Available online at www.sciencedirect.com
ScienceDirect
Computational Statistics and Data Analysis 52 (2008) 4203–4224
COMPUTATIONAL STATISTICS & DATA ANALYSIS
www.elsevier.com/locate/csda

On the estimation of a large sparse Bayesian system: The Snaer program

Dmitry Danilov[a], Jan R. Magnus[b,*]

[a] *Eurandom, Eindhoven University of Technology, P.O. Box 513, 5600 MB Eindhoven, The Netherlands*
[b] *CenTER and Department of Econometrics & OR, Tilburg University, P.O. Box 90153, 5000 LE Tilburg, The Netherlands*

Received 26 February 2007; received in revised form 18 February 2008; accepted 19 February 2008
Available online 4 March 2008

Abstract

The Snaer program calculates the posterior mean and variance of variables on some of which we have data (with precisions), on some we have prior information (with precisions), and on some prior indicator ratios (with precisions) are available. The variables must satisfy a number of exact restrictions. The system is both large and sparse. Two aspects of the statistical and computational development are a practical procedure for solving a linear integer system, and a stable linearization routine for ratios. The numerical method for solving large sparse linear least-squares estimation problems is tested and found to perform well, even when the $n \times k$ design matrix is large ($nk = O(10^8)$).

1. Introduction

Suppose we are given a system of latent variables to be estimated. The information consists of incomplete data (with precisions), priors on a subset of the variables or linear combinations thereof (with precisions), and exact linear restrictions. Some of the priors are the so-called indicator ratios and hence nonlinear.

The system that we are interested in is large and sparse. In this context, a system with 2^6 variables and 2^{12} observations, thus giving 2^{18} ≈260,000 entries in the design matrix, is *not* considered to be large. On the other hand, a system with 2^{11} variables and 2^{13} observations, thus giving $2^{24} \approx 16.8$ million entries in the design matrix, is considered to be large.

Our system is sparse, because information is often available on one variable at a time, and restrictions are often definitions involving only a small number of variables. A matrix is sparse when it has many structural zeros. If a $p \times n$ matrix possesses z structural zeros, then the matrix can be stored as a $(pn - z) \times 3$ matrix, where the ith row contains the row-index, column-index, and value of the ith nonzero entry. This is often useful if storage space is more important than access speed. The number $s := 1 - z/(pn)$ is called the sparsity of the matrix.

In this paper we analyze how we can estimate such a large and sparse system efficiently and accurately. The analysis contains both statistical and computational aspects. The statistical problem was first considered by

* Corresponding author.
E-mail addresses: danilov@eurandom.tue.nl (D. Danilov), magnus@uvt.nl (J.R. Magnus).

0167-9473/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.csda.2008.02.019

Magnus et al. (2000), but the sparse and large-dimension aspects were not treated in that paper. In addition, a number of their proposed solutions – especially for the treatment of indicator ratios – do not work well in large dimensions. Also, the Bayesian solution involves difficult inversions that can be avoided by rewriting the system in a different format. The purpose of this paper is to describe both the statistical and the computational aspects, and to perform tests for the stability, accuracy, and speed of the proposed routines.

The concept of sparsity is now widespread in modern numerical methods, and huge sparse matrices often appear in science and engineering when solving problems of mathematical physics and computational chemistry. Recently, large sparse matrices have become important in economics, for example in studying longitudinal samples of over one million workers from more than 500,000 employing firms; see Abowd et al. (1999, 2002), using an algorithm due to Dongarra et al. (1991). There exists a large literature on sparse matrices. Some recent references can be found in Björck (1996, Chapters 6 and 7) and Pauletto (1997).

The search for accurate and fast software for large and sparse linear estimation problems is driven by the difficulties encountered in estimating national accounts data, especially in developing countries, where one may encounter 5000 or more variables and 20,000 observations. The relevant matrices in such applications are sparse but not structured, that is they do not have a Kronecker, Toeplitz, or other well-known structure. Although iterative methods have been used quite successfully for general sparse matrices, our experience with these methods has been uniformly bad. This made it necessary to look for alternative methods. The lack of structure is exemplified in Fig. 1, where we present four recent cases under investigation. The X-matrix for the Mozambique national accounts has dimensions $n = 77$ and $k = 59$, and contains 171 nonzero entries, so that the sparsity is $s = 3.76\%$. A somewhat larger data set is given in 'CBS-small', based on data from Statistics Netherlands (Centraal Bureau voor de Statistiek, CBS). This concerns data with a high degree of aggregation where X has dimensions $n = 1393$ and $k = 698$, and contains 3884 nonzero entries with $s = 0.40\%$. Slightly less aggregated is the 'CBS-medium' data set with dimensions $n = 2506$ and $k = 1164$, containing 9085 nonzero entries and sparsity $s = 0.31\%$. The largest data set is 'CBS-large', representing the disaggregated data. This has dimensions $n = 13{,}896$ and $k = 10{,}810$, and contains 116,571 nonzero entries with $s = 0.08\%$. It is clear that there is no easily recognizable structure of the nonzeros in these data sets. The Snaer (Software for National Accounts Estimation and Reconciliation) program, developed for this purpose, thus calculates the posterior mean and variance of variables, some of which are given as data (with precisions), on some we have prior information (with precisions) or prior indicator ratios (with precisions), and exact restrictions apply.

Our new theory for the estimation of national accounts has a number of characteristics:

- It specifies data and their precisions, in addition to priors and their precisions;
- It allows indicator ratios (with precisions) as input priors;
- It takes full account of all accounting identities;
- Solutions are continuous rather than discrete;
- It allows for multiple priors on variables or linear combinations of variables;
- The posterior estimates take all prior and data input into account and come with precisions;
- The system is transparent, flexible, and fast;
- Sensitivity analysis is easy to perform, and alternative scenarios may be analyzed;
- It can be applied without major alterations in existing compilation methods.

The first major application of the methodology and the software was in Angola; see Magnus and van Tongeren (in preparation). But also applications in developed countries are now underway. These applications concern not only large economic data sets for national accounts, but also input–output analysis, flow-of-funds analysis, and the integration of quarterly and annual accounts.

There are many other potential applications of the proposed method, outside national accounting, for example in large employment matrices, material flow analysis in environmental accounting, socio-economic and other satellite accounts, integration of quarterly and annual accounts, business accounting, and demographic projections.

The plan of this paper is as follows. We formally describe the set-up in Section 2. In Sections 3 and 4 we present two solutions: first a Bayesian solution to a Bayesian problem; then an equivalent least-squares solution, which involves less inversions and is much easier to estimate. The implementation of the theory together with some extensions and refinements is discussed in Section 5. An important aspect of our theory is the possibility to include indicator ratios as priors. These ratios must be linearized, but this can be done in many ways. Two solutions and their merits are presented in Section 6. In Section 7 we present a flow chart of our algorithm. In Sections 8 and 9 we perform

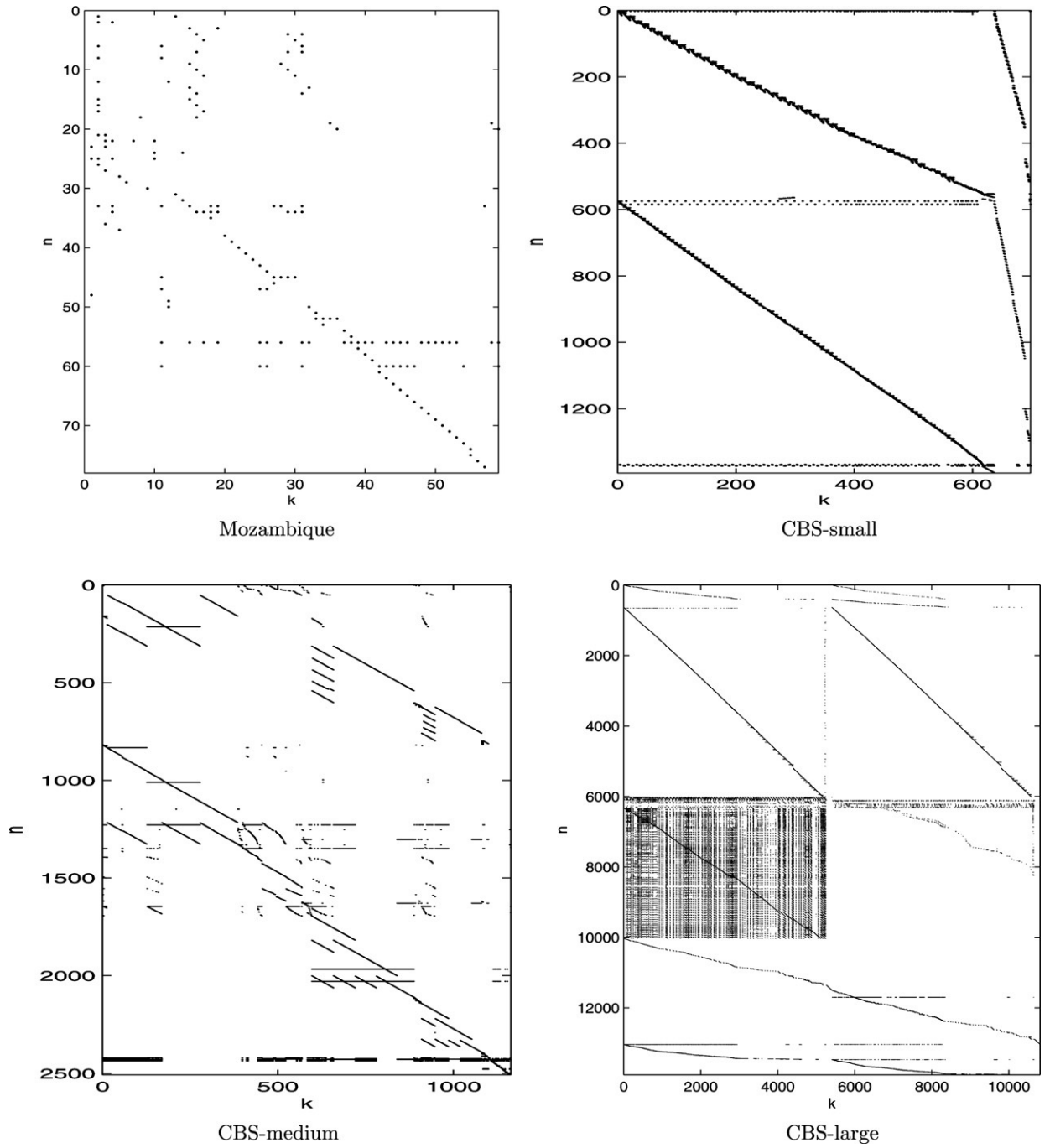

Fig. 1. Patterns of nonzero entries in sparse matrices.

Monte Carlo experiments to test the accuracy and speed of our proposed sparse least-squares routine (which also calculates variances). Section 10 concludes and discusses possible extensions.

2. Set-up: Data, priors, and restrictions

We consider a vector x of n latent variables to be estimated. Data are available on p components (or linear combinations of components) of x. Let d_1 denote the $p \times 1$ data vector, generated by the measurement equation

$$d_1|x \sim \mathrm{N}_p(D_1 x, \Sigma_1). \tag{1}$$

Typically, the $p \times n$ matrix D_1 is a selection matrix, say $D_1 = (I_p, 0)$, so that $D_1 x$ is a subvector of x, but this is not required. Neither is it required that the matrix D_1 has full row-rank; in fact, p may be larger than n. Measurements are unbiased in the sense that $\mathrm{E}(d_1|x) = D_1 x$. The $p \times p$ matrix Σ_1 denotes a positive definite (hence nonsingular) variance matrix, typically (but not necessarily) diagonal.

In addition to the p data, we have access to two further pieces of information: prior views concerning the latent variables or linear combinations thereof, and deterministic (accounting) constraints. In particular, we have m_1 random priors:

$$A_1 x \sim \mathrm{N}_{m_1}(h_1, H_1) \tag{2}$$

and m_2 exact restrictions:

$$A_2 x = h_2 \quad \text{(a.s.)}, \tag{3}$$

in total $m := m_1 + m_2$ pieces of prior information.

We shall assume that the $m_1 \times m_1$ matrix H_1 is positive definite (hence nonsingular) and that the $m_2 \times n$ matrix A_2 has full row-rank m_2. This guarantees that the exact restrictions are linearly independent and thus form a consistent set of equations. Neither of these two restrictions constitutes any loss of generality, because if H_1 is singular some of the priors can be interpreted as exact restrictions, and if A_2 does not have full row-rank, we simply throw away the redundant restrictions, obviously without loss of information.

In order to identify all n variables from the information (data and priors) we need at least n pieces of information: $m + p \geq n$. But this is not sufficient for identification, because some of the information may be on the same variables. The condition

$$\mathrm{rk}\begin{pmatrix} D_1 \\ A_1 \\ A_2 \end{pmatrix} = n \tag{4}$$

is necessary and sufficient for identification.

3. Bayesian solution

In order to estimate the n latent variables x, we define

$$A := \begin{pmatrix} A_1 \\ A_2 \end{pmatrix}, \qquad h := \begin{pmatrix} h_1 \\ h_2 \end{pmatrix}, \qquad H := \begin{pmatrix} H_1 & 0 \\ 0 & 0 \end{pmatrix},$$

and we make the additional assumption that

$$\mathrm{rk}(A) = m$$

which, of course, implies that both A_1 and A_2 have full row-rank. In fact, as shown in Danilov and Magnus (2007), the rank condition on A is not a binding restriction, because we can always move priors to data (and vice versa). This condition will be dropped in the next section.

Since $\mathrm{rk}(A) = m$ we have $m \leq n$. If $m = n$, then all variables are identified. If $m < n$, we define a semi-orthogonal $n \times (n - m)$ matrix L such that $AL = 0$ and $L'L = I_{n-m}$. The condition

$$\mathrm{rk}(D_1 L) = n - m$$

is then an alternative and equivalent identifiability condition to (4), because the definition of L implies that

$$\mathrm{rk}\begin{pmatrix} A \\ D_1 \end{pmatrix} = \mathrm{rk}(A) + \mathrm{rk}(D_1 L).$$

From Theorem 1 of Magnus et al. (2000), we know that the posterior distribution of x is given by

$$x|d_1 \sim \mathrm{N}_n(\mu, V),$$

where

$$\mu = A^+h - (A^+HA^{+\prime} + CK)D_1'\Sigma_0^{-1}(D_1A^+h - d_1) \tag{5}$$

and

$$V = A^+HA^{+\prime} - A^+HA^{+\prime}D_1'\Sigma_0^{-1}D_1A^+HA^{+\prime} + CKC', \tag{6}$$

and the following definitions have been employed:

$$\Sigma_0 := \Sigma_1 + D_1A^+HA^{+\prime}D_1', \qquad C := I_n - A^+HA^{+\prime}D_1'\Sigma_0^{-1}D_1,$$

and

$$K := \begin{cases} L(L'D_1'\Sigma_0^{-1}D_1L)^{-1}L' & \text{if } m < n, \\ 0 & \text{if } m = n. \end{cases}$$

The matrix $A^+ := (A'A)^{-1}A'$ denotes the Moore–Penrose inverse of A.

This is a useful theoretical result, but Eqs. (5) and (6) do not lend themselves well to applications where the dimensions are large. Hence we seek an alternative but equivalent formulation which allows us to tackle large-dimensional problems.

4. Least-squares solution

Because of the normality assumption for both data and priors, there is no mathematical or statistical difference between them (although there is a conceptual difference); see Danilov and Magnus (2007). We may therefore move all random priors to the data and consider a new 'data' vector

$$d := \begin{pmatrix} d_1 \\ h_1 \end{pmatrix}.$$

Defining

$$D := \begin{pmatrix} D_1 \\ A_1 \end{pmatrix}, \qquad \Sigma := \begin{pmatrix} \Sigma_1 & 0 \\ 0 & H_1 \end{pmatrix},$$

we may write the new measurement equation as

$$d|x \sim \mathrm{N}_{p+m_1}(Dx, \Sigma)$$

together with the priors (exact restrictions)

$$A_2x = h_2 \quad \text{(a.s.)}.$$

The Bayesian problem is then equivalent to the constrained least-squares estimation problem of estimating x from

$$d = Dx + \varepsilon, \qquad \varepsilon \sim \mathrm{N}_{p+m_1}(0, \Sigma),$$

under the restriction $A_2x = h_2$.

In many cases, an unconstrained problem is easier to solve in large dimensions than a constrained problem. Thus, let

$$A_2 = (A_{21} : A_{22}),$$

where A_{21} is an $m_2 \times (n - m_2)$ matrix and A_{22} is a nonsingular $m_2 \times m_2$ matrix. Partitioning x accordingly, we write the restriction as

$$A_{21}x_1 + A_{22}x_2 = h_2,$$

so that

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} I \\ -A_{22}^{-1}A_{21} \end{pmatrix} x_1 + \begin{pmatrix} 0 \\ A_{22}^{-1}h_2 \end{pmatrix} \equiv Qx_1 + q.$$

Let

$$B := \Sigma^{-1/2} D = \begin{pmatrix} \Sigma_1^{-1/2} D_1 \\ H_1^{-1/2} A_1 \end{pmatrix}, \qquad b := \Sigma^{-1/2} d = \begin{pmatrix} \Sigma_1^{-1/2} d_1 \\ H_1^{-1/2} h_1 \end{pmatrix}.$$

Then the constrained problem of minimizing $(d - Dx)'\Sigma^{-1}(d - Dx)$ subject to $A_2 x = h_2$ is equivalent to the unconstrained problem

$$\text{minimize } \|(b - Bq) - BQx_1\|$$

with respect to x_1. (The notation $\|v\|$ denotes the norm of a vector, that is, $\|v\| := \sqrt{v'v}$.) The solution, say μ_1, is our estimator for x_1 with associated variance $\text{var}(\mu_1) = (Q'B'BQ)^{-1}$. Since x_2 is a linear function of x_1 we can estimate the complete x-vector and obtain the full variance matrix V.

Solving a constrained least-squares problem by a two-step procedure has the advantage that the dimension of the system is much reduced. It also allows us to take optimal advantage of the integer nature of our constraints, if appropriate (see Section 5, Step 4). Furthermore, in many practical situations (such as in our national accounts projects) the first step needs to be performed only once and the results of the reduction may then be used for many constrained ordinary least-squares (OLS) problems. In particular, the A_2 matrix is usually fixed because it represents the structure of the economy, while the matrices related to A_1 are priors that will vary. Finally, letting $F := D'\Sigma^{-1}D$, the formula for the constrained estimator is

$$\mu = F^{-1}D'\Sigma^{-1}d + F^{-1}A_2'(A_2F^{-1}A_2')^{-1}(h_2 - A_2F^{-1}D'\Sigma^{-1}d),$$

which may still be computable in large systems avoiding matrix inverses. But to compute elements of the variance matrix,

$$V = \text{var}(\mu) = F^{-1} - F^{-1}A_2'(A_2F^{-1}A_2')^{-1}A_2F^{-1},$$

seems prohibitively hazardous in large systems. For all these reasons, direct application of constrained least squares does not seem appealing.

5. Implementation

Our information thus consists of p data:

$$d_1|x \sim \text{N}_p(D_1x, \Sigma_1),$$

m_1 random priors:

$$A_1x \sim \text{N}_{m_1}(h_1, H_1),$$

and m_2 exact priors:

$$A_2x = h_2 \quad \text{(a.s.)}.$$

We need to estimate this system taking into account that the dimensions may be very large and that the matrices are sparse. We proceed as follows.

Step 1: *Rank.* Two rank conditions need to be satisfied:

$$\text{rk}(A_2) = m_2, \qquad \text{rk}\begin{pmatrix} D_1 \\ A_1 \\ A_2 \end{pmatrix} = n.$$

If the rank condition on A_2 is not satisfied, then the m_2 rows of A_2 are linearly dependent. If the rank of A_2 is $m_2^* < m_2$, then there exists a set (not unique) of $m_2 - m_2^*$ rows which can be deleted from A_2, so that the new matrix A_2^* has full row-rank m_2^*. Hence, this rank deficiency can be easily resolved. But if the second rank condition is not satisfied, then there is insufficient information in the data and priors to estimate the whole system, and more information needs to be acquired. The computation of the rank of a (sparse) matrix is no trivial exercise. Several good routines are available,

for example in the NAG Fortran Library (2006) or Harwell Subroutine Library (2004). Our routine Snaer-RANK has two special features. First, it performs a preliminary (iterative) dimension reduction by looking for rows and columns with a single nonzero entry. Second, it provides information about where rank deficiencies (if any) occur, which is of great practical use in repairing the inputs.

Step 2: *Scaling.* The matrices Σ_1 and H_1 must be nonsingular. We scale by defining

$$B := \begin{pmatrix} \Sigma_1^{-1/2} D_1 \\ H_1^{-1/2} A_1 \end{pmatrix}, \qquad b := \begin{pmatrix} \Sigma_1^{-1/2} d_1 \\ H_1^{-1/2} h_1 \end{pmatrix}$$

of orders $(p + m_1) \times n$ and $p + m_1$, respectively.

Step 3: *Re-ordering.* If necessary, we re-order the x-variables such that the matrix A_{22} in the restriction

$$A_2 x = A_{21} x_1 + A_{22} x_2 = h_2$$

is a nonsingular $m_2 \times m_2$ matrix. We re-order the columns of the matrices A_2 and B accordingly. There is no unique way to select m_2 linearly independent columns of A_2. But some selections are more robust than others computationally. Also, some selections require less covariances than others in the computation of the diagonal elements of the variance matrix (Step 8). These considerations are taken into account at the selection stage.

Step 4: *Solving.* The inverse of A_{22} appears in the formulae, but in fact we only need the expressions $A_{22}^{-1} A_{21}$ and $A_{22}^{-1} h_2$. Hence, we solve the sparse matrix equation

$$A_{22} X = (-A_{21} : h_2)$$

and we write the resulting $m_2 \times (n - m_2 + 1)$ matrix X as $X = (Q_1 : q_1)$.

There exist a number of sparse routines for solving linear equations. Our sparse subroutine Snaer-SOLVE builds on the existing routines, but it contains two special features. First, not only the left-hand side matrix A_{22} but also the right-hand side matrix $(-A_{21} : h_2)$ is (or can be) sparse.

Second, there is an integer refinement possibility in the routine, based on the fact that the matrix A_{22} is often an integer matrix, that is, all nonzero elements of A_{22} are integers. The reason for this is that in many applications the restrictions are in fact definitions, such as $x_1 = x_2 + x_3 + x_4$. The nonzero entries in A_{22} are then ± 1. In the case when A_{22} is an integer matrix, the determinant $|A_{22}|$ will be an integer, and the determinant of every submatrix of A_{22} will also be an integer. Hence, the adjoint $A_{22}^{\#}$ of A_{22} is an integer matrix. Now let c be a column of $(-A_{21} : h_2)$, all of whose nonzero elements are integers, and let x be the corresponding column of the solution matrix X, so that $A_{22} x = c$. Then,

$$x = A_{22}^{-1} c = \frac{A_{22}^{\#} c}{|A_{22}|},$$

each element of which is the ratio of two integers. We know $|A_{22}|$ (integer) and x (real) and we can improve on the solution for x using the integer feature, as follows:

(a) Compute $\Delta_1 := |A_{22}|$ and $z_1 := A_{22}^{\#} c = \Delta_1 x$;
(b) Use the fact that Δ_1 and the elements of z_1 are all integers, and obtain 'exact' integer solutions Δ_2 and z_2;
(c) Now obtain an improved solution for $x = z_2/\Delta_2$, each element of which is a rational number.

The new solution is 'exact'. This 'integer smoothing' can be done for all integer columns of the matrix $(-A_{21} : h_2)$.

Step 5: *Multiplication.* Compute the matrix and vector

$$Q := \begin{pmatrix} I_{n-m_2} \\ Q_1 \end{pmatrix}, \qquad q := \begin{pmatrix} 0 \\ q_1 \end{pmatrix},$$

and the matrix and vector

$$R := BQ, \qquad r := b - Bq.$$

Step 6: *Minimization.* We need to minimize the norm

$$\|r - R x_1\|,$$

the solution of which we call μ_1. Notice that the matrix R has full column-rank $n - m_2$. The least-squares problem

$$\min_{x_1}(r - Rx_1)'(r - Rx_1)$$

can be rewritten as

$$\min_{x_1}(r'r - 2r'Rx_1 + x_1'R'Rx_1),$$

which in turn is equivalent to the quadratic minimization problem

$$\min_{x_1}\left(\frac{1}{2}x_1'Hx_1 - c'x_1\right),$$

where $H := R'R$ and $c := R'r$. There are many libraries that offer routines dealing with sparse least squares, in particular Matlab (2002), NAG Fortran Library (2006), and Harwell Subroutine Library (2004), but they are all quite different. We offer a brief comparison of these routines in Sections 7 and 8. Our Snaer-OLS routine is based on the Gould and Nocedal (1998) algorithm, which also underlies the Harwell Subroutine Library (2004) approach. The matrix H is decomposed as $H = PLDL'P'$, where P is a permutation matrix, L is unit lower triangular, and D is block-diagonal with blocks of dimension one or two.

Least-squares routines typically do not compute (elements of) the variance matrix, but the Snaer-OLS routine does, in the following way. Let $e^{(j)}$ denote the jth unit vector, that is, the jth column of I_{n-m_2}. The jth column $v^{(j)}$ of the matrix $V_{11} := (R'R)^{-1}$ can be found by minimizing $\|e^{(j)} - Hv^{(j)}\|^2$ for all j, that is by solving the quadratic minimization problem

$$\min_{v^{(j)}}\left(\frac{1}{2}v^{(j)'}H^2v^{(j)} - e^{(j)'}Hv^{(j)}\right), \quad j = 1, \ldots, n - m_2,$$

whose value at the minimum should be $-1/2$. It is, however, faster and more accurate to obtain the columns of V_{11} by minimizing $\|H^{-1/2}e^{(j)} - H^{1/2}v^{(j)}\|^2$ for all j, that is by solving the quadratic minimization problem

$$\min_{v^{(j)}}\left(\frac{1}{2}v^{(j)'}Hv^{(j)} - e^{(j)'}v^{(j)}\right), \quad j = 1, \ldots, n - m_2,$$

using the Snaer-OLS algorithm. The value at the minimum equals $-v_j^{(j)}/2$.

In practice we will not need to calculate (or store) all elements of $v^{(j)}$. To store all elements is a physical impossibility in large dimensions. We have to indicate in advance which elements need to be stored. These typically include all diagonal elements of V_{11} (the variances) and a selection of covariances. The selected covariances are either of direct interest or they are required to calculate the diagonal (or other) elements of $V_{22} := \text{var}(\mu_2)$. For example, if $x_{21} = x_{11} + x_{12}$, where x_{21} is an element of x_2 and x_{11} and x_{12} are elements of x_1, then we need $\text{cov}(\mu_{11}, \mu_{12})$ in order to calculate $\text{var}(\mu_{21})$.

Step 7: *Completion of* μ. We compute

$$\mu_2 = Q_1\mu_1 + q_1.$$

Step 8: *Completion of* V. The variance of μ is given by

$$V = \text{var}(\mu) = \begin{pmatrix} I \\ Q_1 \end{pmatrix}(R'R)^{-1}(I, Q_1').$$

But only selected elements of V are required and only selected elements of $V_{11} = (R'R)^{-1}$ are stored in Step 6. The subroutine Snaer-OLS automatically selects which elements of V_{11} need to be stored in order to calculate the required elements of V.

6. The treatment of ratios

One of the features of our program is that also ratios can be used as priors. This is of much practical use, for example in the estimation of national accounts, where many ratios are used as indicators and are assumed to be relatively stable, hence predictable.

Let x and y be two latent variables and assume information is available on the ratio $R := y/x$. If this ratio is part of an otherwise linear system, then we can either transform the linear system into a nonlinear one, or we can linearize the ratio. Especially if the system is large, the latter option is often more practical.

Thus, we consider a linearization $y - rx$ for some suitably chosen value of r. If information is available on the first two moments of R, the question then becomes how to transform these moments into moments of the linearization.

Consider first the following special case. Let r denote the expected value of R, if it exists, and suppose that R and x are independent. Then we write $y - rx = (R - r)x$, so that

$$\mathrm{E}(y - rx) = \mathrm{E}(R - r)\mathrm{E}(x) = 0$$

and

$$\mathrm{var}(y - rx) = \mathrm{E}(y - rx)^2 = \mathrm{E}(R - r)^2\mathrm{E}(x^2) = \mathrm{var}(R) \cdot (\mathrm{var}(x) + (\mathrm{E}x)^2).$$

Hence, in this special case, we easily obtain the moments of the linearization. In general, however, this is less straightforward.

6.1. Bayesian solution

Suppose we have data: $d|x \sim (x, \sigma^2)$ and a prior on $R = y/x$,

$$\pi : R \sim (r, \tau^2).$$

We wish to replace the prior on the ratio by a prior on its linearization, so we consider

$$\pi' : y - rx \sim (0, \omega^2).$$

The question is how to choose ω^2. Let us calculate the posterior moments of $y|d$ based on π and π' respectively. Based on the linearization π' we obtain the posterior moments of y given d as

$$y|d \sim (rd, r^2\sigma^2 + \omega^2). \tag{7}$$

Notice that the two pieces of information $x \sim \mathrm{N}(d, \sigma^2)$ and $y - rx \sim \mathrm{N}(0, \omega^2)$ yield $y \sim \mathrm{N}(rd, r^2\sigma^2 + \omega^2)$ if and only if x and $y - rx$ are uncorrelated, that is, if and only if $r = \mathrm{cov}(x, y)/\mathrm{var}(x)$.

Next we calculate the posterior moments of $y|d$ based on π. Following the arguments in Magnus et al. (2000), we obtain

$$\mathrm{E}(y|x) = rx, \qquad \mathrm{var}(y|x) = \tau^2 x^2.$$

Let $z := y|x$. Then the posterior moments of $y|d$ are

$$\begin{aligned} \mathrm{E}(y|d) &= \mathrm{E}_z\mathrm{E}(d|d, z) = \mathrm{E}_z\mathrm{E}(y|d, x) \\ &= \mathrm{E}_z\mathrm{E}(y|x) = \mathrm{E}_z(rx) = rd \end{aligned} \tag{8}$$

and

$$\begin{aligned} \mathrm{var}(y|d) &= \mathrm{E}_z\mathrm{var}(y|d, z) + \mathrm{var}_z(\mathrm{E}(y|d, z)) \\ &= \mathrm{E}_z\mathrm{var}(y|x) + \mathrm{var}_z(\mathrm{E}(y|x)) \\ &= \mathrm{E}_z(\tau^2 x^2) + \mathrm{var}_z(rx) \\ &= \tau^2(d^2 + \sigma^2) + r^2\sigma^2. \end{aligned} \tag{9}$$

In order to reconcile (7) with (8) and (9) we must have

$$\omega^2 = \tau^2(d^2 + \sigma^2). \tag{10}$$

In a practical situation, we know r and τ^2, but we do not necessarily know d and σ^2 (the moments of x). If we would know the mean and variance of x, then we could replace the prior on R by a prior on $y - rx$. Since we do not know the moments of x, we may use a simple iterative procedure, based on the assumption that some reasonable starting

values for $\mathrm{E}(x)$ and $\mathrm{var}(x)$ are available. We use these as proxies for d and σ^2 and obtain posterior moments of all latent variables, in particular $\mathrm{E}(x)$ and $\mathrm{var}(x)$. Next, we set $d = \mathrm{E}(x)$ and $\sigma^2 = \mathrm{var}(x)$, recalculate the prior variance of $y - rx$ from (10), and obtain posterior moments again. We may continue this process until convergence. In practice we will have not one but many indicator ratios. The iteration procedure is then applied to all of them simultaneously.

The iterative procedure is justified by the fact that it leads to the correct first two posterior moments. The process does not, however, always converge, and it turns out to be rather unstable. Therefore we outline an alternative approach, based on a different philosophy, below.

6.2. Invariance

Suppose again that the ratio y/x has a prior mean or median r, so that

$$\Pr\left(\frac{r}{a} < \frac{y}{x} < br\right) = 1 - \alpha,$$

where a and b are determined by the assumed prior standard deviation τ_1 of the ratio y/x. The associated confidence region is given by

$$C_1 = k\left(br - \frac{r}{a}\right) = kr\,(b - 1/a)$$

for some constant k. Let us assume that y and x are positive (almost surely) and that $a > 1$, $b > 1$, and $r > 0$. This gives

$$\Pr\left(-(1 - 1/a)rx < y - rx < (b - 1)rx\right) = 1 - \alpha$$

with associated confidence region

$$C_2 = k\left((b - 1)rx + (1 - 1/a)rx\right) = krx\,(b - 1/a) = C_1 x.$$

We wish to 'translate' the prior information on the ratio into prior information on its linearization. Motivated by the comparison of the two confidence bands, we assume that $y - rx$ is distributed around zero with some standard deviation $\omega_1 = \tau_1 x$.

Although we have written the prior information in terms of the ratio y/x, we could equally well have written it in terms of x/y. Two different linearizations do not, in general, yield the same result. The invariance approach demands that we obtain the same answer irrespective of whether we start from y/x or x/y.

Thus motivated, we rewrite the ratio as

$$\Pr\left(\frac{1}{br} < \frac{x}{y} < \frac{a}{r}\right) = 1 - \alpha$$

with confidence region

$$C_3 = k\left(\frac{a}{r} - \frac{1}{br}\right) = \frac{k}{r}\,(a - 1/b)\,.$$

We obtain the linearization

$$\Pr\left(-(1 - 1/b)\frac{y}{r} < x - (1/r)y < (a - 1)\frac{y}{r}\right) = 1 - \alpha,$$

with associated confidence interval

$$C_4 = k\left((a - 1)\frac{y}{r} + (1 - 1/b)\frac{y}{r}\right) = \frac{ky}{r}\,(a - 1/b) = C_3 y.$$

Let τ_2 be the prior standard deviation of x/y. Using the same argument as above, we assume that the linearization $x - (1/r)y$ is distributed around zero with some standard deviation $\omega_2 = \tau_2 y$.

Since $\omega_1 = r\omega_2$, we obtain $\tau_1 x = r\tau_2 y$. If the prior values for y and x satisfy $y/x = r$ exactly, then this gives

$$\frac{\tau_1}{r} = \frac{\tau_2}{1/r},$$

in other words: we must choose the *relative* precisions of the prior ratios to be the same.

However, the prior values for y and x will almost certainly not satisfy $y/x = r$ exactly. Hence, we must massage the priors y and x in such a way that $y/x = r$.

Let y_0 and x_0 be the given prior values, and assume that at least one of them is available. Now define

$$\bar{x} := \begin{cases} \dfrac{1}{1+r^2}x_0 + \dfrac{r^2}{1+r^2}(y_0/r) & \text{if both } x_0 \text{ and } y_0 \text{ are available;} \\ x_0 & \text{if only } x_0 \text{ is available;} \\ y_0/r & \text{if only } y_0 \text{ is available.} \end{cases}$$

Instead of x_0 and y_0, we use $\bar{x}$ and $\bar{y} := r\bar{x}$. Then, letting $\omega_1 := \tau_1\bar{x}$ and $\omega_2 := \tau_2\bar{y}$, the procedure will be invariant to whether we choose y/x or x/y as our starting point. The weights $1/(1+r^2)$ and $r^2/(1+r^2)$ are 'optimal' in the sense that if x_0 and y_0 are unbiased estimates of x and y, if $\mathrm{E}(y_0 - rx_0) = 0$, and if x_0 and y_0 are uncorrelated with equal variances, then $\bar{x}$ is also an unbiased estimate of x and its variance is minimized.

One can look at the problem in a different way. If we think of the problem as minimizing a weighted sum of squares, then the relevant term in the objective function is

$$\phi_1 := \frac{(y/x - r)^2}{\tau_1^2} = \left(\frac{y - rx}{\tau_1 x}\right)^2$$

or

$$\phi_2 := \frac{(x/y - 1/r)^2}{\tau_2^2} = \left(\frac{y - rx}{\tau_2 r y}\right)^2.$$

Instead of ϕ_1 or ϕ_2, we use the term

$$\phi := \left(\frac{y - rx}{\tau_1\bar{x}}\right)^2 = \left(\frac{y - rx}{\tau_2 r\bar{y}}\right)^2,$$

where, as above, $\tau_1 = r^2\tau_2$, and $\bar{y}$ and $\bar{x}$ are chosen from the 'priors' such that $\bar{y} = r\bar{x}$. This provides further justification of the above method.

Summarizing, we start from a prior $y/x \sim (r, \tau^2)$ and reference data x_0 and y_0 (only one of them needs to be available). The invariant linearization is $y - rx \sim (0, \omega^2)$, where $\omega = \tau\bar{x}$. Notice that no iterations are required in this approach. This procedure works well in practice, because it forces the linearization to be accurate in precisely the region where it matters most.

At the end of the estimation process we obtain estimates μ_x for x and μ_y for y, and also estimates for $\mathrm{var}(\mu_x)$, $\mathrm{var}(\mu_y)$, and $\mathrm{cov}(\mu_x, \mu_y)$. The posterior moments of the ratio $R = y/x$ are then estimated by

$$\mathrm{E}(R) = \frac{\mu_y}{\mu_x}, \qquad \mathrm{var}(R) = \frac{\mathrm{var}(\mu_y - r\mu_x)}{\bar{x}^2} \leq \frac{\omega^2}{\bar{x}^2} = \tau^2,$$

where we notice that the posterior variance of R is never larger than the prior variance, in accordance to standard Bayesian theory.

7. Flow chart of the algorithm

For a better understanding we provide a brief chart of our algorithm. The algorithm is based on the availability of three sets of information on the $n \times 1$ vector x:

$$d_1|x \sim \mathrm{N}_p(D_1x, \Sigma_1), \qquad A_1x \sim \mathrm{N}_{m_1}(h_1, H_1), \qquad A_2x = h_2 \quad \text{(a.s.)},$$

namely the p data, the m_1 random priors, and the m_2 exact restrictions, respectively. The flow chart in Fig. 2 is based on Section 5, but extended because of the fact that some of the random priors will be ratios and therefore need to be linearized in order to fit in the above framework. We start with preliminary checks on the positive definiteness of two variance matrices Σ_1 and H_1 (in large applications these matrices are typically diagonal), two rank conditions, and the availability of references priors for the ratios. If these conditions are not all satisfied, consultation with the user is necessary. Once the conditions are satisfied we proceed as follows:

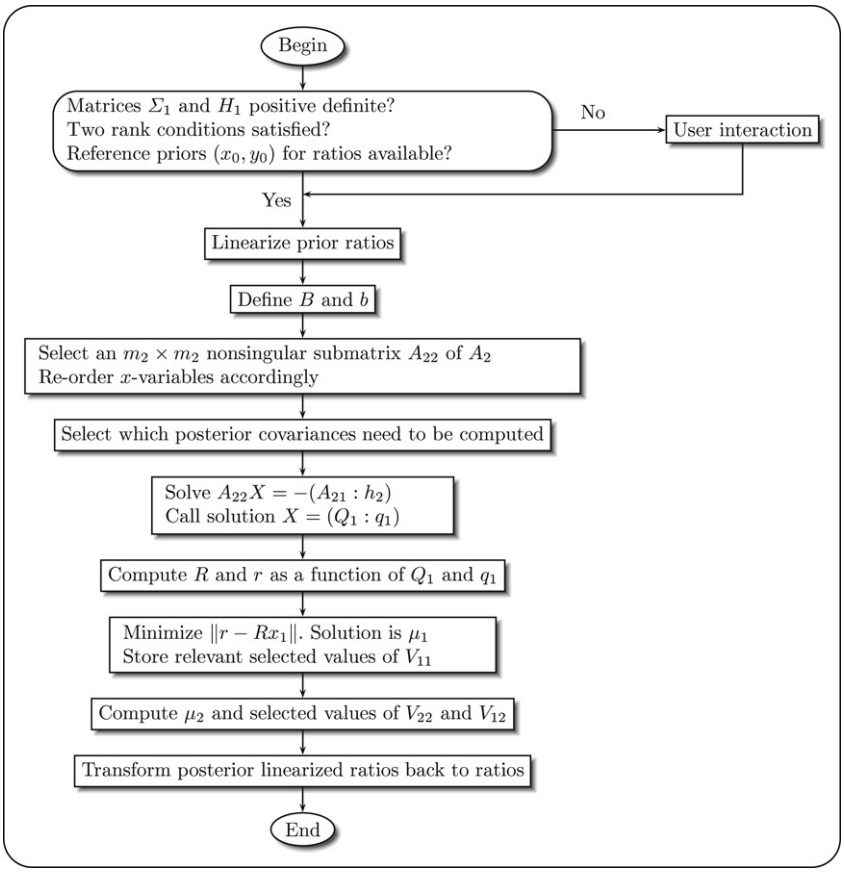

Fig. 2. Flow chart of procedures.

Step 1: *Linearization of ratios.* For each ratio $R := y/x$ we have prior information $R \sim (r, \tau^2)$. For either x or y (preferably both) we need a prior reference point, x_0 and y_0. Define

$$\bar{x} := \begin{cases} \dfrac{1}{1+r^2}x_0 + \dfrac{r^2}{1+r^2}(y_0/r) & \text{if both } x_0 \text{ and } y_0 \text{ are available;} \\ x_0 & \text{if only } x_0 \text{ is available;} \\ y_0/r & \text{if only } y_0 \text{ is available.} \end{cases}$$

Then, we linearize the ratio R as $y - rx \sim \mathrm{N}(0, \omega^2)$, where $\omega^2 := \tau^2\bar{x}^2$. The (new) matrix A_1 then includes both the old linear priors and the new linearized ratios. Now everything is linear.

Steps 2–8: *Implementation.* This is as described in Section 5, except that we may select posterior covariances to be calculated, in excess to those that the program requires and selects itself.

Step 9: *Back-transformation to ratios.* For each ratio $R = y/x$, we now have estimates μ_x and μ_y, and also estimates for $\mathrm{var}(\mu_x)$, $\mathrm{var}(\mu_y)$, and $\mathrm{cov}(\mu_x, \mu_y)$. The posterior moments of the ratio $R = y/x$ are then estimated by

$$\mathrm{E}(R) = \frac{\mu_y}{\mu_x}, \qquad \mathrm{var}(R) = \frac{\mathrm{var}(\mu_y) + r^2\mathrm{var}(\mu_x) - 2r\mathrm{cov}(\mu_x, \mu_y)}{\bar{x}^2}.$$

8. Simulation set-up for the Snaer-OLS routine

In the following two sections we describe some Monte Carlo experiments concerning the Snaer-OLS routine, which we consider the most critical of the program. Our framework is the standard linear regression model

$$y = X\beta + \varepsilon, \quad \varepsilon \sim \mathrm{N}(0, \sigma^2 I_n),$$

from which we obtain the OLS estimates and their variance matrix as

$$\hat{\beta} = (X'X)^{-1}X'y, \quad \text{var}(\hat{\beta}) = \sigma^2(X'X)^{-1}.$$

We shall be concerned with the situation when the $n \times k$ design matrix X is of 'large' dimension and is 'sparse'. (Notice that in Sections 7 and 8 we write the dimensions of the $(p + m_1) \times (n - m_2)$ design matrix as $n \times k$, which is more common and provides and easier link to least squares.) The two questions are whether our method, based on the Gould and Nocedal (1998) algorithm, performs the task which it is supposed to perform, and also whether it compares favorably to other available methods.

We wish to test our method in terms of precision and speed as a function of the dimensions (n and k), the sparsity, and the ill-conditioning of the problem. To conduct the simulations within the framework of the linear regression model, we need values for the parameters $\beta_1, \ldots, \beta_k$, and σ^2, and for the $n \times k$ design matrix X. Throughout the simulation experiment we set $\sigma^2 = 1$ and $\beta = 0$. Given β and σ^2, we generate $\varepsilon_1, \ldots, \varepsilon_n$ as independent draws from a N(0, 1) distribution, and obtain $y = X\beta + \varepsilon = \varepsilon$. The only difficulty in the set-up is the choice of X, because we wish to assess the precision of the method, and hence require an X-matrix such that $(X'X)^{-1}$ is known analytically, and where the sparsity and the ill-conditioning can be controlled.

The *sparsity* s of a matrix X is defined as

$$s(X) := 1 - \frac{\text{number of structural zeros in } X}{\text{number of elements in } X},$$

which is a number between 0 (null matrix, complete sparsity) and 1 (full matrix, no sparsity). For example, the sparsity of a diagonal $k \times k$ matrix is $1/k$. The sparsity s of our problem is defined as the sparsity of X. The ill-conditioning of the problem is defined by the *condition number* c:

$$c := \sqrt{\frac{\text{largest eigenvalue of } X'X}{\text{smallest eigenvalue of } X'X}}.$$

Since X has full column-rank k, we can write

$$X = \left(X(X'X)^{-1/2}\right)(X'X)^{1/2} \equiv ST,$$

where S is a semi-orthogonal $n \times k$ matrix such that $S'S = I_k$ and T is a symmetric nonsingular $k \times k$ matrix. We now construct S and T in such a way that T (and hence $X'X$) has an analytically known inverse.

8.1. Construction of T

Let T_0 be the tridiagonal symmetric $k \times k$ matrix

$$T_0 = \frac{1}{4}\begin{pmatrix} 2 & -1 & 0 & \ldots & 0 & 0 \\ -1 & 2 & -1 & \ldots & 0 & 0 \\ 0 & -1 & 2 & \ldots & 0 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 0 & 0 & 0 & \ldots & 2 & -1 \\ 0 & 0 & 0 & \ldots & -1 & 2 \end{pmatrix}.$$

From Grenander and Szegö (1985 p. 67) and Samarsky (1971, p. 53), we know that the eigenvalues $\lambda_{01} > \lambda_{02} > \cdots > \lambda_{0k}$ of T_0 are given by

$$\lambda_{0j} = \sin^2\left(\frac{(k+1-j)\pi}{2(k+1)}\right), \quad j = 1, \ldots, k,$$

and the corresponding eigenvectors $u_1, u_2, \ldots, u_k$ by

$$u_{ij} = \sqrt{\frac{2}{k+1}}\sin\left(\frac{i(k+1-j)\pi}{k+1}\right),$$

where u_{ij} denotes the ith component of u_j. Let Λ_0 denote the diagonal matrix containing $\lambda_{01}, \lambda_{02}, \ldots, \lambda_{0k}$, and let U denote the orthogonal matrix $U = (u_1, u_2, \ldots, u_k)$. Then,

$$T_0 = U\Lambda_0 U', \qquad T_0^{-1} = U\Lambda_0^{-1}U'.$$

Now consider two positive numbers $0 < c_{\min} < c_{\max} < 1$, and define the $k \times k$ matrix T as

$$T := \alpha T_0 + \beta I_k,$$

where

$$\alpha := \frac{c_{\max} - c_{\min}}{\lambda_{01} - \lambda_{0k}}, \qquad \beta := \frac{\lambda_{01}c_{\min} - \lambda_{0k}c_{\max}}{\lambda_{01} - \lambda_{0k}}.$$

The eigenvalues $\lambda_1 > \lambda_2 > \cdots > \lambda_k$ of T are then given by

$$\lambda_j = \alpha\lambda_{0j} + \beta, \quad j = 1, \ldots, k,$$

so that, by construction, $\lambda_1 = c_{\max}$ and $\lambda_k = c_{\min}$. Moreover, T possesses the same set of eigenvectors as T_0, so that

$$T = U\Lambda U', \qquad T^{-1} = U\Lambda^{-1}U',$$

where

$$\Lambda = \operatorname{diag}(\lambda_1, \lambda_2, \ldots, \lambda_k).$$

Note that T is a sparse matrix (tridiagonal), T^2 is also sparse (five-diagonal), but that T^{-1} is a full matrix for any dimension.

8.2. Construction of S

All dimensions are integer powers of 2:

$$k := 2^j, \quad n := 2^{i+j}, \quad n_k := n/k = 2^i.$$

Let d be an integer satisfying $0 \le d \le n_k$. We construct the $n \times k$ matrix S as

$$S := \frac{1}{\sqrt{n_k}} \begin{pmatrix} S_0 \\ \vdots \\ S_0 \\ I_k \\ \vdots \\ I_k \end{pmatrix} \begin{matrix} \left.\vphantom{\begin{matrix} S_0 \\ \vdots \\ S_0 \end{matrix}}\right\} & d \text{ times} \\ \left.\vphantom{\begin{matrix} I_k \\ \vdots \\ I_k \end{matrix}}\right\} & n_k - d \text{ times,} \end{matrix}$$

where the orthogonal $k \times k$ matrix S_0 is constructed as

$$S_0 := A_1 A_2 \cdots A_{k^*}, \quad 1 \le k^* \le k - 1,$$

with

$$A_j := \begin{pmatrix} I_{j-1} & 0 & 0 \\ 0 & Q(\theta_j) & 0 \\ 0 & 0 & I_{k-j-1} \end{pmatrix}, \quad j = 1, \ldots, k^*,$$

and

$$Q(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}, \qquad \theta_j = \frac{\pi}{2(k^* - j + 1)}.$$

The sparsity of S is a weighted average of the sparsity of S_0 and the sparsity of I_k. Since the number of nonzeros in S_0 is $k + k^*(k^* + 3)/2$, the sparsity of S is given by

$$s(S) = \frac{1}{k} + \frac{k^*(k^* + 3)d}{2nk},$$

which depends on the parameters d and k^*. In most simulations we choose $k^* = k - 1$. For $d = 0$, we have maximal sparsity $s(S) = 1/k$; for $d = n_k$, we have minimal sparsity $s(S) = s(S_0) = 1/k + k^*(k^* + 3)/(2k^2)$.

8.3. Analytical solution

Given S and T, and the definition $X = ST$, we find

$$X'X = T'S'ST = T^2, \qquad (X'X)^{-1} = T^{-2} = U\Lambda^{-2}U' = \sum_{j=1}^{k} \frac{1}{\lambda_j^2} u_j u_j',$$

and

$$\hat{\beta} = (X'X)^{-1}X'y = T^{-2}T'S'y = T^{-1}(S'y) = \sum_{j=1}^{k} \frac{1}{\lambda_j} u_j u_j' z,$$

where $z := S'y$. The estimator for σ^2 can be found as

$$\hat{\sigma}^2 = \frac{1}{n-k} y'(I_n - X(X'X)^{-1}X')y = \frac{1}{n-k}(y'y - z'z)$$

and the condition number is

$$c := \sqrt{\frac{\lambda_1(X'X)}{\lambda_k(X'X)}} = \sqrt{\frac{\lambda_1(T^2)}{\lambda_k(T^2)}} = \frac{\lambda_1(T)}{\lambda_k(T)} = \frac{c_{\max}}{c_{\min}}.$$

Throughout the simulation experiment we set $c_{\max} = 0.9$. The simulations are then controlled by five parameters: i, j, c, d, and k^*, thus providing the dimensions $k = 2^j$ and $n = 2^{i+j}$, the bounds $c_{\max} = 0.9$ and $c_{\min} = c_{\max}/c$, and the sparsity s (as a function of d and k^*). The condition number c can be fully controlled, the sparsity s can be approximately controlled.

9. Simulation results

9.1. Methods considered

We are concerned with accuracy and speed. We consider six sparse methods. The first four methods use various sparse Matlab functions. The next method is also a sparse method, but not used in Matlab. The final method is our Snaer-OLS routine.

Method 1: *Exact numerical inversion (Matlab).* The standard way to calculate the OLS solution is to apply the explicit formulae $\hat{\beta} = (X'X)^{-1}X'y$ and $\widehat{\text{var}}(\hat{\beta}) = \hat{\sigma}^2(X'X)^{-1}$ with $\hat{\sigma}^2 = (y - X\hat{\beta})'(y - X\hat{\beta})/(n-k)$, where the matrix $(X'X)^{-1}$ is obtained by applying the Matlab matrix inversion function.

Method 2: *QR decomposition (Matlab).* Matlab provides another way to calculate the OLS solution for linear regression through the 'economy-size' QR decomposition available as part of the *regress* algorithm. We write $X = QR$, where Q is semi-orthogonal ($Q'Q = I_k$) and R is an upper triangular $k \times k$ matrix with positive diagonal elements; see Chatterjee and Hadi (1986), and Draper and Smith (1998, p. 94). The formulae now depend on the inversion of the triangular matrix R and are given by

$$\hat{\beta} = R^{-1}Q'y, \quad \widehat{\text{var}}(\hat{\beta}) = \hat{\sigma}^2 R^{-1}(R^{-1})', \quad \hat{\sigma}^2 = \frac{y'(I_n - QQ')y}{n-k}.$$

Method 3: *Least-squares minimization (Matlab).* The next method considers the numerical minimization problem

$$\min_{\beta} \frac{1}{2}\|y - X\beta\|^2$$

and uses the Matlab function *mldivide* (\) for the optimization. More specifically, the Matlab command $X \setminus y$ uses Householder reflections to compute an orthogonal-triangular factorization $XP = QR$, where P is the permutation matrix, Q is semi-orthogonal, and R is upper triangular. Then,

$$\begin{aligned}\hat{\beta} &= X \setminus y = (X'X)^{-1}X'y = (PR'Q'QRP')^{-1}PR'Q'y \\ &= P(R'R)^{-1}R'Q'y = P(R \setminus (Q'y)).\end{aligned}$$

This method is thus closely related to the QR decomposition.

In order to compute the variance matrix, we require elements of the inverse of the matrix $X'X$. Using the same Matlab algorithm *mldivide*, the jth column $v^{(j)}$ of $V := (X'X)^{-1}$ can be found by considering the matrix equation $(X'X)V = I_k$ and solving the minimization problem

$$\min_{v^{(j)}} \frac{1}{2}\|e^{(j)} - X'Xv^{(j)}\|^2, \quad j = 1, \ldots, k,$$

where $e^{(j)}$ denotes the jth unit vector, that is the jth column of I_k. For the calculation of $\hat{\beta}$, methods 2 and 3 give the same numerical results, but for the calculation of the elements of the variance matrix, the method of computation and hence the numerical results are different.

Method 4: *Conjugate gradient method (Matlab).* A large number of iterative methods for solving linear equations can be derived as minimization methods. If A is a positive definite $k \times k$ matrix, then the function

$$\varphi(x) = \frac{1}{2}x'Ax - b'x$$

has a unique minimizer which is the solution of $Ax = b$. Therefore minimization of φ is the same as solving the equation $Ax = b$. Many minimization methods for φ can be written in the form

$$x_{k+1} = x_k + \alpha_k p_k, \quad k = 0, 1, \ldots.$$

In particular, the classical *conjugate gradient method* is defined by the iterative sequence

$$\begin{aligned}&\alpha_k = \frac{r_k'r_k}{p_k'Ap_k}, \qquad \beta_k = \frac{r_{k+1}'r_{k+1}}{r_k'r_k}, \\ &r_{k+1} = r_k - \alpha_k Ap_k, \qquad p_{k+1} = r_{k+1} + \beta_k p_k,\end{aligned}$$

with some starting point x_0 and $p_0 = r_0 = b - Ax_0$; see Stoer and Bulirsch (1993 p. 607) and Golub and Ortega (1991, Section 9.3). The name of the method arises from the fact that the direction vectors satisfy $p_i'Ap_j = 0$ for any $i < j$ — such vectors are called 'conjugate'. It can be shown that x_k converges to the exact solution in at most k steps. In practice, however, this does not always happen due to rounding errors.

The Matlab procedure *pcg* has been used to find the solution of the system of normal equations

$$X'X\beta = X'y.$$

This is allowed because $X'X$ is symmetric. It is well-known that the performance of *pcg* is affected by the choice of preconditioner. For the applications that we have in mind the system matrices are of a rather arbitrary structure which prevents an efficient use of preconditioners, because we would need to develop a new preconditioner for each new data set.

Another Matlab function, also based on solving the normal equations, is the generalized minimum residual method (*gmres*). Its performance will be briefly discussed when we present the results for the *pcg* method.

The variance matrix is obtained by considering the equations

$$X'Xv^{(j)} = e^{(j)}, \quad j = 1, \ldots, k,$$

and applying the *pcg* routine.

Method 5: *The iterative Lanczos algorithm.* The Numerical Algorithms Group (NAG Fortran Library, 2006) provides the subroutine *f04qaf* for solving large sparse OLS problems. This is an iterative method, based on the bidiagonalization procedure of Golub and Kahan (1965) and the Lanczos (1950) iteration. In essence it is the

lsqr method of Paige and Saunders (1982a,b) belonging to the family of Krylov methods; see Broyden and Vespucci (2004).

The Lanczos scheme requires a symmetric matrix, and hence we write our system as

$$\begin{pmatrix} I & X \\ X' & 0 \end{pmatrix} \begin{pmatrix} r \\ \beta \end{pmatrix} = \begin{pmatrix} y \\ 0 \end{pmatrix},$$

where r denotes the residual vector. Similarly, the variance matrix is obtained by considering the symmetric system

$$X'Xv^{(j)} = e^{(j)}, \quad j = 1, \ldots, k.$$

The time taken by the routine is likely to be principally determined by the time taken in APROD, which is called twice on each iteration. APROD performs a matrix-vector product, where the sparsity is explicitly taken into account. Although the Lanczos process will usually converge more quickly if some form of preconditioning is employed (Broyden and Vespucci, 2004, Chapter 11), we do not use preconditioners because we are investigating the performance of methods for general sparsity patterns, while preconditioners are structure-dependent.

Method 6: *Snaer-OLS.* This is our method, based on the Gould–Nocedal algorithm, and described before.

All computations were performed on a desktop personal computer, Intel Pentium 4 processor, 2.66 GHz speed, 2 GB RAM, and endowed with Microsoft Windows XP Professional (SP2). To facilitate comparison all methods were compiled as stand-alone Windows executables. The Matlab Compiler version 4.6 for Matlab version 7.4 (R2007a) has been used for methods 1–4. The Fortran programs were compiled using the Compaq Visual Fortran 6.6 compiler.

9.2. Accuracy

In the first two experiments, we set

$$n = 2k, \quad k = 2^j, \quad c = 100{,}000,$$

and we let $j = 5, \ldots, 12$. The total number of elements in X thus equals $nk = 2^{2j+1}$, which ranges from 2^{11} (about two thousand) to 2^{25} (over 33 million). Since $n_k = n/k = 2$, we have $0 \leq d \leq 2$. We consider the two extremes $d = 0$ and $d = 2$, both for $k^* = k - 1$.

The accuracy of the various solutions is controlled by the average relative deviation

$$\text{ARD} = \sqrt{\frac{1}{k}\sum_{j=1}^{k}(\hat{v}_{jj}/v_{jj} - 1)^2},$$

where $\hat{v}_{jj}$ denotes the jth diagonal element of $\hat{V} := \hat{\sigma}^2(X'X)^{-1}$ in one of the six algorithms, and v_{jj} denotes its true value. The ARD measure thus takes into account both the accuracy of $\hat{\beta}$ (through $\hat{\sigma}^2$) and the accuracy of $(X'X)^{-1}$. We have chosen the diagonal elements of the variance matrix not only for simplicity, but also because of their obvious practical importance. In the national accounts applications we always calculate the diagonal elements of $(X'X)^{-1}$, but only occasionally some of the offdiagonal elements. Controlling the accuracy of the variances is therefore our primary concern. The accuracies are reported in Table 1 for very sparse systems ($d = 0$) in the top panel and less sparse systems ($d = 2$) in the bottom panel. In the top panel the sparsity increases from 9.2% to 0.1% as the dimension increases. Method 2 (QR) is the most accurate, while the Lanczos algorithm (method 5) is very inaccurate and has not been calculated for $j > 8$. (The related Matlab *lsqr* function behaves similarly and does not reach acceptable accuracy for any reasonable choice of tolerances and number of iterations.) An analysis of the breakdown causes of the Lanczos method can be found in Broyden and Vespucci (2004, pp. 24–26). The other methods (1, 3, 4, and 6) are all of the same degree of accuracy, indicating that on average $\hat{v}_{jj}/v_{jj} \approx 1.0000001$, which seems sufficiently accurate. If the system becomes large, all methods except Snaer-OLS break down due to memory allocation restrictions. The accuracy of the *gmres* function, related to *pcg* (method 4), is comparable with *pcg*.

In the bottom panel ($d = 2$) there is much less sparsity, ranging only from 57.5% for $j = 5$ to 5.9% for $j = 11$. The conclusions are the same as for the sparse system. Clearly, a sparse procedure will be very inefficient (in terms of computing time) when employed in a nonsparse situation, but the accuracy seems unaffected.

Table 1
Relative accuracy ARD ($\times 10^{-7}$), $n = 2k$, $k = 2^j$, $c = 100{,}000$, and $d = 0$ (top panel) and $d = 2$ (bottom panel)

j	Matlab				NAG	Snaer-OLS	Sparsity
	1	2	3	4	5	6	
5	2.24	0.000015	2.24	2.38	4.58	2.32	0.0918
6	2.49	0.000004	2.49	2.51	22.4	2.51	0.0464
7	1.57	0.000070	1.57	1.56	174	1.76	0.0233
8	1.42	0.000006	1.42	1.38	9,984,000	1.40	0.0117
9	4.05	0.000031	4.05	4.01	–	3.97	0.0059
10	2.42	0.000015	2.42	2.43	–	2.39	0.0029
11	0.81	0.000056	0.81	0.80	–	0.63	0.0015
12	–	–	–	–	–	1.45	0.0007
5	0.56	0.000015	0.56	0.45	2.59	0.14	0.5752
6	0.28	0.000031	0.28	0.17	20.7	0.61	0.5383
7	0.07	0.000025	0.07	0.05	172	0.14	0.5193
8	0.03	0.000006	0.03	0.02	9,670,600	0.26	0.4581
9	0.06	0.000029	0.06	0.06	–	0.21	0.2565
10	0.08	0.000016	0.08	0.07	–	0.16	0.1256
11	0.07	0.000008	0.08	0.08	–	0.36	0.0588

The chosen accuracy measure ARD is quite robust. If instead we calculate the average squared deviation taking *all* elements of the variance matrix, this produces only a marginal increase. For example for $j = 8$ and $d = 0$ (line 4 from the top in Table 1), the alternative measure gives 1.44×10^{-7} instead of 1.42×10^{-7} for method 1 and has essentially no effect on method 3 and other methods.

9.3. Tolerances

In the first three Matlab routines (exact, QR, and *mldivide*) no tolerances can be set, so that the defaults must be used.

The *pcg* algorithm was used with tolerance 10^{-15} and the maximum number of iterations: 5000 for $j < 9$, 10,000 for $j = 9$, 20,000 for $j = 10$, and 120,000 for $j = 11$.

For NAG *f04qaf* the tolerances tol_1 and tol_2 were set to 10^{-14} and 10^{-8}, respectively, and the routine accepts $\hat{\beta}$ as a solution of $X'X\beta = X'y$ if it is estimated that one of the following two conditions is satisfied:

$$\|r\| \leq \text{tol}_1\|X\| \cdot \|\hat{\beta}\| + \text{tol}_2\|y\|, \qquad \|X'r\| \leq \text{tol}_1\|X\| \cdot \|r\|,$$

where $r := y - X\hat{\beta}$ is the residual vector. The parameter *conlim* is the upper limit on condition number of X, and was set to $conlim = 10^{12}$. The parameter *itnlim* is the upper limit on the number of iterations and was set to $itnlim = 1{,}000{,}000$. Further reduction in tolerances and increase in iteration and condition numbers did not lead to higher accuracy. The values of tol_1 and tol_2 are inspired by the recent literature on stopping criteria and error estimation in the conjugate gradient method; see Strakoš and Tichý (2002, 2005) and Arioli (2004).

For the Snaer-OLS functions the default tolerances have been used.

9.4. Computing time

The computing time (in seconds) that each method requires is measured as follows. For the Matlab procedures 1–4, we employ the *tic* and *toc* functions: *tic* saves the current time that *toc* uses later to measure the elapsed time which Matlab has taken to complete the required operations. For the NAG-Lanczos (5) and Snaer-OLS (6) routines we use the *rtc* function. Naturally, computing time grows with the dimension. In Fig. 3 we present the computing time T as a function of the dimension k in the sparse case ($d = 0$). The figures are plotted on a log–log scale, where $j := \log k$ and $\tau := \log T$ are logarithms to base 2, so that $k = 2^j$ and $T = 2^\tau$. The relationship on this scale appears to be close to linear. This suggests that $T \approx \text{constant} \times ((1 - d_s)nk)^\delta$. We expect $\delta > 1$, because all elements of the matrix must be evaluated at least once. A value of δ close to one thus indicates a near-optimal slope; the lower the δ, the faster the method. The Gould–Nocedal method (Snaer-OLS) is the best in this sense with $\delta = 1.9$.

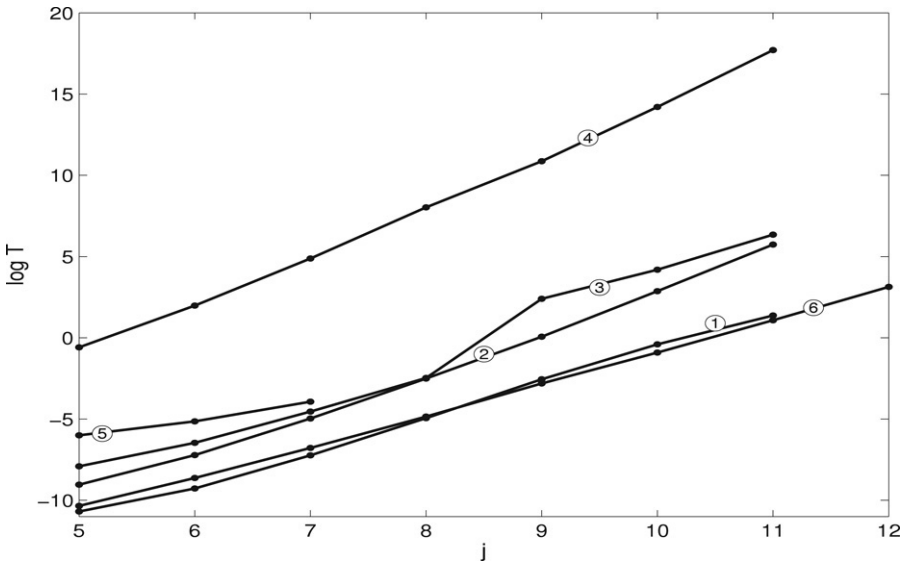

Fig. 3. Computing time, $k = 2^j$, $n = 2k$, $c = 100{,}000$, and $d = 0$.

For large systems, say when $k = 2^{12}$ and $n = 2^{13}$, exact inversion (method 1) and QR (method 2) break down due to memory allocation problems. The Lanczos algorithm (method 5) breaks down earlier, but for another reason: the accuracy becomes unacceptably low, as we have seen. For large sparse systems with $j = 12$ or larger (giving an X-matrix with 33.6 million or more entries), only Snaer-OLS works. For $j = 11$, all methods except the NAG routine work with reasonable and comparable accuracies except the QR method which is much more accurate (see Table 1). However, the computing times of the methods differ greatly. Method 4 takes 59 h to complete its calculations. (For the *gmres* function, related to *pcg* of method 4, the number of iterations required and thus the computation time is so large that the method becomes unusable for $j > 8$.) In contrast, method 3 takes 82 s, while Snaer-OLS's Gould–Nocedal method 6 only requires 2.1 s, and thus performs best for large sparse systems. Fig. 4 illustrates the performance of the methods in a less sparse situation, where $d = 2$. Computations have only been performed for $j \leq 11$. The exact (1) and QR (2) routines work best here, while the Lanczos method, again, was not able to reach sufficient accuracy. Snaer-OLS's Gould–Nocedal method is not the favorite choice now.

Let us consider the largest system, where $j = 11$, so that $k = 2^{11}$ and $n = 2^{12}$, and compare the sparse situation (Fig. 3, sparsity 0.2%) with the less sparse situation (Fig. 4, sparsity 5.9%). All methods take longer in the less sparse situation. In particular, Snaer-OLS's method requires 2 s in the sparse situation and 26 min in the less sparse situation. Apparently, a sparsity of 6% is not sufficient for a sparse procedure to work well. What degree of sparsity, then, is sufficient? Our experiments suggest that a matrix should be called 'sparse' when at least 98% or 99% of its elements are structural zeros. With sparsity thus defined, a sparse method will work well on a sparse matrix.

9.5. Dimension limitations

All methods are constrained by computer memory limitations. When a nonsparse method is used, Matlab is unable to store an $n \times k$ matrix when $nk > 2^{25}$. For example, a matrix with $k = 2^{13}$ (about 8000) and $n = 2k$ or with $k = 2^{12}$ (about 4000) and $n = 8k$ cannot be stored.

This upper bound also provides a limit to the number of elements in the coordinate storage representation of a sparse matrix. Since three numbers are required to store a sparse element (row-index, column-index, value), the maximum number of elements in a sparse matrix in Matlab cannot be more than approximately $2^{25}/3$. Snaer-OLS's Gould–Nocedal routine, however, works for dimensions even higher than this.

10. Conclusions and extensions

The Snaer program discussed in this paper calculates the posterior mean and variance of variables on some of which we have data (with precisions), on some we have prior information (with precisions), and on some prior indicator ratios

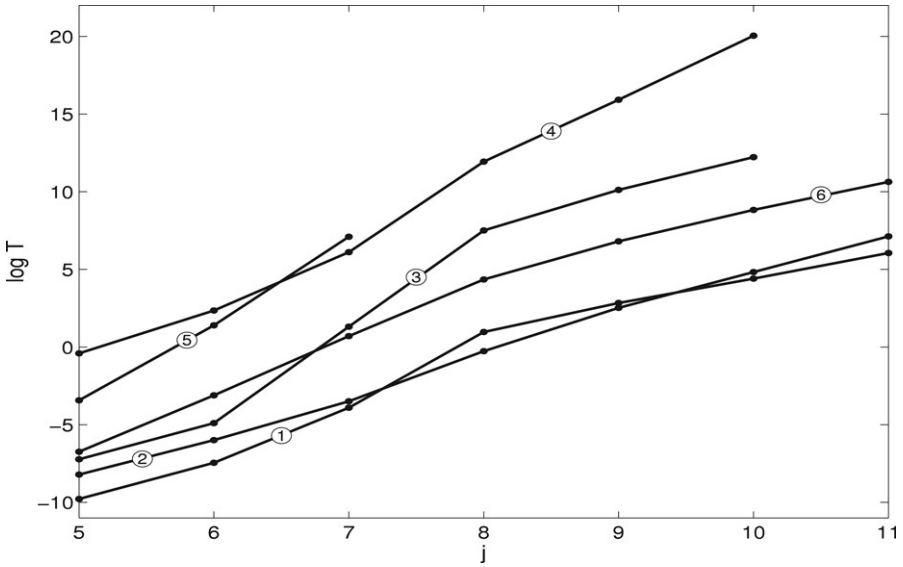

Fig. 4. Computing time, $k = 2^j$, $n = 2k$, $c = 100{,}000$, and $d = 2$.

(with precisions) are available. The variables must satisfy a number of exact restrictions. The system is both large and sparse.

Two aspects of the statistical and computational development are a practical procedure for solving a linear integer system, and a stable linearization routine for ratios.

We tested our numerical method for solving large sparse linear least-squares estimation problems, and found that it performs well, even when the $n \times k$ design matrix is large ($nk > 2^{23} \approx 0.8 \times 10^7$). In fact, Snaer-OLS can handle design matrices of dimension 16,000 × 12,000, where $nk \approx 2^{27.5} \approx 1.9 \times 10^8$.

Our procedures work well if the system is sparse, and in this context we call a matrix 'sparse' when at least 98% or 99% of its elements are structural zeros.

We now discuss three practical difficulties in applying our methods: how to obtain prior precisions, how to deal with skewness, and how to include dynamics in the system.

Prior precisions. A major advantage but also a major challenge is that we allow priors and prior precisions. The priors themselves (that is, the means of the priors) are based on expert opinion, historical data, neighboring information, or otherwise. Their precisions are more difficult to obtain.

If exact precisions are available, then we can use these. But in practice we will only have indications. For example, banking data are more precise than survey data. For practical applications we adopt a small number (say four) levels of precision, defined by the coefficient of variation (CV), that is, the ratio of the standard deviation to the mean: $\text{CV} = \sigma/\mu$. The coefficient of variation is a dimensionless number that allows comparison of the variation of populations that have significantly different mean values, and it is often used in the context of the normal distribution when the means are positive (as in our applications).

A problem arises when the mean is near zero, because the coefficient of variation then becomes sensitive to changes in the standard deviation; in that case we have to assign precisions on an *ad hoc* basis.

We also have to decide which values are appropriate for the four levels of precision. This depends on the situation and requires expert knowledge of the data.

Recall that the variance of μ is given by

$$\text{var}(\mu) = \sigma^2 \begin{pmatrix} I \\ Q_1 \end{pmatrix} (R'R)^{-1}(I, Q_1'),$$

where, up until now, we have set $\sigma^2 = 1$. In fact we may estimate σ^2 by the usual unbiased estimator as

$$\widehat{\sigma}^2 = \frac{\|r - R\mu_1\|}{p + m_1 + m_2 - n}.$$

Including σ^2 implies that the prior precisions only need to be assigned in *relative* terms rather than in absolute terms, thus greatly improving their credibility.

In practical applications one will usually define not only prior precisions but also correlations, which seems even more demanding. But there is no theoretical constraint to include correlations if they appear to be available.

Skewness. Our data and priors are assumed to be normal, hence symmetric. This assumption allows us to rewrite the problem in a manageable format so that we can estimate very large systems. But there is also a cost, because a number of data and priors are known to be skewed. Although we can find *ad hoc* solutions by introducing indicator ratios which provide a one-directional force, the general problem remains unresolved.

Dynamics. There is no problem in combining data from various years and defining the interactions. We simply think of x_t and x_{t-1} as two different variables, linked by some ratio or linear relationship. This allows for a simple form of dynamics in the system.

Finally we briefly discuss four topics that are currently under investigation as part of the Snaer program.

Sensitivity analysis. The program makes it easy to assess how small changes in one assumption will work through the system and affect one or more key variables and their precisions. For example, if more precise information were available on a set of variables (say by financing a survey), then we can calculate the impact of these improved precisions *before* the survey has in fact taken place. The improved precisions of a subset of the data may or may not matter for the key variables in our system — it is hard to decide in such a large system with so many constraints. But in our program such assessments are relatively easy to perform.

Aggregation and hierarchy. On the whole there is a tendency to believe that the more data the better. Certainly this is the view in national accountants circles. But is this true? How useful is it to work with large (micro-) data sets when we are primarily interested in macro-estimates? How much do the (often not very precise) micro-data help in improving the macro-estimates? Of course, it depends. But precisely on what does it depend in practical nonlinear situations? This we do not know yet.

Forecasting. The set-up and focus of this paper are estimation, not forecasting. We want to estimate last year's national accounts. In essence, however, our procedure is a forecasting (more accurately, prediction) procedure. Given all available information, we provide the best possible forecast at a particular point in time. If more information becomes available a new estimate (forecast) can be produced quickly. The same set-up and software can then also be used to produce *next year's* national accounts, of course primarily based on priors because data will not yet be available. This is unusual, but it will be useful in policy debates based on the national accounts.

Interaction between data collection and estimation. Data collection and data analysis are typically performed by different groups of experts. Data collection is the expertise of national accountants working in national statistics offices or in international organizations like the United Nations. Data analysis and economic modeling are the expertise of academics at universities and other research environments. How do the data collection process and the subsequent econometric analysis interact? The econometrician usually takes the data as given, but in reality the data are also based on models, and these could be closely correlated with the research question of the econometrician. This can have serious effects on the reliability and interpretation of the analysis.

Acknowledgements

Earlier versions of this paper were presented at the 3rd IASC World Conference on Computational Statistics & Data Analysis, Cyprus, October, 2005, at Tilburg University, and at the Rutherford Appleton Laboratory, UK. We thank Mario Arioli, Iain Duff, and two referees for their thoughtful and constructive comments, and Jan van Tongeren for his long-term support in this project.

References

Abowd, J.M., Creecy, R.H., Kramarz, F., 2002. Computing person and firm effects using linked longitudinal employer–employee data. Cornell University Working Paper, Ithaca, NY.

Abowd, J.M., Kramarz, F., Margolis, D.N., 1999. High wage workers and high wage firms. Econometrica 67, 251–333.

Arioli, M., 2004. A stopping criterion for the conjugate gradient algorithm in a finite element method framework. Numerische Mathematik 97, 1–24.

Björck, Å, 1996. Numerical Methods for Least Squares Problems. SIAM, Philadelphia, PA.

Broyden, C.G., Vespucci, M.T., 2004. Krylov Solvers for Linear Algebraic Systems. Elsevier, Amsterdam.

Chatterjee, S., Hadi, A.S., 1986. Influential observations, high leverage points, and outliers in linear regression. Statistical Science 1, 379–416.

Danilov, D., Magnus, J.R., 2007. Some equivalences in linear estimation. Quantile 3, 83–90 (in Russian). English original can be downloaded from: http://cdata4.uvt.nl/websitefiles/magnus/paper77b.pdf.

Dongarra, J.J., Duff, I.S., Sorenson, D.C., van der Vorst, H.A., 1991. Solving Linear Systems on Vector and Shared Memory Computers. SIAM, Philadelphia.

Draper, N.R., Smith, H., 1998. Applied Regression Analysis, 3rd edn. John Wiley, New York.

Golub, G.H., Kahan, W., 1965. Calculating the singular values and pseudo-inverse of a matrix. SIAM Journal on Numerical Analysis 2, 205–224.

Golub, G.H., Ortega, J.M., 1991. Scientific Computing and Differential Equations: An Introduction to Numerical Methods. Academic Press, San Diego.

Gould, N.I.M., Nocedal, J., 1998. The modified absolute-value factorization for trust-region minimization. In: De Leone, R., Murli, A., Pardalos, P.M., Toraldo, G. (Eds.), High Performance Algorithms and Software in Nonlinear Optimization. Kluwer Academic Publishers, Dordrecht, pp. 225–241.

Grenander, U., Szegö, G., 1958. Toeplitz Forms and their Applications. University of California Press, Berkeley, Los Angeles.

Harwell Subroutine Library (HSL), 2004. AspenTech Ltd, Reading, UK.

Lanczos, C., 1950. An iteration method for the solution of the eigenvalue problem of linear differential and integral operators. Journal of Research of the National Bureau of Standards 45, 255–282.

Magnus, J.R., van Tongeren, J.W., 2008. Towards a new system of national accounts estimation, with an application to Angola (in preparation).

Magnus, J.R., van Tongeren, J.W., de Vos, A.F., 2000. National accounts estimation using indicator ratios. Review of Income and Wealth 46, 329–350.

Matlab 6.5, Release 13, 2002. The MathWorks Inc, Natick, MA, USA.

NAG Fortran Library, Mark 21, 2006. The Numerical Algorithms Group Ltd, Oxford, UK.

Paige, C.C., Saunders, M.A., 1982a. LSQR: An algorithm for sparse linear equations and sparse least squares. ACM Transactions on Mathematical Software 8, 43–71.

Paige, C.C., Saunders, M.A., 1982b. Algorithm 583 — LSQR: Sparse linear equations and least-squares problems. ACM Transactions on Mathematical Software 8, 195–209.

Pauletto, G., 1997. Computational Solution of Large-Scale Macroeconometric Models. Kluwer Academic Publishers, Dordrecht.

Samarsky, A.A., 1971. Introduction to the Theory of Difference Schemes. Nauka, Moscow (in Russian).

Stoer, J., Bulirsch, R., 1993. Introduction to Numerical Analysis, 2nd edn. Springer, New York.

Strakoš, Z., Tichý, P., 2002. On error estimation in the conjugate gradient method and why it works in finite precision computations. Electronic Transactions on Numerical Analysis (ETNA) 13, 56–80.

Strakoš, Z., Tichý, P., 2005. Error estimation in preconditioned conjugate gradients. BIT Numerical Mathematics 45, 789–817.