CPB Memorandum

CPB Netherlands Bureau for Economic Policy Analysis

Unit(s)	:	OMD
Author(s)	:	B.H. Hasselman
Number	:	2006/OMD/1
Date	:	August 8, 2006

A Newton algorithm for solving an underdetermined system of equations

When making short term macro-economic forecasts, there is often no clear transition from a period with a known and given history to a period with only forecasts. Often some information for a few economic variables such as unemployment, inflation and interest rates are already available for the first few quarters of a forecasting period.
When a macro-economic model is used as a tool for forecasting, it would be very useful if any available partial information on some economic variables could be used to make a conditional forecast.
This paper describes the method to achieve this that the CPB has implemented in its proprietary software package for econometric modelling. It also provides technical background information for the computer implementation.

I wish to thank Arie ten Cate, Debby Lanser, Hans Lunsing and Bert Smid for their comments and suggestions.

Contents

1 Introduction

When an econometric model is used for forecasting, it may be beneficial or required to use preliminary observations in the forecasting period on certain economic quantities such as unemployment or inflation. These observations provide partial additional information for the forecast and one would like to make a forecast with the model conditional on these observations. Sandee, Don and Van den Berg (1984) formulate the problem as follows

> *Economic time-series can be predicted by the solution, period after period, of the equations of a model that uses exogenous projections and past endogenous variables as inputs.*
>
> *For the more distant past, statistical information about all variables will be available. If these data are substituted in the model, the behavioural equations show non-zero residuals.*
>
> *In the more recent periods, still in the past, a limited set of variables has already been observed. To produce an up-to-date forecast it is important to incorporate this recent statistical information.*

The most natural way to do this is to find values for error terms in the model such that the forecasted values for the observed quantities are equal to the observations.

Sometimes it may be necessary for an econometric model to yield predetermined outcomes for certain endogenous variables. Examples are

- making an annual model reproduce forecasts made with a quarterly model.
- making sectoral forecasts conditional on macroeconomic forecasts, which serve as the recent observations.
- forcing a model to generate a “desired outcome”, for example a required medium term growth of GDP.
- one may wish to investigate the behaviour of a model for a zero government budget deficit.

In all these cases the desired result may be achieved by manipulating error terms to force the model to yield the required outcome.

Usually a model has many more relevant error terms than observations. Therefore generally speaking there are infinitely many solutions for the error terms. From a statistical point of view it seems natural to require that the vector of error terms has minimum Euclidean norm. The problem can be formulated mathematically as follows

$$\begin{aligned} &\min_{u} \quad u^T u \\ &\text{subject to} \quad w = h(u) \end{aligned}$$

with $u \in \mathbb{R}^n$, $w \in \mathbb{R}^m$, $h : \mathbb{R}^n \rightarrow \mathbb{R}^m$ and $m \leq n$. The vector w represents the targets for some endogenous variables of the underlying system of equations. The vector u represents the error terms to be used for achieving the targets. The vector valued function h represents the – usually implicit – reduced form equations relating the targeted endogenous variables to the residuals. The error terms in this formulation of the problem are equal to the model error terms scaled with a positive factor such as a standard deviation or a root-mean-squared-error.

In the following sections first the mathematics of the problem are discussed. Then the computation of the jacobian matrix is discussed and the algorithm is presented in the form of pseudo code; it is based on Sandee (1994). Finally, some cautionary remarks are made and possible extensions are discussed.

2 Mathematics

As stated in the introduction the task is to solve the following minimisation program

$$\min_{u} \quad u^T u \tag{2.1a}$$

$$\text{subject to} \quad w = h(u) \tag{2.1b}$$

with $u \in \mathbb{R}^n$, $w \in \mathbb{R}^m$, $h : \mathbb{R}^n \to \mathbb{R}^m$ and $m \leq n$. When $m < n$ the system of equations $w = h(u)$ is underdetermined and the minimisation is equivalent to finding the minimum norm solution of

$$w = h(u) \tag{2.2}$$

We assume that a solution exists.

Lagrange method

Using the Lagrange method the problem can be reformulated as

$$\min_{u,\lambda} \tfrac{1}{2} u^T u + \lambda \big(w - h(u) \big) \tag{2.3}$$

with $\lambda \in \mathbb{R}^m$. The first order necessary conditions are

$$u = D^T \lambda \tag{2.4a}$$

$$w = h(u) \tag{2.4b}$$

where $D \in \mathbb{R}^{m \times n}$. The matrix D is the jacobian of h with $D_{ij} = \partial h_i / \partial u_j$. The matrix D is assumed to have full row rank (rank$(D) = m$).

In order to arrive at an explicit expression for u it is necessary to linearise equation (2.4b) in a neighbourhood of $\bar{u}$

$$w = h(\bar{u}) + D\big(u - \bar{u}\big) \tag{2.5}$$

Rearrange this equation to read

$$Du = w - h(\bar{u}) + D\bar{u} \tag{2.6}$$

Premultiply equation (2.4a) with D to get

$$Du = (DD^T)\lambda \tag{2.7}$$

Substitute this into equation (2.6) and rearrange to get the following expression for λ

$$\lambda = (DD^T)^{-1}\big(D\bar{u} + w - h(\bar{u})\big) \tag{2.8}$$

Finally, substitute this into equation (2.4a) to obtain the following expression for u

$$u = D^T (DD^T)^{-1}\big(D\bar{u} + w - h(\bar{u})\big) \tag{2.9}$$

Linear equation method

Exactly the same – approximate – solution to the original optimisation problem can be obtained by first linearising equation (2.2). But then application of Lagrange's method in equation (2.3) is equivalent to finding the minimum norm solution of the system of linear equations

$$Du = D\bar{u} + w - h(\bar{u}) \tag{2.10}$$

where D is the jacobian of $h(\bar{u})$. The minimum norm solution for this system of linear equations is (see Gill, Murray and Wright (1991) and Cline and Plemmons (1976))

$$u = D^T(DD^T)^{-1}\big(D\bar{u} + w - h(\bar{u})\big) \tag{2.11}$$

or more generally This can also be written as

$$u = D^+\big(D\bar{u} + w - h(\bar{u})\big) \tag{2.12}$$

where D^+ is the Moore-Penrose inverse of D.[1]

If $\bar{u}$ was calculated in a similar manner with a matrix $\bar{D}$ identically equal to the current matrix D, then $\bar{u}$ is the minimum norm solution of a system $Du = b$. That implies $\bar{u} = D^+b$ and $D^+D\bar{u} = D^+b = \bar{u}$. Then equation (2.12) can be simplified to

$$u = \bar{u} + D^+\big(w - h(\bar{u})\big) \tag{2.13}$$

When the jacobian D is evaluated at every iteration of a full Newton-like algorithm then equation (2.12) should be used to obtain a minimum norm residual vector at each iteration. However, when the jacobian D is not evaluated at every iteration – for whatever reason – then equation (2.13) may be used whenever the jacobian is retained across iterations.

It is inadvisable to compute D^+ directly using DD^T since this can lead to loss of numerical accuracy. If $\kappa(D)$ is the condition number of matrix D then the condition number of DD^T will be $\kappa(D)^2$. This implies that even a moderately ill-conditioned D can lead to numerical inaccuracy.[2]

When D has full row rank, D^+ can be computed from the QR factorisation of D^T

$$D^T = QR \quad \Rightarrow \quad D^+ = QR^{-T}$$

where $Q \in \mathbb{R}^{n\times m}$ has orthonormal columns[3] and $R \in \mathbb{R}^{m\times m}$ is upper triangular.[4] Hence $u = D^+b$ can be computed with the following two steps

[1] This equation also holds when D does not have full row rank. See Gill et al. (1991) and Golub and Van Loan (1989).

[2] See Cline and Plemmons (1976).

[3] Therefore $Q^TQ = I$.

[4] where $R^{-T} = (R^T)^{-1}$.

1. solve $R^T z = b$
2. set $u = Qz$

This procedure is numerically stable. See Golub and Van Loan (1989) and Gill et al. (1991) for more details.

In numerical libraries a QR factorisation will often overwrite the original matrix. As a result calculation of $D\bar{u}$ would have to be done using $R^T Q^T$, which could introduce numerical inaccuracies. These may be avoided by using equation (2.13) whenever appropriate.

3 Computing the jacobian D

Column j of the fit jacobian matrix D contains the derivatives of the target endogenous variables with respect to the residual u_j. If the matrix can only be computed numerically, the following steps are taken to calculate a column of the D matrix. We assume a baseline solution has been calculated. Let u be the vector of *scaled* residuals and let $\bar{u}$ be the vector of current values of the residuals. Let q represent the vector of endogenous variables which are being targeted. And let $\bar{q}$ be the vector of current values of the targeted endogenous variables.

The method for computing the successive columns of the jacobian D proceeds as follows

1. Change a residual with a small perturbation (δ)

   $$u_j = \bar{u}_j + \delta$$

2. Solve the model to a specified convergence level
3. Calculate column j of the D matrix

   $$D_{.,j} = (q - \bar{q})/\delta$$

Remember that D is the jacobian with respect to the *scaled* residuals.

The constant δ is set to 0.1, which has proven to be adequate.

In our computer implementation[5] the steps for solving the model in this case are as follows

1. Make one pass through the model equations
2. Compute the Newton step for the feedback variables
3. Adjust the feedback variables
4. Make a second pass through the model equations.

The model is not solved to a specified convergence tolerance but two Newton iterations are taken. It has proven to be sufficient for achieving sensible outcomes.

[5] which uses the feedback method for solving normalised systems of equations; see Don and Gallo (1987).

4 Algorithm

An algorithm for computing the minimum norm solution of our nonlinear system of equations should allow for retaining the Jacobian D between iterations, to avoid possibly very expensive calculations. A norm on $w - h(u)$ is used for testing convergence. Define $Z = w - h(u)$. When the norm on Z does not show a certain minimal reduction and the jacobian matrix is up to date, the algorithm stops due to lack of progress. Finally, if convergence is slow as measured by insufficient relative reduction in the norm on Z, the Jacobian is recalculated. The algorithm in pseudo code is given in figure 4.1. The condition of the jacobian D with full row rank is estimated with the LINPACK condition estimator; if the estimated *inverse condition number* of D is less than the square root of the machine precision, then the algorithm stops; the jacobian is too ill-conditioned for the algorithm to proceed sensibly.

The norm on Z used in the algorithm for testing convergence and progress may be any norm (Euclidian, infinity or scaled infinity). The constant η_w should be small, typically 10^{-3}, but must be larger than the accuracy with which Z has been computed. For example, if Z has been computed using a convergence test of 10^{-4} then η_w should be larger than 10^{-4}.

The fixed constant $\gamma_p \in (0, 1)$ is intended to ensure sufficient decrease; it should be close to 1. In the actual computer implementation it has been set to 0.95; experiments in the distant past have shown that a larger value less than 1. was never successful as remedy for any numerical difficulties.

Finally the constant $\eta_s \in (0, \gamma_p)$ determines when to recalculate the jacobian D (the default is 0.5).

Figure 4.1 Fit algorithm

```
$k = 0$
$jaceval = \text{true}$
$flag = 0$
Initialize $u$
evaluate $Z = w - h(u)$
if $\|Z\| \le \eta_w$ then // Absolute convergence
    $flag = 1$
endif
while $flag = 0$ do
    $u_p = u$ // Save current values
    $Z_p = Z$
    if $jaceval$ then
        evaluate $D$ at $u_p$ using the procedure of paragraph 3
        compute $D\bar{u}$ with $\bar{u} = u_p$ and save result in temporary storage
        if $\text{Inversecondition}(D) < \sqrt{\varepsilon}$ then // Jacobian is too ill conditioned to proceed
            $flag = 4$
            continue // while loop
        endif
        evaluate $u$ from equation 2.12
    else
        evaluate $u$ from equation 2.13
    endif
    evaluate $Z = w - h(u)$
    if $\|Z\| \le \eta_w$ then // Absolute convergence
        $flag = 1$
    elseif $\|Z\| > \gamma_p \|Z_p\|$ then // Very bad or insufficient decrease
        // reset
        $u = u_p$
        $Z = Z_p$
        if $jaceval$ or $k \ge maxfiter$ then // Cannot locate better point
            $flag = 3$
        else // Try fresh jacobian
            $jaceval = \text{true}$
        endif
    elseif $k \ge maxfiter$ then // Too many iterations
        $flag = 2$
    else // Accept but if slow progress:  recalculate jacobian
        $jaceval = \|Z\| > \eta_s \|Z_p\|$
    endif
    $k = k + 1$
endwhile
// Save solution and residuals
```

5 Remarks

The algorithm has been tested on

- a historic medium-size (approximately 1500 equations) quarterly forecasting macro economic model using 27 residuals, 6 observations on the first forecasting quarter and 3 observations in the following three forecasting quarters. The algorithm needs two iterations in each quarter to converge.
- a current quarterly forecasting model with approximately 2500 equations, using 22 residuals and 22 observations and targets for the first quarter of the forecasting period and 8 residuals for the remaining 11 quarters. Here too, the algorithm needs two iterations to converge.
- the current annual medium term macroeconomic forecasting model, having approximately 2500 equations, using 22 residuals and 2 targets (unemployment and enterprise production). The algorithm needs 1 iteration to converge.

The procedure has also been used with success with much larger models and using large number of residuals and observations or targets.

These tests and others not mentioned explicitly give rise to the following remarks

1. when the functions relating targets and residuals – the function *h* in equation 2.2 – are implicitly determined by an iterative process as would normally be the case for an econometric model, great care must be taken in specifying the convergence threshold. It should not be smaller than the convergence threshold used in solving the underlying system and may even have to be considerably larger.
2. no global strategy such as a line search has been incorporated in the algorithm. Experiments with a backtracking line search did not yield satisfactory results, which may also have been caused by the method of solving the underlying system. Numerical problems could always be avoided by changing the specification of the underlying model.
3. since evaluating the Jacobian can be a very expensive process it may be worthwhile to investigate secant like adjustments to the Jacobian analogous to Broyden's method for square nonlinear equations.

References

Cline, R.E. and R.J. Plemmons, 1976, l_2–solutions to underdetermined linear systems, *SIAM Review*, vol. 18, no. 1, pp. 92–106.

Don, F.J.H. and G.M. Gallo, 1987, Solving large sparse systems of equations in econometric models, *Journal of Forecasting*, vol. 6, pp. 167–180.

Gill, P.E., W. Murray and M.H. Wright, 1991, *Numerical Linear Algebra and Optimization, Volume 1*, Addison-Wesley.

Golub, G.H. and C.F. Van Loan, 1989, *Matrix computations*, The John Hopkins University Press.

Sandee, J., 1994, Aanpassen in Simpc, CPB memo.

Sandee, J., F.J.H. Don and P.J.C.M. van den Berg, 1984, Adjustment of projections to recent observations, *European Economic Review*, vol. 26, pp. 153–166.