–180. SIAM, Philadelphia, Penn., 1991.

[TC87] Olivier Talagrand and P. Courtier. Variational assimilation of meteorological observations with the adjoint vorticity equation – Part I. Theory. *Q. J. R. Meteorol. Soc.*, 113:1311–1328, 1987.