Technometrics, Volume 47, number 3, August 2005, pp 249-259

Simplifying Multivariate Second Order Response Surfaces by Fitting Constrained Models Using Automatic Differentiation

Trevor J. Ringrose

Shaun A. Forth

Applied Mathematics and Operational Research Group

Cranfield University, RMCS Shrivenham

Swindon SN6 8LA. U.K.

T.J.Ringrose@cranfield.ac.uk

Multivariate regression models for second-order polynomial response surfaces are proposed. The fitted surfaces for each response variable are constrained so that, when expressed in their canonical forms, they have features in common, such as common stationary points or common sets of eigenvectors. This can greatly reduce the number of parameters required and make the set of surfaces easier to interpret together, at the cost of a greater computational burden. However, the use of automatic differentiation within the package Matlab is shown to be easy and reduces this burden considerably. We describe the models, how to fit them and derive standard errors, report a small simulation study and the application to a data set.

KEY WORDS: Augmented information matrix; Canonical analysis; Multivariate regression; Response surface methodology

1 Introduction

Although response surface methodology (RSM) is widely used, analysis methods are usually univariate, even when the data are multivariate. With $p > 1$ response variables the usual approach is to fit the p response surfaces separately or to fit a single model to a combined response measure (Khuri and Cornell 1996, chap. 7). This paper proposes a multivariate approach to fitting second-order polynomial response surfaces, where the surfaces for the different response variables can be constrained to have certain features in common. Since it is often the canonical form which is interpreted and used we consider constraining the multivariate regression model in order to make

these canonical forms as similar as possible.

If p is large the p canonical forms are difficult to interpret as a set, but sometimes the eigenvectors of each canonical form are similar, although the eigenvalues are different. Flury (1988) noticed this feature in the principal components of grouped multivariate data, where the eigenvectors for different groups were often similar even when the eigenvalues were not. He therefore proposed the Common Principal Component (CPC) model, and here we propose an analogous model, where in this case the 'groups' are the different response variables. In addition, we propose models where the responses have a common stationary point, similarly to the estimator and test for equality proposed by Bockenholt (1989).

The justification for these models is the usual argument of parsimony, where a model with fewer parameters and similar quality of fit is seen as preferable. Specifically, it is hoped that the p response surfaces will be easier to interpret and understand, while predictions from them will not be 'significantly' worse than those from the usual model. Indeed, variance should decrease, even if bias increases, so that mean square error may decrease. Hence the objective is that when a constrained model is valid, using some model selection criterion, it should be easier to understand and have fitted values close to those for the unconstrained model throughout the region of experimentation, while having lower parameter variances and possibly also lower predictive mean square error for new observations.

Fitting such models by maximum likelihood requires a function minimisation routine, and here we use Automatic Differentiation (AD) to supply derivatives to the routine, thus substantially aiding the speed and accuracy of the results.

The following introduces standard terminology and notation while section (2) describes the proposed models. Section (3) describes parameter estimation using maximum likelihood and automatic differentation, and section (4) the derivation of standard errors. Section (5) shows the application of these models to a data set and section (6) reports a small simulation study assessing their performance.

Consider n observations on a single response variable Y with q explanatory variables, typically derived from a central composite design. If $\boldsymbol{x}_i$ is the q-vector containing the values of the explanatory variables for the i-th observation the second order polynomial regression model can be written as

(Myers and Montgomery 1995, chap. 6)

$$E(Y_i) \;=\; \beta_0 + \boldsymbol{x}_i^T \boldsymbol{\beta}_1 + \boldsymbol{x}_i^T B \boldsymbol{x}_i \qquad i = 1 \ldots n \tag{1}$$

The parameter vector $\boldsymbol{\beta}$ is partitioned so that β_0 contains the constant (or block) term, $\boldsymbol{\beta}_1$ the linear terms and $\boldsymbol{\beta}_2$ the square and cross-product terms, with B a symmetric matrix containing the elements of $\boldsymbol{\beta}_2$ written in the form

$$B \;=\; \begin{bmatrix} \beta_{11} & \frac{1}{2}\beta_{12} & \cdots & \frac{1}{2}\beta_{1q} \\ \frac{1}{2}\beta_{21} & \beta_{22} & \cdots & \frac{1}{2}\beta_{2q} \\ \vdots & \vdots & \vdots & \vdots \\ \frac{1}{2}\beta_{q1} & \frac{1}{2}\beta_{q2} & \cdots & \beta_{qq} \end{bmatrix}$$

where $\beta_{uv} = \beta_{vu}$ is the coefficient of $x_u x_v$. The fitted model can be rewritten into its canonical form as

$$\widehat{Y}(\boldsymbol{x}) \;=\; \hat{\beta}_0 + \boldsymbol{x}^T \hat{\boldsymbol{\beta}}_1 + \boldsymbol{x}^T \hat{B} \boldsymbol{x} \;=\; \widehat{Y}(\hat{\boldsymbol{x}}_0) + \sum_{k=1}^{q} \hat{\lambda}_k w_k^2 \tag{2}$$

where the stationary point on the surface is $\hat{\boldsymbol{x}}_0 = -(1/2)\hat{B}^{-1}\hat{\boldsymbol{\beta}}_1$ and the fitted value at the stationary point is $\widehat{Y}(\hat{\boldsymbol{x}}_0) = \hat{\beta}_0 + (1/2)\hat{\boldsymbol{x}}_0^T \hat{\boldsymbol{\beta}}_1$, while $\hat{\lambda}_k$ and $\hat{\boldsymbol{m}}_k$ are the eigenvalues and eigenvectors of $\widehat{B}$ with $w_k = \hat{\boldsymbol{m}}_k^T(\boldsymbol{x} - \hat{\boldsymbol{x}}_0)$ being the coordinate of $\boldsymbol{x}$ with respect to the axis defined by $\hat{\boldsymbol{m}}_k$. Hence (2) centres the fitted response surface at the stationary point $\hat{\boldsymbol{x}}_0$ and resolves it into q orthogonal directions, defined by the eigenvectors $\hat{\boldsymbol{m}}_k$, where the corresponding eigenvalue $\hat{\lambda}_k$ gives the curvature in that direction. This is the 'B canonical form' of the fitted response surface. The explanatory variables must be coded, typically so that in a CCD the central design point is at zero and the corner points are ± 1. This canonical form allows easier interpretation of the response surface, and is often used to look for 'ridges' where the eigenvalue is close to zero.

The similarity to Principal Component Analysis (PCA) is clear, with $\hat{B}$ playing the role of the covariance matrix, although note that it need not be positive definite.

2 Constrained multivariate response surfaces

Now consider p response variables, using the notation of section 1, except that the superscript $^{(j)}$ denotes quantities relating to the j-th response variable. The second-order multivariate linear

regression model with p responses, b blocks and q explanatory variables can be partitioned as

$$Y = X\beta + \Upsilon = X_0\beta_0 + X_1\beta_1 + X_2\beta_2 + \Upsilon$$

where $\beta_0(b \times p)$ contains the block/constant terms, $\beta_1(q \times p)$ the linear terms, $\beta_2(q^\star \times p)$ the second-order terms and the design matrix X is partitioned conformably, where $q^\star = q + q(q-1)/2$. Hence $\boldsymbol{\beta}_u^{(j)}$ is the j-th column of β_u $(u = 0, 1, 2)$, $\boldsymbol{x}_i$ is the i-th row of X_1, and let $\boldsymbol{x}_i^\star$ be the i-th row of X_2. We make the usual assumption of multivariate normality, but no assumptions about the correlation structure, so the i-th row of Υ is

$$\boldsymbol{v}_i \sim N_p(\mathbf{0},\ \Sigma) \qquad i = 1 \ldots n\ , \quad \text{independent and identically distributed}$$

All models are now stated in terms of $E(Y_{ij})$, the i-th observation on the the j-th response variable. These will be written to emphasize the structure, rather than in the shortest form. The unconstrained multivariate regression model (model U) can be written as

$$\begin{aligned} E(Y_{ij}) &= \beta_0^{(j)} + \boldsymbol{x}_i^T\boldsymbol{\beta}_1^{(j)} + \boldsymbol{x}_i^{\star T}\boldsymbol{\beta}_2^{(j)} \qquad i = 1 \ldots n\ , \quad j = 1 \ldots p \qquad (3) \\ &= \beta_0^{(j)} + \boldsymbol{x}_i^T\boldsymbol{\beta}_1^{(j)} + \boldsymbol{x}_i^T B^{(j)}\boldsymbol{x}_i \end{aligned}$$

where $B^{(j)}$ is $\boldsymbol{\beta}_2^{(j)}$ rewritten as in section 1. The eigendecomposition giving the canonical form is

$$B^{(j)} = M^{(j)}\Lambda^{(j)}M^{(j)T}$$

where the eigenvectors $\boldsymbol{m}_k^{(j)}(k = 1 \ldots q)$ of $B^{(j)}$ are the columns of the orthogonal matrix $M^{(j)}$ and $\Lambda^{(j)}$ is a diagonal matrix containing the eigenvalues $\lambda_k^{(j)}(k = 1 \ldots q)$. Since $\boldsymbol{\beta}_1^{(j)} = -2B^{(j)}\boldsymbol{x}_0^{(j)}$, the unconstrained model can also be written as

$$\begin{aligned} E\,(Y_{ij}) &= \beta_0^{(j)} + \boldsymbol{x}_i^T\boldsymbol{\beta}_1^{(j)} + \boldsymbol{x}_i^T M^{(j)}\Lambda^{(j)}M^{(j)T}\boldsymbol{x}_i \qquad i = 1 \ldots n\ , \quad j = 1 \ldots p \qquad (4) \\ &= \beta_0^{(j)} + \boldsymbol{x}_i^T M^{(j)}\Lambda^{(j)}M^{(j)T}(\boldsymbol{x}_i - 2\boldsymbol{x}_0^{(j)}) \qquad (5) \end{aligned}$$

In the following we will use whichever of (3), (4) or (5) is the more convenient.

Three possible constraints on the parameter matrix β are considered here.

(a) Stationary points constrained to equality so that, for a common stationary point $\boldsymbol{x}_0$,

$$\boldsymbol{x}_0^{(j)} = -\frac{1}{2}(B^{(j)})^{-1}\boldsymbol{\beta}_1^{(j)} = \boldsymbol{x}_0 \quad \Longrightarrow \quad \boldsymbol{\beta}_1^{(j)} = -2B^{(j)}\boldsymbol{x}_0 \qquad \forall j = 1 \ldots p$$

(b) Eigenvectors of $B^{(j)}$ constrained to equality so $M^{(j)} = M$ for all j and

$$B^{(j)} = M\Lambda^{(j)}M^T \qquad j = 1, \ldots, p$$

This is directly analogous to CPC. The principal directions of variation are the same for each response, but the curvatures are unconstrained and hence so are the orderings of the directions.

(c) If the eigenvectors of $B^{(j)}$ are constrained to equality, then the sets of eigenvalues may also be constrained to be scalar multiples of each other

$$\Lambda^{(j)} = c_j \Lambda \quad \Longrightarrow \quad B^{(j)} = c_j M \Lambda M^T \qquad j = 1, \ldots, p$$

Since $B^{(j)}$ is just a recoding of $\boldsymbol{\beta}_2^{(j)}$, this implies that the columns of β_2, the matrix of second order terms, are scalar multiples of each other and hence, for vector $\boldsymbol{c} = (c_1, \ldots, c_p)^T$,

$$\boldsymbol{\beta}_2^{(j)} = c_j \boldsymbol{\beta}_2 \implies \beta_2 = \boldsymbol{\beta}_2 \boldsymbol{c}^T \qquad j = 1, \ldots, p$$

so that β_2 is rank 1. Hence both the directions and the relative sizes of the curvatures are the same for all responses, only the scaling of the curvatures varies. This is similar to reduced rank regression (Reinsel and Velu 1998), although here all the second-order terms are expressed in reduced-rank form, rather than all or a subset of the variables.

These constraints are combined to give the 5 constrained models considered here. Various parameterisations can be used, here each model is given firstly in terms of (5) to illustrate the differences between the models and secondly, if applicable, a parameterisation which is superior from a computational point of view. In many cases heavily constrained models can be expressed so that optimisation is unconstrained and involves few parameters. In particular, optimisation can be very unreliable when an unconstrained stationary point $\boldsymbol{x}_0^{(j)}$ is included in the model, since if any eigenvalues are close to zero then this has very high variability. For maximisation purposes, a parameterisation not including $\boldsymbol{x}_0^{(j)}$ or $\boldsymbol{x}_0$ should be used whenever possible. The cases where the optimisation routine has failed to converge within a reasonable number of iterations have nearly all been for models S and SE, where $\boldsymbol{x}_0$ must be included directly.

E: Stationary points are unconstrained, while all sets of eigenvectors are identical but eigenvalues are unconstrained. Hence $M^{(j)} = M \ \forall j$ and so

$$E(Y_{ij}) = \beta_0^{(j)} + \boldsymbol{x}_i^T M \Lambda^{(j)} M^T (\boldsymbol{x}_i - 2\boldsymbol{x}_0^{(j)}) \tag{6}$$

However, a computationally superior form is

$$E(Y_{ij}) = \beta_0^{(j)} + \boldsymbol{x}_i^T \boldsymbol{\beta}_1^{(j)} + \boldsymbol{x}_i^T M \Lambda^{(j)} M^T \boldsymbol{x}_i \tag{7}$$

EE: Stationary points are unconstrained, while all sets of eigenvectors are identical and all sets of eigenvalues are scalar multiples of each other. Hence $M^{(j)} = M$ and $\Lambda^{(j)} = c_j\Lambda\ \forall j$ so

$$E\left(Y_{ij}\right) \quad = \quad \beta_0^{(j)} + \boldsymbol{x}_i^T M c_j \Lambda M^T (\boldsymbol{x}_i - 2\boldsymbol{x}_0^{(j)}) \tag{8}$$

For computational purposes, we use $\beta_2 = \boldsymbol{\beta}_2\boldsymbol{c}^T$ and so

$$E\left(Y_{ij}\right) \quad = \quad \beta_0^{(j)} + \boldsymbol{x}_i^T \boldsymbol{\beta}_1^{(j)} + \boldsymbol{x}_i^{\star T} c_j \boldsymbol{\beta}_2 \tag{9}$$

for vectors $\boldsymbol{c}, \boldsymbol{\beta}_2$. For identifiability $\boldsymbol{c}$ is constrained by either $\boldsymbol{c}^T\boldsymbol{c} = p$ or $c_p = 1$.

S: All stationary points are identical, while eigenvectors and eigenvalues are unconstrained. Hence $\boldsymbol{x}_0^{(j)} = \boldsymbol{x}_0\ \forall j$ and so

$$E\left(Y_{ij}\right) = \beta_0^{(j)} + \boldsymbol{x}_i^T M^{(j)} \Lambda^{(j)} M^{(j)T} (\boldsymbol{x}_i - 2\boldsymbol{x}_0) \tag{10}$$

However, for computational purposes it is unnecessary to fit in terms of the eigendecomposition, so instead use

$$E\left(Y_{ij}\right) = \beta_0^{(j)} + \boldsymbol{x}_i^T(-2B^{(j)}\boldsymbol{x}_0) + \boldsymbol{x}_i^T B^{(j)} \boldsymbol{x}_i \tag{11}$$

for matrix β_2 (whose j-th column is recoded to construct $B^{(j)}$).

SE: All stationary points and all sets of eigenvectors are identical, while eigenvalues are unconstrained. Hence $\boldsymbol{x}_0^{(j)} = \boldsymbol{x}_0$ and $M^{(j)} = M\ \forall j$ so that

$$E\left(Y_{ij}\right) = \beta_0^{(j)} + \boldsymbol{x}_i^T M \Lambda^{(j)} M^T (\boldsymbol{x}_i - 2\boldsymbol{x}_0) \tag{12}$$

SEE: All responses are linearly related, so that all stationary points and all sets of eigenvectors are identical, while all sets of eigenvalues are scalar multiples of the sets for the other responses. Hence $\boldsymbol{x}_0^{(j)} = \boldsymbol{x}_0$, $M^{(j)} = M$ and $\Lambda^{(j)} = c_j\Lambda\ \forall j$ so

$$E\left(Y_{ij}\right) = \beta_0^{(j)} + \boldsymbol{x}_i^T M c_j \Lambda M^T (\boldsymbol{x}_i - 2\boldsymbol{x}_0) \tag{13}$$

However, adding models S and EE together it is clear that this implies that β_1 and β_2 are both rank 1, so that for maximisation we use

$$E\left(Y_{ij}\right) = \beta_0^{(j)} + \boldsymbol{x}_i^T c_j \boldsymbol{\beta}_1 + \boldsymbol{x}_i^{\star T} c_j \boldsymbol{\beta}_2 \tag{14}$$

where again $\boldsymbol{c}$ is constrained by either $\boldsymbol{c}^T\boldsymbol{c} = p$ or $c_p = 1$.

The number of parameters and constraints in each model, for the 'natural' and the computationally optimal forms listed above, are as follows (equation number in brackets):

	natural parameterisation			computational parameterisation		
model	eqn	parameters p_m	constr. p_c	eqn	parameters p_m	constr. p_c
U	(5)	$p(b+q+q+q^2)$	$pq^\star$	(3)	$p(b+q+q^\star)$	0
E	(6)	$p(b+q+q)+q^2$	$q^\star$	(7)	$p(b+q+q)+q^2$	$q^\star$
EE	(8)	$p(b+q)+p+q+q^2$	$q^\star+1$	(9)	$p(b+q)+(p-1)+q^\star$	0
S	(10)	$p(b+q+q^2)+q$	$pq^\star$	(11)	$p(b+q^\star)+q$	0
SE	(12)	$p(b+q)+q+q^2$	$q^\star$	(12)	$p(b+q)+q+q^2$	$q^\star$
SEE	(13)	$pb+q+p+q+q^2$	$q^\star+1$	(14)	$pb+(p-1)+q+q^\star$	0

(15)

Here $q^\star$ is also the number of constraints required for a $q \times q$ matrix to be orthogonal. For each model, the degrees of freedom are $np-(p_m-p_c)$, which is the same in both parameterisations.

3 Maximum likelihood estimation using numerical optimisation and automatic differentiation

Due to the assumption of multivariate normality, these models can be fitted using maximum likelihood in the usual way (Anderson 1984, p.60). In all cases the concentrated likelihood with $\hat{\Sigma} = n^{-1}(Y - X\hat{\beta})^T(Y - X\hat{\beta})$ has been used (Seber and Wild 1989, pp.37–42, pp.581–585). For models such as (11) the constraints are implemented by parameterising the expression for $E(Y_{ij})$ so that an unconstrained optimisation routine can be applied, in this case the MATLAB routine `fminunc`. In other cases M must be orthogonal and/or $\boldsymbol{c}$ must be of fixed length, as in (8), so a constrained optimisation routine using Lagrange multipliers is used, in this case the MATLAB routine `fmincon`. Both `fminunc` and `fmincon` are general function minimisation routines, described in Coleman, Branch and Grace (1999), used to minimise minus the log-likelihood, written in terms of one of (6) to (14).

Both routines are gradient-based, in particular their default algorithms approximate function gradients with finite-differences and use them for a BFGS updating of an approximate Hessian (matrix of second derivatives). Use of Hessian information conveys a theoretical superlinear convergence to local minima onto the resulting quasi-Newton scheme. For more on numerical optimisation in statistics see Monahan (2001).

However, optimisation of these likelihood functions proved to be slow and unreliable in many cases. At this stage, we could have algebraically differentiated each of the log-likelihood functions with respect to each of the parameters and hence hand-coded MATLAB functions for the gradients of the objective functions and constraints to replace the default finite-differencing. Given the complexity of the objective functions this would have been time-consuming and error-prone. Hence Automatic Differentiation (AD) was used to provide the optimisation routines with exact derivatives, resulting in faster and more accurate results as well as more reliable convergence (Ringrose and Forth 2002). More details of timings are given in Forth and Ketzscher (2004, sec. 4).

Automatic Differentiation (AD) refers to the software tools and associated mathematical analysis required to differentiate functions defined by computer programs (Griewank 2000). There are two standard AD algorithms for calculating first derivatives, forward and reverse. In the *forward method*, systematic application of the chain rule is used to calculate a variable's derivatives simultaneously with its values, following the control-flow of the program. The *reverse* or *adjoint method* is a two stage process. First the original program, augmented with statements to store additional data, is executed in a conventional forward direction. In the second phase, the program is executed in reverse, propagating sensitivities of the program's outputs to internal variables, with the data stored in the first phase being used to ensure that variables take their correct values and that the program's control-flow is reversed. By the end of this second phase the calculated sensitivities of the output variables to the inputs yield the derivatives. Reverse mode is typically faster (Griewank 2000) when, as with maximum likelihood estimation, there are fewer outputs than inputs. However, the data storage requirements for naive use of reverse mode may be impracticably large.

Further, AD software is implemented in one of two ways, source transformation or operator overloading. In the source transformation approach compiler technology is used to read in a (usually FORTRAN or C) program, augment it with statements to calculate derivatives, then write out the desired derivative code, which can then be compiled. Examples of source-transformation tools are ADIFOR (Bischof, Carle, Khademi and Mauer 1996), TAMC (Giering and Kaminski 1998), and ADIC (Bischof, Roh and Mauer 1997). The second implementation strategy takes advantage of operator overloading, a feature of many modern object-oriented programming languages. In this approach a new class is defined to hold an object's derivative information as well as value. Arithmetic operations, such as the adding of two objects, are defined for the new class and correspond to applying the operation to the object's value and simultaneously updating the derivative

information. Examples of the operator-overloading approach are AD01 (Pryce and Reid 1998) and ADOL-C (Griewank *et al* 1999).

Tools for AD in MATLAB are not yet as mature as those for Fortran and C. In a monumental effort, Verma (1998) produced ADMAT, an operator-overloaded implementation of forward and reverse mode AD for both first and second derivatives. More recently Bischof, Bucker, Lang, Rasch and Vehreschild (2002) have implemented a source-transformation AD tool. One of us (Forth 2001) has designed and implemented an operator-overloaded AD tool for MATLAB named MAD. At present MAD only utilizes the forward mode of AD for first derivatives via the `fmad` class. The major difference from the operator-overloaded forward mode implemented in ADMAT is MAD's use of an optimised class named `derivvec` for storing and manipulating directional derivatives. This both improves maintainability and, for more than about 5 directional derivatives, yields improved performance.

In the context of AD applied to maximum likelihood estimation, Skaug (2002) states that "tuning a general purpose AD-tool, such as ADIFOR, to a particular optimization problem may require substantial knowledge about how AD works". Though we believe this statement to be true in general, we have found use of AD to be straightforward when using MAD's `fmad` class to supply derivatives to MATLAB functions for the solution of differential equations (Shampine, Ketzscher and Forth 2003) or maximum likelihood optimisation problems (Ringrose and Forth 2002).

A few lines of MATLAB code are now shown to demonstrate how little needs doing to enable the use of AD. When not using AD, the minimisation command for model E is

```
x = fmincon('Eobjfun',x, ... ,'Econ',options,X0,X1,Y)
```

where `x` is a vector containing all of the parameter estimates, `options` the tolerances used in the algorithm and `X0,X1,Y` the data, while `Eobjfun` is a MATLAB function which calculates the value of the log-likelihood and `Econ` is a MATLAB function defining the constraints. When using MAD only two changes are needed. Firstly a 'wrapper' function is written for each function in which MAD is to be used, in this case `Eobjfun` and `Econ`. These wrapper functions, `Emad` and `Emadcon` respectively, are then called instead:

```
x = fmincon('Emad',x, ... ,'Emadcon',options,X0,X1,Y)
```

The wrapper function `Emad` (`Emadcon` is almost identical) is simply

```
function [f,g] = Emad(x,X0,X1,Y);
if nargout == 1             % Only function value required
  f=Eobjfun(x,X0,X1,Y);     % calculate function value
else                        % Function value and gradient required
  x=fmad(x,eye(length(x))); % define x to be of type fmad
  f=Eobjfun(x,X0,X1,Y);     % calculate function value and derivatives
  g=getinternalderivs(f);   % extract derivatives
  f=getvalue(f);            % extract function value
end
```

If `fmincon` requires only the function value `f`, `Emad` passes it straight to `Eobjfun`. If the gradient `g` is also required then `Emad` uses the supplied vector `x` and the equivalently-sized identity matrix `eye(length(x))` to convert `x` to be a vector of `fmad` class whose value is unchanged but which now has derivatives given by the identity matrix. Within the ensuing call to `Eobjfun`, all arithmetic operations using `x` are therefore performed using `fmad` library routines which calculate both an object's value and its derivatives, and so return an `fmad` object. The object on the left-hand side of the assignment is consequently also of `fmad` class, so that anything calculated either directly or indirectly as a function of `x` will itself be of class `fmad`. The newly calculated objects' derivatives are calculated from those of `x` by the chain rule so, since `x`'s derivatives have been set as the identity matrix, all derivatives may therefore be regarded as gradients with respect to `x`. For example, in `Eobjfun` the first block of elements in `x` are unpacked into the $q01 \times p$ matrix `Beta01` using `Beta01=reshape(x(1:q01*p),[q01 p])` so that `Beta01` is of class `fmad`, and its derivatives with respect to `x` stored, and this will also be the case for any other variable calculated from `Beta01`. However, sometimes it is efficient to initialise variables to be a given size, but a command such as `D=zeros(n,p)` will mean that `D` is not of class `fmad`. Hence the second change required is that we take care to explicitly define such variables in terms of `x` or a function thereof, for example `D=zeros(n,size(Beta01,2))` instead, using the fact that `Beta01` is an `fmad` variable with p columns. Once the calculation of `Eobjfun` is complete, the `fmad` object `f` is passed back to `Emad` and its value and derivatives are extracted using the MAD `getvalue` and `getinternalderivs` functions.

4 Standard errors

When using unconstrained optimisation the asymptotic covariance matrix of the estimates can be estimated in the usual way by inverting the Hessian of the log-likelihood, evaluated at the maximum. A source-transformation AD tool can of course be applied twice to obtain exact second derivatives. In the present case, with an operator-overloading AD tool which does not currently calculate second derivatives, AD can be applied to central finite differences to obtain the approximate answer using $f''(x) \approx (2\delta)^{-1}(f'(x+\delta) - f'(x-\delta))$ for small δ.

When optimisation is constrained a result of Aitchison and Silvey (1958), also given in Silvey (1975, pp.79–81), allows this approach to be used with a small modification. A similar method is used in Reinsel and Velu (1998, sec. 3.5). The following description is rewritten slightly from Silvey (1975, p.81). In all cases everything is evaluated at the maximum $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$.

Let all p_m parameters be contained in the vector $\boldsymbol{\theta}$, with log-likelihood function $L(\boldsymbol{\theta})$, and let all p_c constraints be written in the form $c_j(\boldsymbol{\theta}) = 0$. Let H be the Hessian and C be a matrix containing the derivatives of the constraints with respect to the parameters, defined to have i, j-th elements

$$\begin{aligned} \{H\}_{ij} &= \frac{\partial^2 L(\boldsymbol{\theta})}{\partial\theta_i \partial\theta_j} \qquad i, j = 1 \ldots p_m \\ \{C\}_{ij} &= n\frac{\partial c_j(\boldsymbol{\theta})}{\partial\theta_i} \qquad i = 1 \ldots p_m \ , \ j = 1 \ldots p_c \end{aligned} \tag{16}$$

C is always very sparse with the non-zero elements easy to find algebraically (appendix A.1).

In unconstrained optimisation the asymptotic covariance matrix $\mathrm{var}(\hat{\boldsymbol{\theta}})$ is estimated by $V = -H^{-1}$. For constrained optimisation this is replaced by what is called in the Factor Analysis literature, apparently following Lawley (1975), an 'augmented information matrix', so that

$$\widehat{\mathrm{var}}(\hat{\boldsymbol{\theta}}) = V \qquad \text{where} \qquad \begin{bmatrix} -H & C \\ C^T & 0 \end{bmatrix}^{-1} = \begin{bmatrix} V & \star \\ \star & \star \end{bmatrix} \tag{17}$$

(the $\star$ values are irrelevant). This can be multiplied by $np/(np - (p_m - p_c))$ from (15) to correct for the usual downward bias in maximum likelihood variance estimates.

Confidence and prediction intervals for the fitted values can be constructed using linear approximations based on the Taylor series expansion (Seber and Wild 1989, p.193). The variance for the fitted value at any point $\boldsymbol{x}_r$ is (appendix A.2)

$$\widehat{\mathrm{var}}(\hat{Y}_r^{(j)}) = \boldsymbol{f}_r^{(j)T} V \boldsymbol{f}_r^{(j)} \qquad \text{where} \qquad \boldsymbol{f}_r^{(j)} = \frac{\partial y_{rj}}{\partial \boldsymbol{\theta}} \tag{18}$$

Similarly, the Taylor series-based methods in Seber and Wild (1989, p.24, p.532) can also be used to calculate V. For a single response, $F = \partial \boldsymbol{y}/\partial \boldsymbol{\theta}$ takes the role played by the design matrix X in univariate linear regression, so that the approximate variance matrix is $\sigma^2(F^TF)^{-1}$. In the present multivariate case put $\boldsymbol{y} = \mathrm{vec}(Y)$ and $F = \partial \boldsymbol{y}/\partial \boldsymbol{\theta}$ so that for unconstrained optimisation

$$\widehat{\mathrm{var}}(\hat{\boldsymbol{\theta}}) = V = (F^T(\Sigma^{-1} \otimes I_n)F)^{-1} \tag{19}$$

For constrained optimisation, it seems similarly reasonable to substitute $F^T(\Sigma^{-1} \otimes I_n)F$ for $-H$ in (17). It is straightforward to calculate F algebraically or by AD (appendix A.2).

The models were all fitted using the computationally-superiors forms, such as (9). However, since for any function $\widehat{g(\boldsymbol{\theta})} = g(\hat{\boldsymbol{\theta}})$ these MLEs can be reparameterised into the more 'natural' forms, such as (8), so that (17) can be used to derive estimated asymptotic standard errors for the eigenvalues, eigenvectors and stationary points directly.

Since the unconstrained model can be fitted directly in its canonical form using (4), it is instructive to compare both the parameter estimates and the standard errors from (17) to the results calculated in the usual way. Ringrose and Forth (2002) showed that parameter estimates were consistently close to those from the usual $\hat{\beta} = (X^TX)^{-1}X^TY$. Similarly, the covariance matrix for the block terms, first order terms, eigenvalues and eigenvectors can be calculated using (17), and multiplied by $np/(np - (p_m - p_c))$. In all cases investigated, the variances for the block and first orders terms agree very closely with those from $(X^TX)^{-1}$. Similarly, the eigenvalue variances agree very closely with those from the 'double linear regression' method of Bisgaard and Ankenman (1996). This is unsurprising as the method used here is similar to the equivalent (Bisgaard and Ankenman 1996, sec. 5) delta-method approach of Carter, Chinchilli and Campbell (1990). However, the approach here also gives covariances between the eigenvalues and both variances and covariances for the eigenvector coefficients. The results for confidence intervals (18) were also very close to those calculated in the usual way. In most cases the variances calculated by plugging (19) into (17) gave answers which were slightly closer to those from the standard theory than were those from approximating the Hessian by AD and finite differences. Since it is also quicker in all cases, this is the preferred method to calculate standard errors.

Since the full unconstrained model is the most difficult to fit in the canonical form, with more parameters requiring more constraints to orthogonality, these results suggest that the approach adopted here should be giving reliable parameter estimates and standard errors for the constrained models too.

Model selection can be effected using the likelihood ratio test, since each letter added to the model name denotes a nested model, allowing backward elimination from the unconstrained model. Alternatively, information criteria allow all of the models to be compared at once. Although variants have been proposed, here we use the standard Akaike Information Criterion (AIC) described in Sakamoto, Ishiguro and Kitagawa (1986), so that the 'best' model is taken to be that which minimises $-2L(\hat{\boldsymbol{\theta}}) + 2(p_m - p_c)$.

5 Example

The flour dough data in Gilmour and Ringrose (1999) have $n = 28$ observations of $p = 9$ responses describing aspects of dough quality, with $q = 3$ explanatory variables (flow rate, moisture content and screw speed). The 9 responses are in 3 sets of 3, the sets measuring size, strength and colour. The models proposed here have been fitted to all 9 variables and to the obvious subsets of 3 and 6. None of the models proposed here fit all of the sets together, but model E fits the 6 responses measuring size and strength, giving an AIC of 751.2 compared to an AIC of 752.2 for the unconstrained model (and 770.4 for SEE). Hence we consider the $p = 6$ responses measuring dough size (cross-sectional expansion, longitudinal expansion and specific volume) and dough strength (shear strength, hardness and compression curve area).

Omitting the $b = 7$ block effects, which are almost identical in all models, estimated regression coefficients for both the unconstrained second-order model (U) and model E are in table 1. Clearly the first order terms are very similar but the second-order terms are quite different, as expected. Residual plots are quite similar in the two cases, and give no reason to seriously doubt the second-order model. Model E took 78 seconds to fit with finite differences or 63 seconds with AD, although the difference can be much greater, for example model EE took 103 seconds with finite differences but 8 seconds with AD, while fitting the unconstrained model directly in the canonical form takes 300–600 seconds using finite differences and 50–100 seconds using AD, depending on the initial estimates used.

The eigenvalues and eigenvectors, and their standard errors, for the canonical forms are given in table 2 for U and table 3 for E. Careful inspection of the two models shows that they are fairly similar, even though the model has been reduced from 54 to 39 (non-block term) parameters. For example, for the 3rd response variable, under the unconstrained model the largest eigenvalue is 0.8511,

with a standard error of 0.2357, corresponding to the eigenvector $(-0.1376, -0.8794, 0.4558)^T$, while for model E it is 0.8903, with a standard error of 0.1494, corresponding to the eigenvector $(0.0970, 0.9453, -0.3115)^T$.

The standard errors are all calculated using (19) and (17), and those for the eigenvalues in U agree with those from the double linear regression method to at least 3 decimal places. In every case apart from Y_3 standard errors for the eigenvalues are smaller for E than for U. In addition, the spread of eigenvalues is smaller in E than in U, again except for Y_3. Since, in most eigenvalue problems, the spread of the sample eigenvalues tends to exceed that of the population eigenvalues (Friedman 1989, p.166; Ringrose and Benn 1997, p.1528) this is probably a good thing. As expected, the standard errors for the eigenvector coefficients are much smaller for E than for U, since these parameters have been estimated using the entire data set rather than just a sixth of it. The eigenvector coefficients are of course highly correlated, and given their constraints it would not make sense to construct normal-based confidence intervals. Moderately-large coefficients tend to have the largest standard errors, as might be expected.

The common set of eigenvectors in the more parsimonious model makes the canonical forms much easier to interpret as a whole, in particular the similarities and differences between the responses. In this case in model E the ordering of the eigenvectors is the same for each of Y_1 to Y_3, agreeing very well with the unconstrained first eigenvectors. Although constrained and unconstrained 2nd and 3rd eigenvectors differ by more, the size of the standard errors of the eigenvalues shows that confidence intervals for the 3rd eigenvalues will include zero, while the 2nd and 3rd eigenvectors could easily 'swap over' between population and sample. Hence the 2nd and 3rd eigenvectors may only define a plane rather than distinct directions, and that defined by the constrained eigenvectors is as good as any. Indeed these results illustrate just how variable the estimated features of canonical forms can be.

These data were collected to help understand the relationship between input and output variables in general, although with particular interest in process control. Gilmour and Ringrose (1999) discussed the use of simplified eigenvectors (small elements set to zero) for this. This paper addresses simplification of a different kind, and process control could be aided particularly if model SE or SEE fits, since the effect of moving along any eigenvector on each response can be seen immediately from the different eigenvalues. In this case only model E fits, so that the different stationary points (not shown here) must be allowed for, and hence the main benefit is to interpretability rather than

to process control directly.

6 Simulations

In order to use these models in practice we require that, when the constrained model seems to give an adequate fit, its fitted values within the region of experimentation do not deviate from the true response values by a substantially greater margin than do those for the unconstrained model, while confidence intervals for these fitted means have inclusion rates close to the nominal value. Extensive simulations have been performed on these proposed models, here only a subset is reported to address the above questions, while for brevity's sake only a brief graphical summary of the results is given. The performance of the parameter estimates and their standard errors are not reported, but the results in section 5 are fairly typical.

For given p and q, X is created as a CCD with axial points on the unit hypersphere, q centre points and no blocking. Only fairly small p, q are used as these are where the models are most likely to be used. Here $\Sigma = I_p$ is taken as the standard, with alternatives of random, unequal variances and random, unequal correlations, giving four different types of Σ.

For each combination of p, q, type of Σ and underlying model (S, SE, SEE, E, EE, U), 250 underlying true response surfaces (β) and covariance matrices (Σ) are randomly generated, with the true stationary point within, or very close to, the region of experimentation, and so that a second-order fit will almost certainly be appropriate. A simulated data set (Y) is generated from each of these and each of the 6 models fitted. For each of the constrained models in each of the simulated data sets, if the AIC is lower than that for the unconstrained model, the fitted values and variances at each of 1000 test cases are calculated and compared to those for the unconstrained model. These 1000 test cases are generated randomly from within the hypersphere defining the region of experimentation.

The accuracy of the fitted values and confidence intervals are then summarised over the 1000 test cases for the up to 250 simulated data sets where each constrained model fit the data adequately. Note that this involves averaging over different β and Σ, albeit ones of similar type, but this seems preferable to considering only a single β and Σ, or small number thereof, for each combination of p, q and underlying model. The study was run as a full factorial design with factors $p = \{2, 4\}$, $q = \{3, 6\}$, true model = {S, SE, SEE, E, EE, U}, variances = {unity, random unequal (average

roughly unity)} and correlations = {zero, random}.

Many summary measures of performance were calculated, but for simplicity only two 'headline' figures are reported here. In both cases, the estimates are based only on those simulated data sets where, according to the AIC, the particular constrained model fitted adequately. The value of $(\hat{y} - E(Y))/\sqrt{\{\Sigma\}_{jj}}$ is calculated at each test case point for each response variable, and the mean square error (MSE) of this quantity estimated over the up to 250 simulated data sets where the constrained model fits. Note that this means that only cases where a model fit more than once will be included in the results below. These estimated mean square errors are then averaged over the 1000 test cases and the p response variables. The average for the constrained model is subtracted from that for the unconstrained model, so that if the difference is positive then the constrained model has a better estimated average mean square error. In addition, confidence intervals are constructed around $\hat{y}$ and compared to $E(Y)$. The observed inclusion rates are also averaged over all p responses and 1000 test cases.

The results for MSE difference are fitted well by a weighted (by n) ANOVA, including interactions between p and q and between underlying model and fitted model. This shows that the type of Σ had no effect, while the constrained models performed slightly worse with $p = 2$. This is largely a result of the small number of cases where a more constrained model apparently fit a less constrained underlying model, in other words a result of type II error in model selection. The results for MSE are shown graphically in figure 1. Each sector of the plot is an underlying model, each column within the sector a fitted model and each plotted point a single combination of p, q, Σ. If the fitted constrained model has lower MSE than the unconstrained model then the points plot above zero. Hence from the j-th column of the j-th sector ($j = 1..5$) we see that in each case the fitted model equivalent to the true model does indeed have lower MSE than the unconstrained model. Similarly, when the underlying model is a special case of the fitted model we would expect the fitted model to perform well. This is the case for all constrained models when the true model is SEE, for fitted S and E when the true model is SE and for fitted E when the true model is EE.

When model SEE is true all of the constrained models have consistently lower MSE than the unconstrained, as expected. When SE is true both SE and S do consistently well, with E also offering a slight improvement, while EE and SEE rarely fit but when they do they do worse than the unconstrained model. When S is true, model S is consistently better while the others rarely fit, but when they do fit they tend to be worse. Surprisingly, E often fits but does slightly worse. When

EE is true EE and E do consistently much and slightly better, respectively, than the unconstrained model. However, SE and S fit quite often but usually do worse, while SEE also fits quite often but usually does better. When E is true there is a small but consistent advantage in fitting model E, while the others fit rarely but when they do are consistently worse. When the underlying model is unconstrained only S and E fit very often, doing on average a little worse.

The cases with poor MSE are nearly all for $p = 2$ where the model fitted only rarely, and in the equivalent $p = 4$ case did not fit at all. This is related to the point about type II error above. The fact that model S fits quite often, but can do badly, is probably due to the simulated β being constructed so that the stationary points are all in or close to the region of experimentation, and hence close to each other.

An encouraging feature is that model E consistently produces MSEs slightly better (for underlying SEE, SE, EE and E) than the unconstrained model, or only slightly worse (for underlying S, U).

A similar simple ANOVA works well for the 90% CI inclusion rates, and suggests that rates are very slightly higher for $p = 4$, $q = 6$ and $\Sigma = I_p$. Figure 2 shows that in most cases where the model fitted a large number of times the true inclusion rate was close to the nominal 90%, on average slightly under. However, this is also true for the fit of the unconstrained model using standard theory, as shown by the rightmost set of points. Where the inclusion rate is very low this is nearly always when a model did not fit very often. The obvious exception to this is, again, the poor performance when model S appears to fit an underlying true model EE. Unsurprisingly, poor performance in inclusion rate usually accompanies poor performance in MSE, as failures in both are usually due to bias.

7 Conclusions

The models proposed here can be used to reduce the number of parameters in a multivariate regression model while increasing its interpretability, and puts related work into a common model-based format. The results in section 6 suggest that if a constrained model seems to fit then it is unlikely to go badly wrong, and in particular model E, the CPC-type model, is consistently competitive with the unconstrained model in terms of performance, at least for the small p and q reported here.

The basic ideas can be expanded, for example by methods analogous to Partial CPC (Flury 1988,

chap. 6), so that different models can be fit to different subsets of variables. This is, to a certain extent, what was done in section 5, with E fitted to variables 1–6 and U to variables 7–9. The models could also be applied to the A canonical form, where the surface is not centred at the stationary point.

Automatic Differentiation can be used to provide derivatives to any optimisation problem, and hence to any maximum likelihood problem where the log-likelihood can be written as a computer routine. It avoids the need to calculate the derivatives directly, with all the opportunities for mistakes in both derivation and coding that this would entail. The package MAD allows this to be done very easily within MATLAB (appendix B).

APPENDIX A: DERIVATIVES

In both cases below $\boldsymbol{\theta}$ contains all of the model parameters in some user-defined order, and is equivalent to the MATLAB vector `x` in section 3. The only practical difficulty is ensuring that the ordering of the parameters in $\boldsymbol{\theta}$ is respected.

A.1 From (16) C is constructed by expressing the constraints in the form $c_j(\boldsymbol{\theta}) = 0$, and then differentiating each constraint $c_j(\boldsymbol{\theta})$ with respect to each element of $\boldsymbol{\theta}$. Since the constraints only involve eigenvectors $\boldsymbol{m}_k$ or constant vectors $\boldsymbol{c}$, elements of C corresponding to other parameters are zero. The non-zero elements of C can be obtained from

$$\frac{\partial(\boldsymbol{m}_k^T\boldsymbol{m}_k - 1)}{\partial \boldsymbol{m}_k} = 2\boldsymbol{m}_k \quad , \qquad \frac{\partial(\boldsymbol{m}_k^T\boldsymbol{m}_l - 0)}{\partial \boldsymbol{m}_k} = \boldsymbol{m}_l \quad , \qquad \frac{\partial(\boldsymbol{c}^T\boldsymbol{c} - p)}{\partial \boldsymbol{c}} = 2\boldsymbol{c}$$

This is evaluated at $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}$ so that the maximum likelihood estimates are then plugged into these expressions.

A.2 Only the derivatives for the unconstrained model are given here, since the appropriate changes required for the constrained models are all straightforward.

If y_{ru} is the value of the u-th response at any point $\boldsymbol{x}_r$ then we require $\boldsymbol{f}_r^{(u)} = \partial y_{ru}/\partial\boldsymbol{\theta}$, the derivative with respect to all of the model parameters. In any parameterisation the first elements of $\boldsymbol{f}_r^{(u)}$ are $\partial y_{ru}/\partial\boldsymbol{\beta}_0^{(j)} = \mathbf{1}$ if $u = j$ and zero otherwise. If the model is parameterised as (4) then the other elements of $\boldsymbol{f}_r$ are

$$\frac{\partial y_{ru}}{\partial \boldsymbol{\beta}_1^{(j)}} \quad = \quad \boldsymbol{x}_r \qquad \text{if } u = j \text{ or } 0 \text{ if } u \neq j$$

$$\begin{aligned}
\frac{\partial y_{ru}}{\partial \lambda_k^{(j)}} &= \boldsymbol{x}_r^T \boldsymbol{m}_k^{(j)} \boldsymbol{m}_k^{(j)T} \boldsymbol{x}_r \quad \text{if } u = j \text{ or } 0 \text{ if } u \neq j \\
\frac{\partial y_{ru}}{\partial \boldsymbol{m}_k^{(j)}} &= 2\lambda_k^{(j)} (\boldsymbol{x}_r \boldsymbol{x}_r^T) \boldsymbol{m}_k^{(j)} \quad \text{if } u = j \text{ or } 0 \text{ if } u \neq j
\end{aligned}$$

while for the model parameterised as (5)

$$\begin{aligned}
\frac{\partial y_{ru}}{\partial \boldsymbol{x}_0^{(j)}} &= -2\boldsymbol{x}_r^T M^{(j)} \Lambda^{(j)} M^{(j)T} \quad \text{if } u = j \text{ or } 0 \text{ if } u \neq j \\
\frac{\partial y_{ru}}{\partial \lambda_k^{(j)}} &= \boldsymbol{x}_r^T \boldsymbol{m}_k^{(j)} \boldsymbol{m}_k^{(j)T} (\boldsymbol{x}_r - 2\boldsymbol{x}_0^{(j)}) \quad \text{if } u = j \text{ or } 0 \text{ if } u \neq j \\
\frac{\partial y_{ru}}{\partial \boldsymbol{m}_k^{(j)}} &= \lambda_k^{(j)} (\boldsymbol{x}_r (\boldsymbol{x}_r - 2\boldsymbol{x}_0^{(j)})^T + (\boldsymbol{x}_r - 2\boldsymbol{x}_0^{(j)}) \boldsymbol{x}_r^T) \boldsymbol{m}_k^{(j)} \quad \text{if } u = j \text{ or } 0 \text{ if } u \neq j
\end{aligned}$$

In the various constrained models some of the $^{(j)}$ disappear, so that derivatives are not zero when $u \neq j$, but otherwise the above results still apply.

The resulting $\boldsymbol{f}_r^{(u)}$ is used directly in constructing confidence and prediction intervals (18). Similarly the matrix F in (19) is the matrix whose $(u-1)p + i$-th row is $\boldsymbol{f}_i^{(u)}$, the derivative for the u-th response at the i-th design point.

These can also be calculated using AD, similarly to the `Emad` routine in section 3. The vector of maximised parameter values is converted to type `fmad`, the fitted values calculated and their derivatives extracted. In all cases investigated the two methods agreed to at least 12 decimal places.

APPENDIX B: COMPUTING

All calculations were performed in MATLAB release 13 on SUN UNIX workstations.
The MATLAB code is available from the first author at T.J.Ringrose@cranfield.ac.uk.
MAD is now available as part of the TOMLAB optimisation package (Forth and Edvall 2004).

REFERENCES

Aitchison, J. and Silvey, S.D. (1958), "Maximum likelihood estimation of parameters subject to restraints", *Annals of Statistics*, **29**, 813–828.

Anderson, T.W. (1984), *An Introduction to Multivariate Statistical Analysis*, Wiley, New York.

Bischof, C.H., Carle, A., Khademi, P. and Mauer, A. (1996), "ADIFOR 2.0: Automatic differentiation of Fortran 77 programs", *IEEE Computational Science and Engineering*, **3**, 18–32.

Bischof, C.H., Roh, L. and Mauer, A. (1997), "ADIC – an extensible automatic differentiation tool for ANSI-C", *Software – Practice and Experience*, **27**, 1427–1456.

Bischof, C.H., Bücker, H.M., Lang, B., Rasch, A. and Vehreschild, A. (2002), "Combining source transformation and operator overloading techniques to compute derivatives for MATLAB programs", In *Proceedings of the Second IEEE International Workshop on Source Code Analysis and Manipulation (SCAM 2002)*, 65-72. IEEE Computer Society.

Bisgaard, S. and Ankenman, B. (1996), "Standard errors for the eigenvalues in second-order response surface models", *Technometrics*, **38**, 238–246.

Bockenholt, U. (1989), "Analyzing optima in the exploration of multiple response surfaces", *Biometrics*, **45**, 1001–1008.

Box, G.E.P. and Draper, N.R. (1987), *Empirical Model Building and Response Surfaces*, Wiley, New York.

Carter, W.H., Chinchilli, V.M. and Campbell, E.D. (1990), "A large-sample confidence region useful in characterising the stationary point of a quadratic response surface", *Technometrics*, **32**, 425–435.

Coleman, T., Branch, M.A. and Grace, A. (1999), *Optimization Toolbox for use with Matlab: User's Guide Version 2*, The Mathworks Inc., Natick.

Flury, B. (1988), *Common Principal Components and Related Multivariate Models*, Wiley, New York.

Forth, S.A. (2001), *User guide for MAD - a Matlab automatic differentiation toolbox*, AMOR Technical Report 2001/5, Cranfield University, RMCS Shrivenham.

Forth, S.A. and Edvall, M.M. (2004), *User Guide for MAD - MATLAB Automatic Differentiation Toolbox TOMLAB/MAD, Version 1.1 The Forward Mode*, TOMLAB Optimisation Inc, San Diego.

Forth, S.A. and Ketzscher, R. (2004), "High-Level Interfaces for the MAD (Matlab Automatic Differentiation) Package", *In* Neittaanmäki, P., Rossi, T., Korotov, S., Oñate, E., Périaux, J. and Knörzer, D. (eds.) *4th European Congress on Computational Methods in Applied Sciences and Engineering* (CD only), University of

Jyväskylä.

Friedman, J.H. (1989), "Regularised discriminant analysis", *Journal of the American Statistical Association*, **84**, 165–175.

Giering, R. and Kaminski, T. (1998), "Recipes for adjoint code construction", *ACM Trans. Math. Software*, **24**, 437–474.

Gilmour, S.G. and Ringrose, T.J. (1999), "Controlling processes in food technology by simplifying the canonical form of fitted response surfaces", *Applied Statistics*, **48**, 91–101.

Griewank, A., Juedes, D., Mitev, H., Utke, J., Vogel, O. and Walther, A. (1999), *ADOL-C: A package for the automatic differentiation of algorithms written in C/C++*, Technical Report, Technical University of Dresden Institute of Scientific Computing and Institute of Geometry.

Griewank, A. (2000), *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*, SIAM, Philadelphia.

Khuri, A.I. and Cornell, J.A. (1996), *Response Surfaces: Designs and Analyses*, Dekker, New York.

Lawley, D.N. (1975), "The inversion of an augmented information matrix occuring in factor analysis", *Proceedings of the Royal Society of Edinburgh*, **75**, 171–178.

Monahan, J. (2001), *Numerical Methods of Statistics*, CUP, Cambridge.

Myers, R.H. and Montgomery, D.C. (1995), *Response Surface Methodology: Process and Product Optimization using Designed Experiments*, Wiley, New York.

Pryce, J.D. and Reid, J.K. (1998), *ADO1, a Fortran 90 code for automatic differentiation*, Technical report RAL-TR-1998-057, Rutherford Appleton Laboratory, Didcot, England.

Reinsel, G.C. and Velu, R.P. (1998), *Multivariate Reduced Rank Regression*, Springer, New York.

Ringrose, T.J. and Benn, D.I. (1997), "Confidence regions for fabric shape diagrams", *Journal of Structural Geology*, **19**, 1527–1536.

Ringrose, T.J. and Forth, S.A. (2002), "Improved Fitting of Constrained Multivariate Regression Models using Automatic Differentiation", *In* Hardle, W. and Ronz, B. (eds.) *COMPSTAT: Proceedings in Computational Statistics. 15th Symposium held in Berlin, Germany, 2002*, Physica-Verlag, Heidelberg. pp 383–388.

Sakamoto, Y., Ishiguro, M. and Kitagawa, G. (1986), *Akaike Information Criterion Statistics*, Reidel, Dordrecht.

Seber, G.A.F. and Wild, C.J. (1989), *Nonlinear Regression*, Wiley, New York.

Shampine, L.F., Ketzscher, R. and Forth, S.A. (2003), *Using AD to solve BVPs in Matlab*, AMOR Technical Report 2003/4, Cranfield University, RMCS Shrivenham.

Silvey (1975), *Statistical Inference*, Chapman and Hall, London.

Skaug, H.J. (2002), "Automatic differentiation to facilitate maximum likelihood estimation in nonlinear random effects models", *Journal of Computational and Graphical Statistics*, **11**, 458-470.

Verma, A. (1998), "ADMAT: automatic differentiation in MATLAB using object oriented methods", In *SIAM Interdisciplinary Workshop on Object Oriented Methods for Interoperability*, 174-183, SIAM, New York.

Tables

Unconstrained						
term	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
x_1	0.9944	-0.0100	0.4989	-9.0000	-8.5167	-25.1111
x_2	-1.4556	-0.6233	-1.1967	16.1056	29.7222	73.6667
x_3	0.7556	0.3256	0.6661	-7.7389	-15.9778	-37.3500
x_1^2	0.3818	0.1450	0.3823	-7.0591	2.1364	16.7682
x_2^2	1.1318	0.5050	0.7423	-13.7091	-32.2136	-88.0318
x_3^2	0.8318	0.0950	0.5273	-4.8091	-3.6136	-2.9818
x_1x_2	0.4799	0.1381	0.2127	-4.6247	-6.4123	-19.2006
x_1x_3	0.1534	0.0827	0.1273	1.3997	-1.7543	-2.7744
x_2x_3	-0.3451	-0.1702	-0.3556	-5.9330	0.4043	4.4910
Constrained (E)						
term	Y_1	Y_2	Y_3	Y_4	Y_5	Y_6
x_1	0.9944	-0.0100	0.4989	-8.9998	-8.5157	-25.1084
x_2	-1.4556	-0.6233	-1.1967	16.1058	29.7231	73.6691
x_3	0.7556	0.3256	0.6661	-7.7390	-15.9780	-37.3506
x_1^2	0.6615	0.2045	0.4712	-8.4684	-8.3176	-13.1695
x_2^2	1.1285	0.4056	0.8240	-6.8082	-16.2826	-43.1325
x_3^2	0.5555	0.1349	0.3566	-10.3012	-9.0901	-17.9417
x_1x_2	0.1489	0.0723	0.1243	1.0271	-1.6584	-5.6008
x_1x_3	0.1109	0.0703	0.1162	1.7599	0.5288	3.7398
x_2x_3	-0.4421	-0.2110	-0.3637	-2.8242	5.3199	18.4069

Table 1:

Unconstrained									
	Y_1			Y_2			Y_3		
	1st	2nd	3rd	1st	2nd	3rd	1st	2nd	3rd
$\hat{\lambda}$	1.253	0.809	0.283	0.531	0.168	0.046	0.851	0.510	0.291
	(0.417)	(0.416)	(0.427)	(0.199)	(0.149)	(0.139)	(0.236)	(0.174)	(0.178)
x_1	-0.223	0.310	0.924	-0.155	-0.847	-0.508	-0.138	0.608	0.782
	(0.225)	(0.402)	(0.173)	(0.176)	(0.637)	(1.085)	(0.222)	(0.793)	(0.640)
x_2	-0.916	0.259	-0.307	-0.972	0.040	0.230	-0.879	0.288	-0.379
	(0.196)	(0.601)	(0.296)	(0.041)	(0.415)	(0.183)	(0.237)	(0.713)	(0.404)
x_3	0.334	0.915	-0.226	0.175	-0.530	0.830	0.456	0.740	-0.495
	(0.562)	(0.187)	(0.400)	(0.166)	(1.049)	(0.685)	(0.484)	(0.465)	(0.786)
	Y_4			Y_5			Y_6		
	1st	2nd	3rd	1st	2nd	3rd	1st	2nd	3rd
$\hat{\lambda}$	-3.372	-7.087	-15.119	2.562	-3.742	-32.511	17.762	-3.051	-88.956
	(5.103)	(5.625)	(6.189)	(12.217)	(12.350)	(12.539)	(35.417)	(35.770)	(35.666)
x_1	0.373	-0.895	-0.247	-0.986	0.143	-0.092	0.993	0.078	0.090
	(1.060)	(0.404)	(0.357)	(0.095)	(0.641)	(0.100)	(0.038)	(0.453)	(0.093)
x_2	-0.332	0.120	-0.936	0.092	-0.009	-0.996	-0.092	0.018	0.996
	(0.357)	(0.651)	(0.119)	(0.099)	(0.123)	(0.009)	(0.094)	(0.111)	(0.009)
x_3	0.867	0.431	-0.253	0.143	0.990	0.004	-0.076	0.997	-0.025
	(0.544)	(0.993)	(0.281)	(0.644)	(0.093)	(0.104)	(0.455)	(0.035)	(0.100)

Table 2:

Constrained (E)				
$\hat{\lambda}$	Y_1	1.2089 (0.2468)	0.6848 (0.3282)	0.4517 (0.3015)
	Y_2	0.4441 (0.0985)	0.2190 (0.1341)	0.0819 (0.1228)
	Y_3	0.8903 (0.1494)	0.4952 (0.2080)	0.2663 (0.1874)
$\hat{\lambda}$	Y_4	-6.2902 (3.6187)	-8.1155 (4.5670)	-11.1721 (4.4155)
	Y_5	-17.2443 (5.9464)	-8.2359 (7.5468)	-8.2101 (7.1432)
	Y_6	-46.4528 (16.9943)	-12.5159 (21.7059)	-15.2750 (20.4032)
$\boldsymbol{x}$	x_1	0.0970 (0.2616)	0.9325 (0.0706)	0.3480 (0.0988)
	x_2	0.9453 (0.1331)	0.0231 (0.0459)	-0.3254 (0.0978)
	x_3	-0.3115 (0.1086)	0.3605 (0.1287)	-0.8792 (0.2507)

Table 3:

Figures

Figure 1: Estimated mean square error for unconstrained model minus that for the constrained model, for cases where the constrained model fit at least twice. Each point is a combination of p, q and Σ. Point labelled + where constrained model fit 2–10 times, $\times$ if fit 11–30 times and $*$ if fit more than 30 times.

Figure 2: Inclusion rates for calculated 90% confidence intervals, averaged over all response variables and over all test cases, for cases where the constrained model fit at least twice. Each point is a combination of p, q and Σ. Point labelled + where constrained model fit 2–10 times, $\times$ if fit 11–30 times and $*$ if fit more than 30 times.

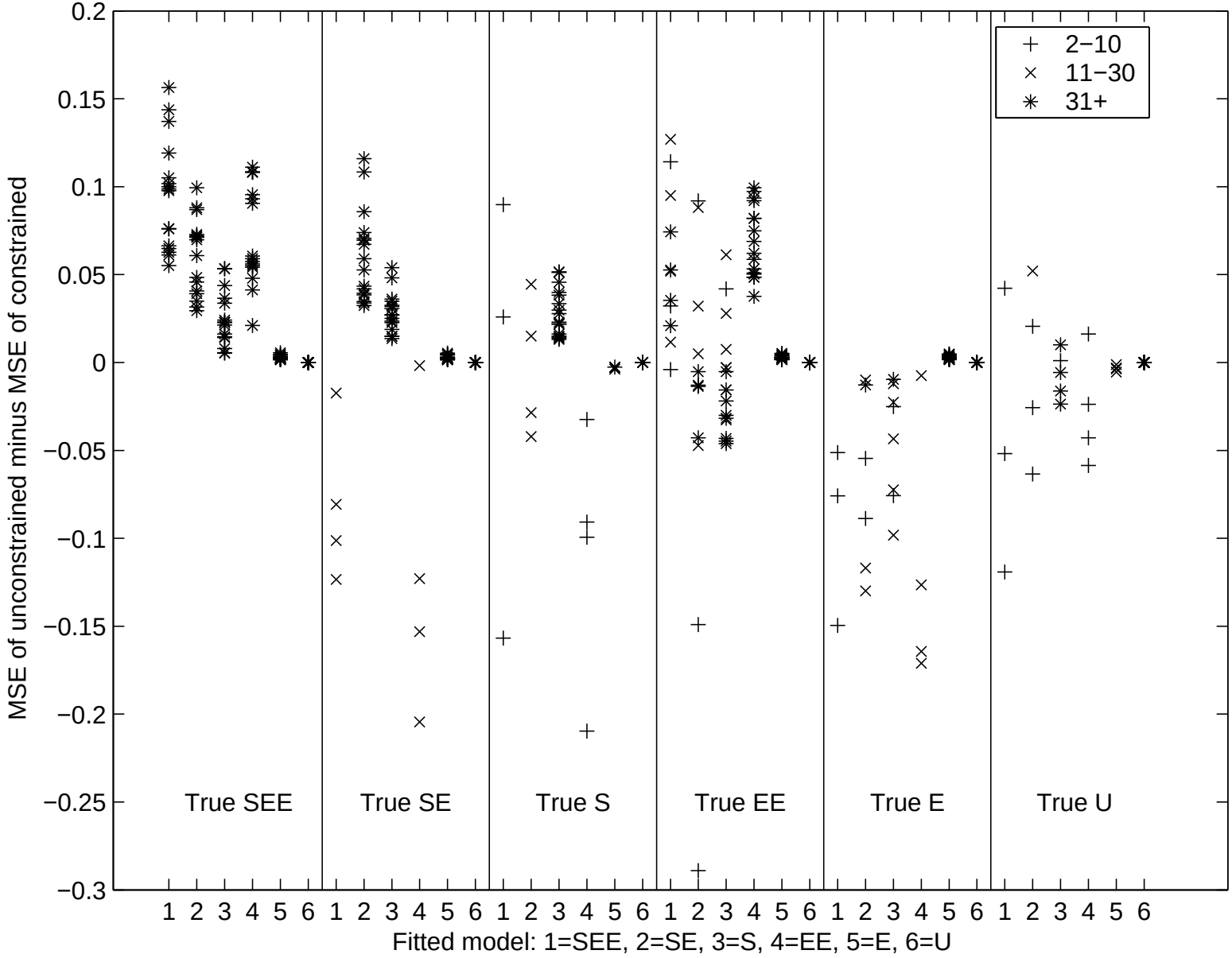
MSE of unconstrained minus MSE of constrained
0.2
0.15
0.1
0.05
0
−0.05
−0.1
−0.15
−0.2
−0.25
−0.3
2−10
11−30
31+
True SEE
True SE
True S
True EE
True E
True U
1 2 3 4 5 6
Fitted model: 1=SEE, 2=SE, 3=S, 4=EE, 5=E, 6=U

Figure 1

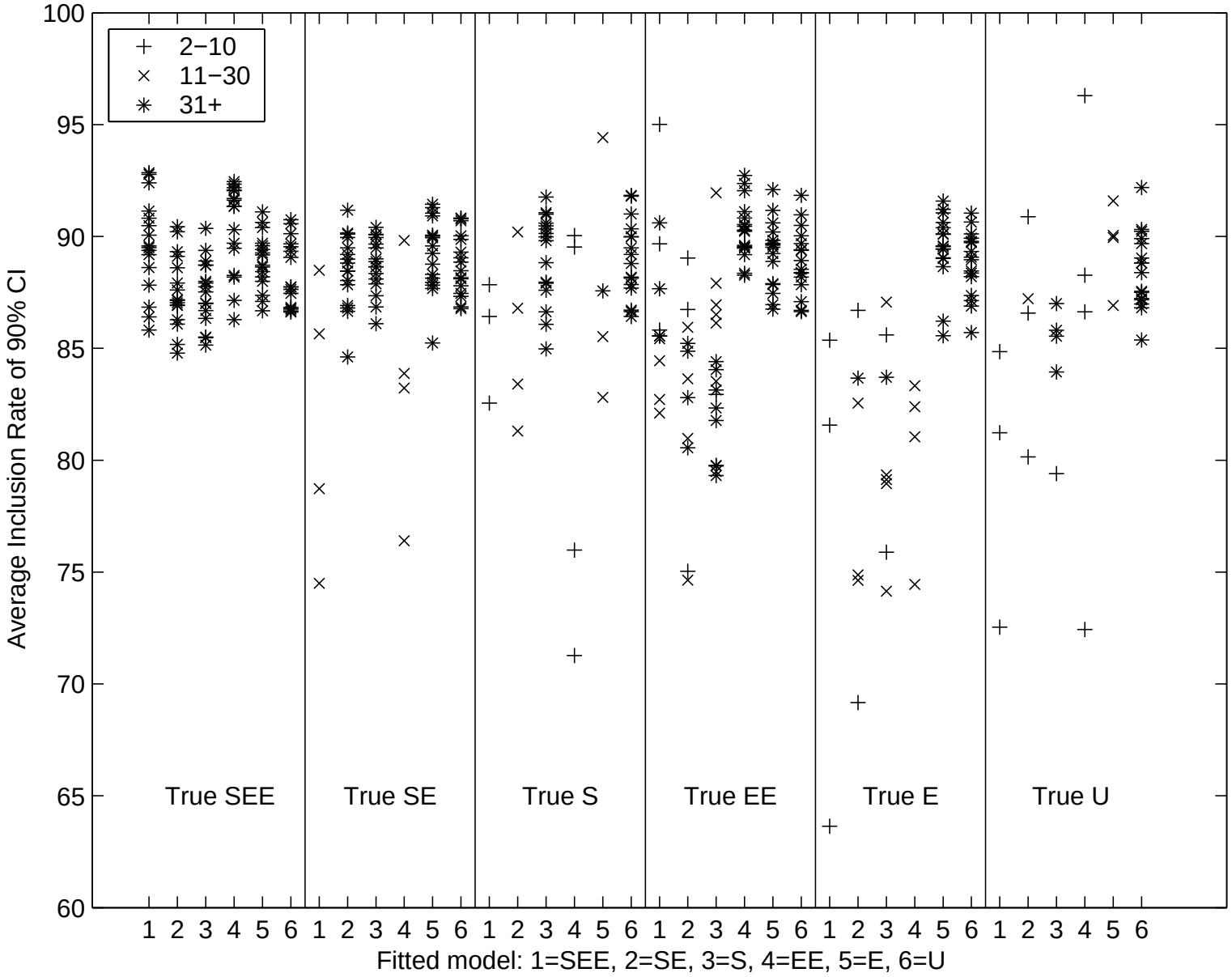
2−10
11−30
31+
Average Inclusion Rate of 90% CI
100
95
90
85
80
75
70
65
60
True SEE
True SE
True S
True EE
True E
True U
1 2 3 4 5 6
Fitted model: 1=SEE, 2=SE, 3=S, 4=EE, 5=E, 6=U

Figure 2