Optimization Methods and Software

ISSN: 1055-6788 (Print) 1029-4937 (Online) Journal homepage: https://www.tandfonline.com/loi/goms20

AD Model Builder: using automatic differentiation for statistical inference of highly parameterized complex nonlinear models

David A. Fournier , Hans J. Skaug , Johnoel Ancheta , James Ianelli , Arni Magnusson , Mark N. Maunder , Anders Nielsen & John Sibert

To cite this article: David A. Fournier , Hans J. Skaug , Johnoel Ancheta , James Ianelli , Arni Magnusson , Mark N. Maunder , Anders Nielsen & John Sibert (2012) AD Model Builder: using automatic differentiation for statistical inference of highly parameterized complex nonlinear models, Optimization Methods and Software, 27:2, 233-249, DOI: 10.1080/10556788.2011.597854

To link to this article: https://doi.org/10.1080/10556788.2011.597854

Optimization Methods & Software
Vol. 27, No. 2, April 2012, 233–249

Taylor & Francis
Taylor & Francis Group

AD Model Builder: using automatic differentiation for statistical inference of highly parameterized complex nonlinear models

David A. Fournier[a], Hans J. Skaug[b]*, Johnoel Ancheta[c], James Ianelli[d], Arni Magnusson[e], Mark N. Maunder[f], Anders Nielsen[g] and John Sibert[c]

[a]*Otter Research Ltd., Sidney, Canada;* [b]*Department of Mathematics, University of Bergen, Norway;* [c]*Joint Institute for Marine and Atmospheric Research, University of Hawaii at Mānoa, USA;* [d]*REFM Division, Alaska Fisheries Science Center, NOAA, Seattle, WA, USA;* [e]*Marine Research Institute, Reykjavik, Iceland;* [f]*Inter-American Tropical Tuna Commission, La Jolla, CA, USA;* [g]*Tech. Univ. Denmark, Natl. Inst. Aquat. Resources, Charlottenlund, Denmark*

(*Received 27 September 2010; final version received 10 June 2011*)

Many criteria for statistical parameter estimation, such as maximum likelihood, are formulated as a nonlinear optimization problem. Automatic Differentiation Model Builder (ADMB) is a programming framework based on automatic differentiation, aimed at highly nonlinear models with a large number of parameters. The benefits of using AD are computational efficiency and high numerical accuracy, both crucial in many practical problems. We describe the basic components and the underlying philosophy of ADMB, with an emphasis on functionality found in no other statistical software. One example of such a feature is the generic implementation of Laplace approximation of high-dimensional integrals for use in latent variable models. We also review the literature in which ADMB has been used, and discuss future development of ADMB as an open source project. Overall, the main advantages of ADMB are flexibility, speed, precision, stability and built-in methods to quantify uncertainty.

Keywords: ADMB; automatic differentiation; parameter estimation; optimization; Laplace approximation; separability

1. Introduction

Answering real-world management and research questions using data typically requires complex nonlinear models to adequately represent the system under study. Historically, simplifying assumptions were made to allow explicit expression for estimates of model parameters and their uncertainty based on the available data. With the advent of computers and numerical procedures these simplifications are no longer needed and the analysts can apply the model that they consider most appropriate. Parameters are often estimated by optimizing an objective function (e.g. likelihood) that measures how well the model predicts the data. Efficient methods for optimizing the objective function are based on derivatives, which typically are approximated numerically using finite differences, or alternatively by evaluating an analytical expression for the derivatives. The former method can be unstable, and is inefficient when there are many parameters. Analytical

*Corresponding author. Email: skaug@math.uib.no

ISSN 1055-6788 print/ISSN 1029-4937 online

http://dx.doi.org/10.1080/10556788.2011.597854
http://www.tandfonline.com

derivatives are preferable, but may be hard to derive for complex models implemented as thousands of lines of computer code. A third alternative is provided by automatic differentiation (AD), which is a technique for numerical evaluation of derivatives of a mathematical function available in the form of a computer program [22,23].

Many software packages and libraries can perform AD on user-supplied programs written in standard languages such as FORTRAN and C++ [6,24,26]. These packages, however, focus only on derivative calculations; the user must incorporate the resulting derivative code into applications such as a function minimization problems or sensitivity analysis. More specialized mathematical programming environments such as MATLAB and NAG have also incorporated AD. Similar functionality is, however, very limited in statistical softwares such as R, SAS, SPSS and STATA. This is somewhat surprising, since the method of least-squares and likelihood-based parameter estimation, which are commonly used in statistical inference, involve minimizing an objective function. The availability of accurate derivative information thus seems crucial in all but trivial estimation problems. For many models found in statistical packages, the derivatives of the objective function have been hand-coded. In settings where the user specifies the objective function, as opposed to choosing from prefixed options, it seems that AD has an important role to play. Derivatives are also useful for many other components of statistical inference.

In this paper, we describe AD Model Builder (ADMB), an AD-based framework for doing statistical parameter estimation, that includes a complete suite of tools for developing nonlinear statistical models. ADMB seamlessly integrates AD with a function minimizer which is used to minimize an objective function (e.g. a negative log-likelihood function). ADMB has functionality not found in other statistical software, e.g. profile likelihood and Laplace approximations of high-dimensional integrals. In both of these, AD plays a crucial role. Besides profile likelihood and the delta method, all ADMB models have built-in capability to use Markov-chain Monte Carlo (MCMC) analysis to evaluate the uncertainty of estimated quantities.

Originally developed by David A. Fournier in the late 1980s, ADMB was designed for fish stock assessments with hundreds of parameters and highly nonlinear objective functions. Its use has since spread within the broader ecological community and other scientific fields. ADMB is today available as a free open-source software, see `http://www.admb-project.org`. This paper gives an overview of ADMB as a model-building tool, and is intended to be the main ADMB reference.

The rest of this paper is organized as follows. Section 2 explains the basic working of ADMB with an emphasis on the role played by AD, Section 3 presents the use of the Laplace approximation, Section 4 presents an overview of studies in which ADMB has been used, Section 5 presents the open source project and Section 6 is a summary and discussion. Readers less interested in the technical details and wanting an overview of ADMB can skip Sections 2.1–2.3, and all of Section 3.

2. What is AD Model Builder?

From the user's point of view, AD Model Builder is similar to a fourth-generation computer language. The ADMB source code contains shorthand declarations of data and of specialized types supporting AD along with C++ code defining the user's model. The ADMB source code is converted to C++ by the ADMB pre-processor, and the resulting C++ code is then compiled and linked with the AD library to create an executable program (see Figure 1).

These steps are automated in scripts and in an integrated development environment (IDE). When run, the executable program produces parameter estimates, estimates of standard deviations, and other quantities useful for statistical inference. Here, we provide a simple example for the benefit of the user without previous exposure to AD before proceeding with a more complete description of ADMB.

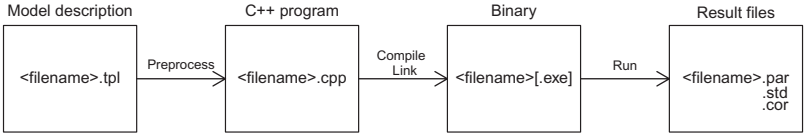

Figure 1. The different steps starting with the model specification and ending with the estimated model parameters and uncertainties. The preprocessing, compiling and linking is typically collected in a single command.

2.1 *A simple example*

Consider a simple regression model,

$$y_i = a + bx_i + \varepsilon_i, \tag{1}$$

for a sequence of n experiments. Here, y_i is the response of the ith experiment, x_i the explanatory variable and ε_i the zero-mean measurement error. Table 1 shows the ADMB code for this model, using concentrated negative log-likelihood as the objective function.

The simplest least-squares approach, found in many textbooks, is based on the objective function

$$S = -\sum_{i=1}^{n} \{y_i - (a + bx_i)\}^2. \tag{2}$$

The function is optimized when the derivative with respect to the parameter being estimated is equal to zero. Therefore, the least squares estimates of a and b are found by solving

$$\begin{aligned} \frac{\partial S}{\partial a} &= \sum_{i=1}^{n} 2\{y_i - (a + bx_i)\} = 0, \\ \frac{\partial S}{\partial b} &= \sum_{i=1}^{n} 2x_i\{y_i - (a + bx_i)\} = 0. \end{aligned} \tag{3}$$

ADMB accomplishes the same thing numerically for any differentiable objective function. The AD routines compute the required derivatives, and the function minimizer decreases $-S$ until the derivatives are arbitrarily close to zero (by default $<10^{-4}$).

The AD algorithms in ADMB take advantage of the fact that C++ compilers reduce even the most complex statement into a sequence of elemental unary or binary operations on floating point representations of real numbers. At the time that these operations are executed, all of the

Table 1. ADMB listing for a simple linear regression model.

```
DATA_SECTION
  init_int n
  init_vector x(1,n)
  init_vector y(1,n)
PARAMETER_SECTION
  init_number a
  init_number b
  vector yfit(1,n)
  objective_function_value f
PROCEDURE_SECTION
  yfit=a+b*x;
  f=0.5*n*log(norm2(y-yfit));
```

Table 2. Simple example of automatic differentiation.

Step	Operation	Value	Derivatives
1	$t_1 = bx_i$	bx_i	$\frac{\partial t_1}{\partial b} = x_i$
			$\frac{\partial t_1}{\partial x_i} = b$
2	$t_2 = a + t_1$	$a + bx_i$	$\frac{\partial t_2}{\partial a} = 1$
			$\frac{\partial t_2}{\partial t_1} = 1$
3	$t_3 = y_i - t_2$	$y_i - (a + bx_i)$	$\frac{\partial t_3}{\partial y} = 1$
			$\frac{\partial t_3}{\partial t_2} = -1$
4	$t_4 = t_3^2$	$\{y_i - (a + bx_i)\}^2$	$\frac{\partial t_4}{\partial t_3} = 2t_3$
5	$S_i = t_4$	$\{y_i - (a + bx_i)\}^2$	$\frac{\partial S_i}{\partial t_4} = 1$

Note: Each term (2) is broken down into elemental operations, and the chain rule of differentiaton is applied.

information required to compute the derivatives is available for use. For example, one possible way for a compiler to parse the ith term in (2) into elemental operations is shown in Table 2.

The partial derivatives in the above table can be combined using the chain rule:

$$\frac{\partial S_i}{\partial a} = \frac{\partial t_2}{\partial a}\frac{\partial t_3}{\partial t_2}\frac{\partial t_4}{\partial t_3}\frac{\partial S_i}{\partial t_4} = 2\{y_i - (a + bx_i)\}, \tag{4}$$

$$\frac{\partial S_i}{\partial b} = \frac{\partial t_1}{\partial b}\frac{\partial t_2}{\partial t_1}\frac{\partial t_3}{\partial t_2}\frac{\partial t_4}{\partial t_3}\frac{\partial S_i}{\partial t_4} = 2x_i\{y_i - (a + bx_i)\}. \tag{5}$$

2.2 *Implementation of AD*

There exist two AD strategies: the forward mode and the reverse mode. In the forward mode, derivatives of temporary variables with respect to the independent variables (a and b in Section 2.1) are propagated in parallel with the general program flow. This amounts to sequential accumulation of the products (4) and (5). The forward mode is the intuitive way of organizing the calculations for a person with a traditional calculus background. It, however, has the drawback that the cost of propagating the derivatives grows in proportion to the number of independent variables. This is limiting in problems with many estimated parameters. For the reverse mode, on the other hand, there is the celebrated result that the full gradient of the objective function can be obtained at a cost of no more than four times the cost of evaluating the objective function [22, p. 57]. For this reason, ADMB uses the reverse mode as a general strategy for calculating first-order derivatives.

The reverse mode involves first evaluating the objective function, storing in memory the value of each intermediate quantity, that is $t_1, \ldots, t_4$ in the program listing of Table 2. This is known as the 'forward sweep', in which no derivatives are calculated. The derivatives in the rightmost column of Table 2 are evaluated as part of the 'reverse sweep', where the main goal is to find the derivative of the output variable with respect to each intermediate variable, resulting in a sequence of 'sensitivities': $\partial S/\partial t_4$, $\partial S/\partial t_3$, $\partial S/\partial t_2$ and $\partial S/\partial t_1$. Each step involves the chain rule, and the reverse order of the calculations is essential. Now, the gradient is easily extracted:

$$\frac{\partial S}{\partial a} = \frac{\partial S}{\partial t_2}\frac{\partial t_2}{\partial a}, \quad \frac{\partial S}{\partial b} = \frac{\partial S}{\partial t_1}\frac{\partial t_1}{\partial b}.$$

For a full description of the reverse mode the reader is referred to [22].

The drawback of the reverse mode is the need to store in memory values of the temporary variables ($t_1, \ldots, t_4$) calculated during the forward sweep. For a simple program listing as in Table 2

this is not a problem, but for programs containing loops that unfold into long lists, the computational graph may become too large to fit in the physical memory, and hence must be written to a file. Writing to file greatly slows down the execution of the program. To counteract this phenomenon ADMB applies various techniques collectively known as pre-accumulation [22, p. 185].

ADMB implements reverse mode AD by overloading C++ operators [22, p. 93]. With this strategy the chain rule is applied at run time (of the executable program), as opposed to generating derivative code at compilation time. The latter technique is known as 'source transformation' in the AD literature. The internal part of ADMB, that overloads all the mathematical operators and functions in C++ and sets up the data structures needed for the reverse sweep, is a library known as AUTODIF [2].

AUTODIF also provides 'adjoint code' for most standard arithmetic operations involving numbers, vectors and matrices, and combinations of these. The use of adjoint code avoids the need to store the value of each elemental operation during the forward sweep. As an example, take the Cholesky decomposition `chol(X)` of a matrix `X`. During the forward sweep, only the value of the elements of `X` are stored, not those internal to the Cholesky algorithm. During the reverse sweep, `X` is restored and `chol(X)` is calculated again, this time with a code that saves all intermediate quantities within the Cholesky algorithm. Using this information, a 'local' reverse sweep of the Cholesky code is performed. (This approach is very similar to what is known as 'checkpointing' in the AD literature [22, p. 319].) Unlike the general reverse sweep, which is implemented using operator overloading in C++, this local reverse sweep is performed by a hand-coded algorithm, which is typically able to reduce the total storage requirements by overwriting variables no longer in use. Details of implementation are given in [60] in the case of the Cholesky algorithm. It is further shown in [60] that the adjoint code (including both forward and reverse sweeps) for `chol(X)` can be implemented using only the storage allocated for `X`. What is referred to above as 'hand-coded' derivative code could have been generated automatically using a source transformation tool such as TAPENADE [26], but this is not currently part of ADMB. The existence of adjoint code for common functions is a key part of ADMB, and an important reason why it can be applied to large real-world problems.

2.3 *Parameter estimation and uncertainty*

Although the method of least squares provides a criterion for obtaining point estimates of parameters in many situations, most of the features of ADMB require that we adopt a likelihood framework based on a full probabilistic formulation of the model. For the simple regression (1) this amounts to assuming that the measurement error is normally distributed with mean 0 and variance σ^2, written as $\varepsilon_i \sim N(0, \sigma^2)$. Now the log likelihood is given as

$$l(a, b, \sigma) = \sum_{i=1}^{n} \log\{p(y_i)\} = \sum_{i=1}^{n} \left\{ -\log(\sqrt{2\pi}\sigma) - \frac{\{y_i - (a + bu_i)\}^2}{2\sigma^2} \right\}, \tag{6}$$

where $p(y_i)$ denotes a probability density. ADMB will maximize $l(a, b, \sigma)$ using a quasi-Newton algorithm with derivatives obtained using AD (For historical reasons ADMB actually minimizes the objective function so in practice the negative log-likelihood is used). Once convergence has been reached, ADMB optionally calculates the covariance matrix by inverting the observed Fisher information (Hessian of l), $-(\partial^2/\partial\theta^2)l(\theta)$, where $\theta = (a, b, \sigma)$. The second-order derivatives are obtained by applying finite differences to $-(\partial/\partial\theta)l(\theta)$, which itself is obtained by AD. The use of finite differences in this context is much less problematic than it is for maximizing $l(\theta)$, as the Hessian only provides approximate standard deviations, with the approximation error typically being larger than the numerical error arising from the use of finite differences.

2.3.1 *Profile likelihood*

The profile likelihood feature of ADMB lets one study the log-likelihood surface for one parameter at a time, while maximizing with respect to the remaining parameters. ADMB also allows profiling with respect to a function $\phi = \phi(a, b, \sigma)$ of the parameters. From a computational perspective this leads us to the constrained optimization problem:

$$l(\phi_0) = \underset{a,b,\sigma}{\text{argmax}}, l(a, b, \sigma); \quad \text{subject to } \phi(a, b, \sigma) = \phi_0.$$

This is particularly useful when ϕ is an 'interest parameter', while the remaining parameters of model are nuisance parameters. The profile likelihood curve $l(\phi)$ is a graphical display of the information contained in the data about the interest parameters.

2.3.2 *Markov chain Monte Carlo*

During the 1990s, Markov chain Monte Carlo (MCMC) techniques started to have a large impact on the way statistics was practiced, particularly on Bayesian statistics [21]. The basic MCMC algorithm implemented in ADMB is a random walk Metropolis–Hastings algorithm [21, p. 289], where the initial point is the mode of the specified objective function, and the proposal covariance is the inverse Hessian, both calculated using AD as described previously. The literature of ADMB applications shows that the MCMC features are widely used. ADMB also has an MCMC routine that integrates over some parameters using Laplace approximation (see Section 3) and a hybrid MCMC [47] that both take additional advantage of AD to improve the MCMC performance.

2.4 *Model specification*

ADMB partitions the model specification into three logical steps: (1) read in the data (x's and y's in the simple regression); (2) declare model parameters (a and b), possibly with boundaries and a specification of the order in which each parameter becomes 'active' in the optimization process; and (3) code the negative log-likelihood function (essentially S) to be minimized with respect to the model parameters. This partitioning helps users organize and structure their model implementation and thus aids thinking about the problem at hand. The basic syntax is based on C++, but with many helpful features.

In the data section, common constant objects (number, vector, matrix, and many more) are declared and read in with a single command. For instance, to read in a 100 by 5 matrix A from the associated data file, the user would write:

```
init_matrix A(1,100,1,5)
```

This greatly simplifies the process of reading in data, and is especially helpful for users unfamiliar with C++.

The syntax for declaring parameters has a number of helpful features. Each of the different model parameter objects (numbers, vectors, matrices, and many more) can be declared by a single line, with optional bounds on the model parameter and an optimization phase. For instance, to declare a vector with five model parameters each between 0 and 1, but to be estimated only after another set of model parameters is optimized, the user would write:

```
init_bounded_vector theta(1,5,0.0,1.0,2)
```

The first two numbers specify the range of valid indices for the parameter vector. The third and fourth arguments specify the bounded interval, and the final argument specifies the order in which

these model parameters are to be estimated. Model parameters assigned to phase 1 are estimated first, while other parameters are fixed at their initial values. After phase 1 converges, the optimizer starts phase 2, where parameters assigned to phases 1 and 2 are estimated, and so on, until all parameters have been estimated simultaneously in the final phase. Bounding and estimation in phases is especially helpful in large (say >20 model parameters) nonlinear optimization problems. Bounding can prevent the optimization algorithm from searching among parameters where the objective function is not well defined. Optimization in phases can be helpful in getting model parameters with no obvious initial values into the right domain before the full optimization is started. For instance, if a good initial value is known for parameter ϑ but not for parameter τ, then it makes sense to set ϑ fixed at its initial value and optimize w.r.t. τ in the first phase, then – in second phase – optimize both, but initializing τ at its estimate from the first phase.

Unlike data and model parameters, which can be specified without any knowledge of C++, specification of the function to be minimized (typically a negative log-likelihood function) requires a basic understanding of C++ syntax. In addition to functions normally available in C++, AD Model Builder provides a variety of mathematical and statistical functions, such as vector and matrix operations, sums, probability density functions, etc. The user is required to code the function to be minimized, but the task is greatly simplified because of these built-in functions.

Compared to generic AD software, ADMB provides unique features useful for statistical applications. One feature allows the declaration of variables which are functions of a number of model parameters and are considered as 'derived' parameters (as opposed to 'fundamental' parameters that are directly estimated). The errors in the estimates of the 'fundamental' parameters are propagated through these 'derived' parameters when the standard deviation of all parameters are calculated. The following simple declaration will invoke computation of the standard deviation of the 'derived' parameters:

```
sdreport_number MyQuantity_of_interest
```

The variable `MyQuantity_of_interest` is then assigned a value in the model code as some function of model parameters, e.g. `MyQuantity_of_interest = a*b`. When the Hessian matrix and covariance matrix calculations are carried forward, ADMB tracks the Jacobian. The appropriate first-order Taylor-series propagation of errors (i.e. the 'delta method') is applied, and the asymptotic approximation of the variance–covariance matrix includes the 'derived' parameter.

Running a correctly specified executable will then maximize the coded likelihood function with respect to the model parameters. It will produce a number of output files containing, for example, the function value at the minimum, the estimated model parameters and the covariance matrix of the model parameters as estimated by the second derivatives of the function at the minimum.

2.5 *User interfaces*

The default ADMB user interface consists of a command-line program that translates the ADMB source code to C++ source code, along with shell scripts to simplify the process of translating, compiling and linking an application. Once built, model applications support a standard set of command line options [1] to specify optimization and other details. All input and output are in plain text files.

An integrated development environment called ADMB-IDE [36] is available, streamlining the installation and workflow between editor, compiler and debugger. A typical model-building session involves modifying and recompiling the code between test runs (Figure 2), but fully developed general models have been used for years without modification.

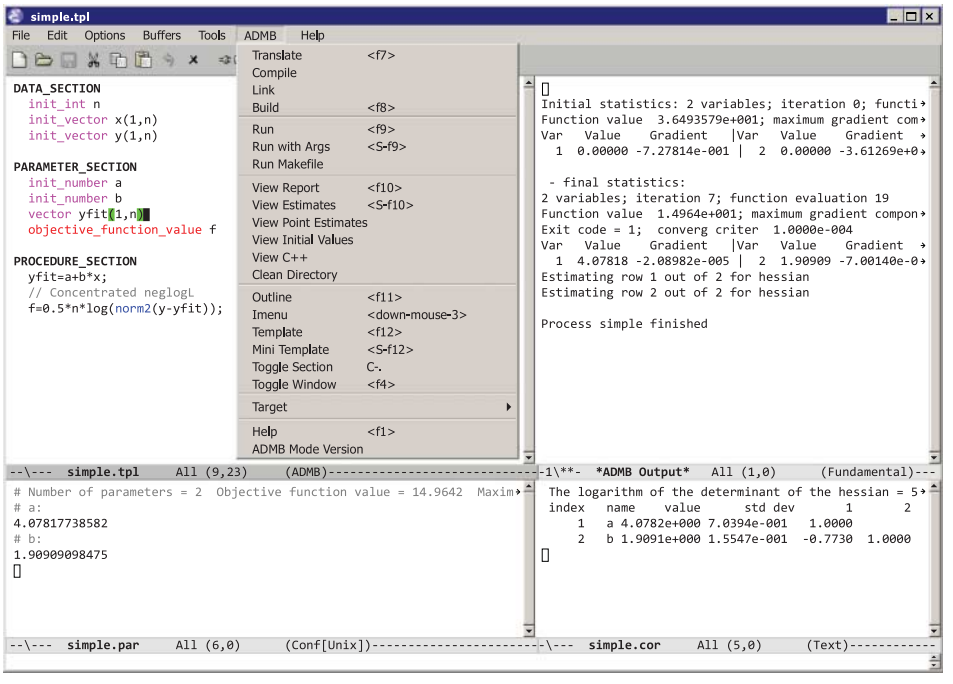

Figure 2. Typical AD Model Builder session, using ADMB-IDE. The top-left window shows the model code, demonstrating some of the sections and classes recognized by the ADMB-to-C++ translator. The menu includes commands to build a model, run and view the output. Other windows show point estimates, standard errors and correlation of estimated quantities.

Several extensions that interact with the R programming language have been developed [7,38, 55,62], and the process of converting models between ADMB and the BUGS (Bayesian inference Using Gibbs Sampling) programming language has been described [40].

3. Random effects

The ability to implement random effects is a recent addition to ADMB, using the Laplace approximation. This optional module, ADMB-RE, allows users to specify both random effects and regular model parameters for estimation. The task for this type of model is to find the maximum likelihood for all parameters while integrating over the random effects. The Laplace approximation of the marginal likelihood leads to a nested optimization problem and a need for calculating third-order derivatives by AD. This is done by successive application of forward and reverse mode ADs. This approach is described in [61]. Note that the term 'random effect' is defined more broadly in ADMB than in the statistical literature. In ADMB, random effects include not only regression coefficients that are random variables, but also state–space models, Markov random fields, and frailty models in survival analysis.

Analysts often split unknown model quantities into two categories: random effects and fixed effects. The former, which we denote by u, are thought of as arising from some random experiment or event, and hence there is an associated probability density $p(u)$. The fixed effects are what we have called 'parameters' up to now, and are non-random quantities which describe the 'population' from which u, together with data y, are generated. We denote the fixed effects by a vector θ. In the simple regression model of Section 2, we had $\theta = (a, b, \sigma^2)$ and no random effects. Models

are usually taken to be hierarchical: first we sample u from $p_\theta(u)$, and then we sample y from the conditional probability density $p_\theta(y|u)$. That is, in drawing the value of y, we use the realized value of u. ADMB treats θ and u differently, but before discussing computational aspects, we look at a simple example to clarify the concepts.

To make the simple regression a bit more complicated (and realistic) we shall consider the errors-in-variables problem: there is a measurement error associated with x_i, not just with y_i. If we disregard this error, then the estimate of b will be biased. We remedy this by introducing u_i, a version of x_i without a measurement error. The model now has two response variables, x and y, and is given by

$$\begin{aligned} y_i &= a + bu_i + \varepsilon_i, \\ x_i &= u_i + e_i, \end{aligned} \tag{7}$$

where $\varepsilon_i \sim N(0, \sigma^2)$ as above, and $e_i \sim N(0, \sigma_e^2)$ is the measurement error associated with x_i. To complete the probabilistic specification of the model, we assume that that $u_i \sim N(\mu, \sigma_u^2)$. The reason this assumption is needed is that the u_i's are unobserved.

The ADMB objective function is (the negative logarithm of) the joint probability density of the (u, x, y)'s:

$$\begin{aligned} \prod_{i=1}^{n} p_\theta(x_i|u_i)\, p_\theta(y_i|u_i)\, p_\theta(u_i) = \prod_{i=1}^{n} \Bigg[& \frac{1}{\sqrt{2\pi}\sigma_e} \exp\left(-\frac{(x_i - u_i)^2}{2\sigma_e^2}\right) \\ & \times \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\{y_i - (a + bu_i)\}^2}{2\sigma^2}\right) \\ & \times \frac{1}{\sqrt{2\pi}\sigma_u} \exp\left(-\frac{\{u_i - \mu\}^2}{2\sigma_u^2}\right)\Bigg]. \end{aligned} \tag{8}$$

The parameter vector has grown to $\theta = (a, b, \sigma^2, \sigma_x^2, \mu, \sigma_u^2)$. The fact that (8) is a product of terms, each depending only on a single u_i, has important computational implications, but is not a requirement imposed by ADMB. Models without this property include time-series models, with state variable u_i and observation y_i, for which the joint density is given by

$$\prod_{i=1}^{n} p_\theta(y_i|u_i)\, p_\theta(u_i|u_{i-1}). \tag{9}$$

Further, the distribution of y_i need not be Gaussian, but can be any other probability distribution, such as the Poisson, $p_\theta(y_i|u_i) = \lambda_i^{y_i}/(y_i!)\exp(-\lambda_i)$, where $\lambda_i = \exp(u_i)$. Similarly, the distribution of u_i need in principle not be Gaussian. However, the Laplace approximation discussed below works best if non-Gaussian random effects are obtained by transformation of an underlying Gaussian random variable u_i'. For example, if we want to obtain a log-normal random effect, we let $u = \exp(u')$, where $u' \sim N(0, 1)$. This trick adds another level to the model hierarchy. In summary, the requirement is that an expression is available for the joint density of y and u, and that this density is three times differentiable with respect to both u and θ (but not y).

A full explanation of the philosophical basis for viewing u as random and θ as non-random is beyond the scope of this paper. However, we note that our position is intermediate between that of the two competing traditions in statistics: Bayesianism (in which θ is also random) and frequentism (in which the u_i are non-random, but still unknown), and hence represents a reasonable compromise to many practitioners. The same philosophical distinction between fixed and random parameters is also found in other more narrowly targeted mixed model software.

3.1 *Parameter estimation via the Laplace approximation*

Parameters are estimated using a two-stage approach known as empirical Bayes [11]. First, θ is estimated by maximum likelihood, i.e., $\hat{\theta} = \arg\max_\theta L(\theta)$, where $L(\theta) = p_\theta(y)$ is the likelihood function. Then, a Bayesian posterior mode estimate is used for u:

$$\hat{u} = \underset{u}{\operatorname{argmax}}\, p_{\hat{\theta}}(y|u)\, p_{\hat{\theta}}(u).$$

The main computational challenge is to evaluate the marginal density of y,

$$p_\theta(y) = \int p_\theta(y, u)\, \mathrm{d}u, \tag{10}$$

which has no general analytical solution. Here, $p_\theta(y, u)$ is given by (8) and (9) or derived from some other model of interest. Because of the high dimension of u ($n = 1000$, say), ordinary quadrature rules for numerical integration cannot be used, and an alternative approximation is required. ADMB uses the Laplace approximation of the integral (10), which yields the likelihood approximation

$$L^*(\theta) = \det\{H(\theta)\}^{-1/2} \exp[g\{\hat{u}(\theta), \theta\}], \tag{11}$$

where

$$g(u, \theta) = \log\{p_\theta(y, u)\}, \tag{12}$$

$$\hat{u}(\theta) = \underset{u}{\operatorname{argmax}}\; g(u, \theta), \tag{13}$$

$$H(\theta) = -\frac{\partial^2}{\partial u^2}\, g(u, \theta)\bigg|_{u=\hat{u}(\theta)}. \tag{14}$$

For simplicity, we have suppressed the dependency of g on y. Note that without the term $\det(H)^{-1/2}$, the right-hand side of (11) is a profile likelihood (Section 2.3.1). ADMB evaluates the Hessian, H, by AD using a forward pass followed by a reverse pass. As pointed out in [61], the combination of the Laplace approximation and AD makes parameter estimation in random effects models automatic from a user perspective. The only responsibility of the ADMB user is to write down the correct expression for g, the joint density. Cholesky decomposition of H is used to evaluate $\det(H)$, and hence the Laplace approximation is only well defined in the part of θ space where $H(\theta)$ is positive definite. Using the mechanisms built into ADMB for doing constrained optimization, the user can ensure that the numerical search for the optimum of $L^*(\theta)$ is constrained to this region.

From a computational perspective, the Laplace approximation represents a nested optimization problem, where the inner level (12) must be solved for each value of θ that is requested by the outer level (11). For the purpose of maximizing (11), or in practice its logarithm (to increase numerical stability), it is advantageous to obtain the exact gradient of $L^*(\theta)$. To this end, the formula

$$\frac{\partial \hat{u}(\theta)}{\partial \theta} = -\{H(\theta)\}^{-1} \frac{\partial^2}{\partial u \partial \theta}\, g(\hat{u}, \theta)$$

has been noted in the literature [5,61]. The mixed partial derivatives of $g(u, \theta)$ are evaluated as part of the same process that evaluates H by AD. Further, by evaluating $\partial/\partial u\; g(u, \theta)$ and $\partial/\partial\theta\; g(u, \theta)$ by reverse mode AD, the gradient with respect to θ of the exponential term in (11) is obtained. Making analytical progress in evaluating the gradient of the remaining determinant term is possible, but ADMB uses an entirely AD-based approach. The expression $\partial\theta\, \det(H(\theta))$ involves

third-order mixed partial derivatives of $g(u, \theta)$. Second-order derivatives by AD are already involved in evaluating $\det(H(\theta))$. When traversing the computational graph of the computation $(u, \theta) \to H \to \det(H)$ in reverse, a third layer of AD is added.

One may view the Laplace approximation as a mapping: $(\theta, \hat{u}(\theta), H(\theta)) \to F(\theta, \hat{u}(\theta), H(\theta))$, where $F(\theta, u, H) = \det(H)^{-1/2} \exp\{g(u, \theta)\}$. Other integral approximations such as importance sampling and adaptive Gaussian quadrature are obtained for other choices of F [61], and are implemented in ADMB as (more accurate) alternatives to the Laplace approximation. Adaptive Gaussian quadrature is only applicable to low-dimensional integrals, but is nevertheless applicable to many regression models.

3.2 *Conditional independence and partial separability*

The above recipe for evaluating the Laplace approximation by AD is in principle applicable to any model for which a computer program can be written that evaluates $g(u, \theta)$. However, significant computational gains can be achieved in ADMB by exploiting the special structure of certain types of models. For instance, in model (8) the integral (10) becomes a product of n one-dimensional integrals, for which ADMB performs a series of n one-dimensional Laplace approximations (or optionally adaptive Gaussian quadrature). The state–space model (9) does not factorize completely, but instead becomes

$$g(u, \theta) = \sum_{i=2}^{n} g(u_i | u_{i-1}; \theta), \tag{15}$$

where $g(u_i|u_{i-1}; \theta) = \log\{p_\theta(y_i|u_i)\, p_\theta(u_i|u_{i-1})\}$. That each term in the sum depends only on two u_i terms is known in the AD literature as partial separability [20], and is exploited by ADMB both in the AD computations and when evaluating $\det(H)$, as H is a tridiagonal matrix.

Separability also has an important probabilistic interpretation in terms of conditional independence under the (Bayesian) posterior distribution

$$p_\theta(u|y) = \frac{p_\theta(y|u)\, p_\theta(u)}{p_\theta(y)}, \tag{16}$$

where $p_\theta(y)$ is given by (10). When element (i, j) of the Hessian H is zero, u_i and u_j become conditionally independent under (16), given the remaining u's. For the state–space model, with its banded H, we can make the more specific statement that for $i < j < k$, u_i and u_k are independent given u_j.

Understanding the link between conditional independence and separability is important for assessing the computational complexity of a given model. Further, there may be two different ways of implementing a model, one leading to a separable g, the other not. A simple example is the random walk, $u_i = u_{i-1} + e_i$, where e_i is an error term. The joint probability distribution of the u's is now $p(u_1) \prod_{i=2}^{n} p(u_i|u_{i-1})$, yielding separability, but a probabilistically equivalent model is obtained if we consider the e_i to be our random effects. Now, the u_i's are simply intermediate variables in our model, on the way to the response variables $y_i = u_i + \varepsilon_i$. Because $y_n = \varepsilon_n + \sum_{i=1}^{n} e_i$ depends directly on all of $e_1, \ldots, e_n$, the model is not separable, even though the joint density of the e's is $\prod_{i=1}^{n} p(e_i)$. The ADMB user must explicitly point out the separability structure by letting each term in (15) be a function call. Although automatic detection of separability is possible in principle, this requirement forces the user to think correctly about the structure of the model.

4. ADMB users and applications

The AUTODIF Library and ADMB were initially developed to implement highly parameterized integrated statistical fisheries assessment models that were essentially impossible to estimate in the software available at the time [16,17,18,19,28]. For this reason, the majority of early users were fisheries stock assessment modellers [43,56]. Fisheries models are still the majority of ADMB applications, but the user community is becoming more diverse through the direct promotion of ADMB by the ADMB Foundation and ADMB Project, and through courses taught by several groups, including modules within university graduate courses.

ADMB-based computer models are used globally to monitor populations of many endangered and commercially valuable species. Population models written in ADMB are used to assess more than 150 commercial fish stocks around the world, worth billions of dollars, and every U.S. NOAA Fisheries Science Center uses ADMB in their research. There are more than 150 peer-reviewed publications and numerous reports based on analyses using ADMB. Users range from universities and research institutions to government departments and private companies.

4.1 *Stock assessment models using ADMB*

Several general fisheries stock assessment models have been developed with ADMB or the AUTODIF Library alone. These models integrate a variety of data types to estimate parameters representing the fish population dynamics and how the population responds to fishing. For example, MULTIFAN-CL (http://www.multifan-cl.org/), which is based on AUTODIF, was developed to analyse tuna populations [18] and extended to include spatial structure in the population dynamics and fitting to tagging data [25].

Coleraine [27] was the first general fisheries stock assessment model developed with ADMB, and as a consequence, provided the first general Bayesian stock assessment model, which included the use of priors and modelling future projections for management strategy evaluation and decision analysis. Coleraine has been applied to several stocks globally, been used as a teaching tool in university courses and was included in the UN Food and Agriculture Organization's Bayesian stock assessment user's manual [52].

One of the largest and most used ADMB applications is Stock Synthesis [45], a very general fisheries stock assessment program. Stock Synthesis was initially used to assess U.S. Pacific groundfish stocks, but is now used globally.

4.2 *Other models using ADMB*

The efficiency and stability of parameter estimation in ADMB makes it ideal for nonlinear models that are highly parameterized [18,25,41], simpler nonlinear models with hundreds of thousands of data points [39,42], or complex nonlinear models that are parameter-rich and include large data sets [58]. It is also ideal for smaller models that need to be repeated a large number of times when performing analyses such as bootstraps, cross-validation, profile likelihoods, simulation testing [37,54] or management strategy evaluation [4,50,53]. The random effects module is ideal for implementing nonlinear random effects models [44,65] or modelling process variability in state–space models [48]. The Bayesian MCMC module can also be used to implement these models [44]. There have been several ADMB applications outside fisheries. Many of these are fisheries-related such as modelling the dynamics of marine mammals [8,9,30,63,64] and seabirds. ADMB is starting to make its way into the wildlife [10,14,15,44], genetic [12,13,32,33], botany

[31], agriculture [49], oceanography [29] and paleontology [34] literature. ADMB has also been used in economic [46,51] and medical research [35].

5. The ADMB project

The ADMB Project began in 2007 with the primary goals of sustaining ADMB into the future and expanding its user base. The first steps in the project were to make ADMB freely available and to open the source code[1]. The ADMB Foundation administers the project, the University of California holds the copyright, and ADMB is released as a free software under the BSD license.[2] ADMB is supported on all common operating systems (Windows, Linux, Mac OS, OpenSolaris), for all common C++ compilers (GCC, Visual Studio, Borland), and for both 32 and 64 bit architectures.

The software is written in C++ and GNU Flex, currently at 172,000 and 9000 lines of code, respectively. Currently, a Subversion[3] repository is maintained at the University of Hawaii, and formal releases are distributed on Google Code. The source code is tested continuously on a Buildbot[4] system, and Doxygen[5] comments are used to prepare documentation of the application programming interface (API). Extensive manuals are available for ADMB [1], the random effects module [3] and the internal AUTODIF library.

6. Discussion

AD Model Builder is a flexible and powerful tool for development of complex nonlinear statistical estimation problems. Its speed and unique capabilities, e.g. Laplace approximation, make it the tool of choice for many applications.

Although the origin of ADMB is in resource management, its use is gradually spreading to other disciplines in ecology and the social and medical sciences. Transformation of ADMB into a free open-source software vastly increases the number of potential users and should enable wider adoption of ADMB.

Opening the software to a wider community of developers offers the possibility that ADMB users might contribute specialized 'packages', bundles of ADMB functions that other users can apply in their models. The infrastructure to solicit, test and distribute such contributed packages is one of the priorities for future development. The ADMB Project welcomes scientists and programmers with the ability and enthusiasm to contribute to ADMB.

Many ADMB users are also R users, and several approaches to creating interfaces between ADMB and R have been used, for example, the glmmADMB, R2admb, and PBSadmb packages for R and the ADMB2R C++ interface. Similarly, ADMB users have expressed interest in developing translators from BUGS (Bayesian inference Using Gibbs Sampling).

There have been relatively few comparisons of performance of ADMB to that of other statistical software packages. The work comparing ADMB and other software [57] is somewhat out of date, very limited in scope and needs to be updated, but did show that ADMB performed much more efficiently than other modelling tools [57] in situations without random effects. More recently, Skaug and Fournier [61] compared ADMB to a number of statistical packages in a mixed model setting. It was concluded that ADMB performs favourably compared to the SAS procedure NLMIXED, which offers the same level of flexibility as ADMB (within a limited class of random effects models), and hence constitutes the most direct comparison.

The speed at which ADMB reaches solutions to estimation problems is one of the features that attracts users. A top priority for future development is parallel processing, which will

further improve performance. Many users are also attracted to ADMB for its MCMC capability. Improvement of the MCMC algorithms to increase both speed and accuracy have also been flagged as a development priority.

In summary, the advantages of ADMB over other statistical packages gained by the use of AD include:

(1) *Flexibility*. The user is free to define any desired model rather than being limited to a set of predefined models. Indeed, any C++ program can be written within ADMB.
(2) *Speed*. AD, the optimizer and the specialized adjoint code for key functions can make the difference between waiting hours versus seconds for model convergence.
(3) *Precision*. AD calculates the derivatives as accurately as analytical derivatives (to machine precision).
(4) *Quantification of uncertainties*. With almost no extra effort by the user AD Model builder produces several different estimates of the uncertainties of model parameters and selected derived quantities.

In addition, the generic implementation of optimization, Laplace approximation, importance sampling, MCMC, profile likelihood and estimation of the variance–covariance matrix for estimated parameters and derived quantities allows these to be applied to the desired model rather than limiting the analysis to specific models or forcing the analyst to make simplifying assumptions.

Acknowledgements

We thank the Gordon and Betty Moore Foundation for the financial support to begin the ADMB open source project and the National Oceanic and Atmospheric Administration for the ongoing financial support. We also thank Kevin Weng of the University of Hawaii Pelagics Fisheries Research Program and Mark Schildhauer of the University of California at Santa Barbara National Center for Ecological Analysis and Synthesis for logistical support. This paper was funded in part by Cooperative Agreement NA17RJ1230 between the Joint Institute for Marine and Atmospheric Research and the National Oceanic and Atmospheric Administration. Views expressed in this paper do not necessarily represent the views of these agencies, their subagencies, or their member countries. Finally, we thank the following persons for giving useful comments on the manuscript: James Bence, Cleridy Lennert-Cody, Lennart Frimannslund, Allan Hicks, Tore Selland Kleppe, Kasper Kristensen and Ian Taylor.

Notes

1. For a brief history of ADMB, see http://en.wikipedia.org/wiki/ADMB
2. http://www.opensource.org/licenses/bsd-license.php
3. http://subversion.apache.org/
4. http://buildbot.net
5. http://doxygen.org

References

[1] ADMB Project, *An introduction to AD Model Builder for use in nonlinear modeling and statistics*, Version 9.0.0, ADMB Foundation, Honolulu, HI, 2008.
[2] ADMB Project, *AUTODIF: A* C + + *array language extension with automatic differentiation for use in nonlinear modeling and statistics*, ADMB Foundation, Honolulu, HI, 2008.
[3] ADMB Project, *Random effects in AD Model Builder: ADMB-RE user guide*, ADMB Foundation, Honolulu, HI, 2009.
[4] Z.T. A'mar, A.E. Punt, and M.W. Dorn, *The impact of regime shifts on the performance of management strategies for the Gulf of Alaska walleye pollock (Theragra chalcogramma) fishery*, Can. J. Fish. Aquat. Sci. 66 (2009), pp. 2222–2242.
[5] B. Bell, *Approximating the marginal likelihood estimate for models with random parameters*, Appl. Math. Comput. 119 (2001), pp. 57–75.
[6] C. Bischof, A. Carle, G. Corliss, A. Griewank, and P. Hovland, *ADIFOR—generic derivative codes from Fortran programs*, Sci. Program. 1 (1992), pp. 1–29.

[7] B. Bolker, *Using AD Model Builder and R together: Getting started with the* R2admb *package*; software manual available at http://r-forge.r-project.org/projects/r2admb/.

[8] T. A. Branch, K. Matsuoka, and T. Miyashita, *Evidence for increases in Antarctic blue whales based on Bayesian modelling*, Mar. Mam. Sci. 20 (2004), pp. 726–754.

[9] P.A. Breen, R. Hilborn, M.N. Maunder, and S.W. Kim, *Effects of alternative control rules on the conflict between a fishery and a threatened sea lion (Phocarctos hookeri)*, Can. J. Fish. Aquat. Sci. 60 (2003), pp. 527–541.

[10] K. Broms, J.R. Skalski, J.J. Millspaugh, C.A. Hagen, and J.H. Schulz, *Using statistical population reconstruction to estimate demographic trends in small game populations*, J. Wildl. Manage. 74 (2010), pp. 310–317.

[11] B.P. Carlin and T.A. Louis, *Bayes and Empirical Bayes Methods for Data Analysis*, Chapman & Hall, London, 1996.

[12] G. Cui, A.E. Punt, L.A. Pastene, and M. Goto, *Bayes and Empirical Bayes approaches to addressing stock structure questions using mtDNA data, with an illustrative application to the North Pacific minke whales*, J. Cetacean Res. Manage. 4 (2002), pp. 123–134.

[13] E. Edeline, A. Le Rouzic, I.J. Winfield, J.M. Fletcher, J.B. James, N.C. Stenseth, and L.A. Vøllestad, *Harvest-induced disruptive selection increases variance in fitness-related traits*, Proc. R. Soc. B 276 (2009), pp. 4163–4171.

[14] E.P. Fenichel and R.D. Horan, *Gender-based harvesting in wildlife disease management*, Am. J. Agr. Econ. 89 (2007), pp. 904–920.

[15] J.R. Fieberg, K.W. Shertzer, P.B. Conn, K.V. Noyce, and D.L. Garshelis, *Integrated population modeling of black bears in Minnesota: Implications for monitoring and management*, PLoS ONE 5 (2010), e12114.

[16] D. Fournier and I. Doonan, *A length-based stock assessment method utilizing a generalized delay-difference model*, Can. J. Fish. Aquat. Sci. 44 (1987), pp. 422–437.

[17] D. Fournier and A. Warburton, *Evaluating fisheries management models by simulated adaptive control-introducing the composite model*, Can. J. Fish. Aquat. Sci. 46 (1989), pp. 1002–1012.

[18] D.A. Fournier, J. Hampton, and J.R. Sibert, *MULTIFAN-CL: A length-based, age-structured model for fisheries stock assessment, with application to South Pacific albacore, Thunnus alalunga*, Can. J. Fish. Aquat. Sci. 55 (1998), pp. 2105–2116.

[19] D. Fournier, J. Sibert, J. Majkowski, and J. Hampton, *MULTIFAN: A likelihood-based method for estimating growth parameters and age composition from multiple length frequency data sets illustrated using data for southern bluefin tuna (Thunnus maccoyii)*, Can. J. Fish. Aquat. Sci. 47 (1990), pp. 301–317.

[20] D.M. Gay, *More AD of nonlinear AMPL models: Computing Hessian information and exploiting partial separability*, in *Computational Differentiation: Techniques, Applications, and Tools*, M. Berz, C. Bischof, G. Corliss, and A. Griewank, eds., SIAM, Philadelphia, PA, 1996, pp. 173–184.

[21] A. Gelman, J.B. Carlin, H.S. Stern, and D.B. Rubin, *Bayesian Data Analysis*, Chapman & Hall, London, 1995.

[22] A. Griewank, *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, SIAM, Philadelphia, PA, 2000.

[23] A. Griewank and G.F. Corliss (eds.), *Automatic Differentiation of Algorithms: Theory, Implementation, and Application*, SIAM, Philadelphia, PA, 1992.

[24] A. Griewank, D. Juedes, and J. Utke, *ADOL-C: A package for the automatic differentiation of algorithms written in C/C++*, ACM Trans. Math. Softw. 22 (1996), pp. 131–167.

[25] J. Hampton and D.A. Fournier, *A spatially disaggregated, length-based, age-structured population model of yellowfin tuna (Thunnus albacares) in the western and central Pacific Ocean*, Mar. Freshw. Res. 52 (2001), pp. 937–963.

[26] L. Hascoët and V. Pascual, *TAPENADE 2.1 user's guide*, Tech. Rep. 300, INRIA, Sophia Antipolis, France, 2004.

[27] R. Hilborn, M. Maunder, A. Parma, B. Ernst, J. Payne, and P. Starr, *Coleraine: A generalized age-structured stock assessment model*, Version 2.0, Rep. SAFS-UW-0116, University of Washington, Seattle, 2003.

[28] K.N. Holland, R. Brill, R. Chang, J. Sibert, and D. Fournier, *Physiological and behavioural thermogregulation in bigeye tuna (Thunnus obesus)*, Nature 358 (1992), pp. 410–412.

[29] G.W. Holtgrieve, D.E. Schindler, T.A. Branch, and Z.T. A'mar, *Simultaneous quantification of aquatic ecosystem metabolism and reaeration using a Bayesian statistical model of oxygen dynamics*, Limnol. Oceanogr. 55 (2010), pp. 1047–1063.

[30] S.D. Hoyle and M.N. Maunder, *A Bayesian integrated population dynamics model to analyze data for protected species*, Anim. Biodiv. Cons. 27 (2004), pp. 247–266.

[31] S.W. Kim, I.L. Hudson, and M.R. Keatley, *Modelling the flowering of four eucalypts species via MTDg with interactions*, in *18th World IMACS/MODSIM Congress*, Cairns, Australia, 13–17 July 2009, Modelling and Simulation Society of Australia and New Zealand, Canberra, 2009, pp. 2625–2631.

[32] T. Kitakado, S. Kitada, H. Kishino, and H.J. Skaug, *An integrated-likelihood method for estimating genetic differentiation between populations*, Genetics 173 (2006), pp. 2073–2082.

[33] A. Le Rouzic, H.J. Skaug, and T.F. Hansen, *Estimating genetic architectures from artificial-selection responses: A random-effect framework*, Theoret. Popul. Biol. 77 (2010), pp. 119–130.

[34] L.H. Liow, H.J. Skaug, T. Ergon, and T. Schweder, *Global occurrence trajectories of microfossils: Environmental volatility and the rise and fall of individual species*, Paleobiology 36 (2010), pp. 224–252.

[35] A. Lunde, K.K. Melve, H.K. Gjessing, R. Skjærven, and L.M. Irgens, *Genetic and environmental influences on birth weight, birth length, head circumference, and gestational age by use of population-based parent-offspring data*, Amer. J. Epidemiol. 165 (2007), pp. 734–741.

[36] A. Magnusson, *ADMB-IDE: Easy and efficient user interface*, ADMB Foundation Newsl. 1(3) (2009), pp. 1–2.

[37] A. Magnusson and R. Hilborn, *What makes fisheries data informative?* Fish and Fish. 8 (2007), pp. 337–358.

[38] J.L. Martin, M.H. Prager, and A. Stephens, *User's guide to ADMB2R: A set of AD Model Builder output routines compatible with the R statistics language*, Tech. Memo. NMFS-SEFSC-546, NOAA, Miami, FL, 2006.

[39] M.N. Maunder, *A general framework for integrating the standardization of catch per unit of effort into stock assessment models*, Can. J. Fish. Aquat. Sci. 58 (2001), pp. 795–803.
[40] M.N. Maunder, *Converting WinBUGS into ADMB*, ADMB Foundation Newsl. 2(1) (2010), pp. 3–6.
[41] M.N. Maunder and G.M. Watters, *A-SCALA: An age-structured statistical catch-at-length analysis for assessing tuna stocks in the eastern Pacific Ocean*, Inter-Am. Trop. Tuna Comm. Bull. 22 (2003), pp. 433–582.
[42] M.N. Maunder, M.G. Hinton, K.A. Bigelow, and A.D. Langley, *Developing indices of abundance using habitat data in a statistical framework*, Bull. Mar. Sci. 79 (2006), pp. 545–559.
[43] M.N. Maunder, J. Schnute, and J. Ianelli, *Computers in fisheries population dynamics*, in *Computers in Fisheries Research*, B.A. Megrey and E. Moksness, eds., Springer, New York, 2009, pp. 337–372.
[44] M.N. Maunder, H.J. Skaug, D.A. Fournier, and S.D. Hoyle, *Comparison of estimators for mark-recapture models: Random effects, hierarchical Bayes, and AD Model Builder*, in *Modeling Demographic Processes in Marked Populations*, D.L. Thomson, E.G. Cooch, and M.J. Conroy, eds., Springer, New York, 2009, pp. 917–948.
[45] R.D. Methot Jr., *Stock assessment: Operational models in support of fisheries management*, in *The Future of Fisheries Science in North America*, R.J. Beamish and B.J. Rothschild, eds., Springer, New York, 2009, pp. 137–165.
[46] R. Meyer, D. Fournier, and A. Berg, *Stochastic volatility: Bayesian computation using automatic differentiation and the extended Kalman Filter*, Econom. J. 6 (2003), pp. 408–420.
[47] R.M. Neal, *An improved acceptance procedure for the hybrid Monte Carlo algorithm*, J. Comput. Phys. 111 (1994), pp. 194–203.
[48] A. Nielsen and J. Sibert, *State space model for light based tracking of marine animals*, Can. J. Fish. Aquat. Sci. 64 (2007), pp. 1055–1068.
[49] P.P. Olea and P. Mateo-Tomás, *The role of traditional farming practices in ecosystem conservation: The case of transhumance and vultures*, Biol. Conserv. 142 (2009), pp. 1844–1853.
[50] A.M. Parma, *Bayesian approaches to the analysis of uncertainty in the stock assessment of Pacific halibut*, in *Incorporating Uncertainty into Fishery Models*, J.M. Berkson, L.L. Kline, and D.J. Orth, eds., American Fisheries Society, Bethesda, MD, 2002, pp. 113–136.
[51] J. Paulsen, A. Lunde, and H.J. Skaug, *Fitting mixed-effects models when data are left truncated*, Insurance: Math. Econ. 43 (2008), pp 121–133.
[52] A.E. Punt and R. Hilborn, *Bayesian Stock Assessment Methods in Fisheries: User's Manual*, Comput. Inf. Ser. Fish. 12, FAO, Rome, 2001.
[53] A.E. Punt and A.D.M. Smith, *Harvest strategy evaluation for the eastern stock of gemfish (Rexea solandri)*, ICES J. Mar. Sci. 56 (1999), pp. 860–875.
[54] A.E. Punt, A.D.M. Smith, and G. Cui, *Evaluation of management tools for Australia's South East Fishery* 2: *How well can management quantities be estimated?*, Mar. Freshw. Res. 53 (2002), pp. 631–644.
[55] J. Schnute and R. Haigh, *User's guide to the R package PBSadmb*; software manual available at http://pbs-admb.googlecode.com/files/PBSadmb-UG.pdf.
[56] J. Schnute, M.N. Maunder, and J. Ianelli, *Designing tools to evaluate fishery management strategies: Can the scientific community deliver?*, ICES J. Mar. Sci. 64 (2007), pp. 1077–1084.
[57] J.T. Schnute, L.J. Richards, and N. Olsen, *Statistics, software, and fish stock assessment*, in *Fishery Stock Assessment Models*, F. Funk, T.J. Quinn II, J. Heifetz, J.N. Ianelli, J.E. Powers, J.F. Schweigert, P.J. Sullivan, and C.I. Zhang, eds., Sea Grant Program, Fairbanks, 1998, pp. 171–184.
[58] I. Senina, J. Sibert, and P. Lehodey, *Parameter estimation for basin-scale ecosystem-linked population models of large pelagic predators: Application to skipjack tuna*, Progr. Oceanogr. 78 (2008) 319–335.
[59] J. Sibert, J. Hampton, D. Fournier, and P. Bills, *An advection-diffusion-reaction model for the estimation of fish movement parameters from tagging data, with application to skipjack tuna (Katsuwonus pelamis)*, Can. J. Fish. Aquat. Sci. 56 (1999), pp. 925–938.
[60] S.P. Smith, *Differentiation of the Cholesky algorithm*, *J. Comput. Graph. Stat.* 4 (1995), pp. 134–147.
[61] H. Skaug and D. Fournier, *Automatic approximation of the marginal likelihood in non-Gaussian hierarchical models*, Comput. Stat. Data Anal. 56 (2006), pp. 699–709.
[62] H. Skaug, D. Fournier, A. Nielsen, and A. Magnusson, *Package glmmADMB: Generalized linear mixed models using AD Model Builder*; software available at vailable at http://r-forge.r-project.org/projects/glmmadmb/.
[63] H.J. Skaug, L. Frimannslund, and N. Øien, *Historical population assessment of Barents Sea harp seals* (Pagophilus groenlandicus), ICES J. Mar. Sci. 64 (2007), pp. 1356–1365.
[64] H.J. Skaug, N. Øien, T. Schweder, and G. Bøthun, *Abundance of minke whales (Balaenoptera acutorostrata) in the Northeastern Atlantic*, Can. J. Fish. Aquat. Sci. 61 (2004), pp. 870–886.
[65] V. Trenkel and H.J. Skaug, *Disentangling the effects of trawl efficiency and population abundance on catch data using random effects models*, ICES J. Mar. Sci. 62 (2005), pp. 1543–1555.

Appendix: A selection of major applications of the AUTODIF Library and AD Model Builder

Model	Model class	Reference
Coleraine	Stock assessment	Hilborn *et al.* [27]
Stock Synthesis	Stock assessment	Methot [45]
MULTIFAN-CL	Stock assessment	Fournier *et al.* [18]
stockassessment.org	Online stock assessment tool	
SEAPODYM	Ecosystem and tuna population dynamics	Senina *et al.* [58]
TAGEST	Large-scale tuna diffusion and mortality	Sibert *et al.* [59]
TRACKIT	Electronic tag track reconstruction	Nielsen and Sibert [48]