CPB Discussion Paper

No 54
24th January 2006

Solving large scale normalised rational expectations models

Olaf van 't Veer

CPB Netherlands Bureau for Economic Policy Analysis
Van Stolkweg 14
P.O. Box 80510
2508 GM The Hague, the Netherlands

Telephone +31 70 338 33 80
Telefax +31 70 338 33 50
Internet www.cpb.nl

ISBN 90-5833-251-9

Abstract in English

This paper discusses a new approach to solving models containing rational expectations. Instead of solving the model for each period consecutively as in the Fair-Taylor method, the method in this paper uses the idea of the Stacked-Time method to solve the model for all periods simultaneously.
The novelty of the method presented here is, that it is applied to a small subset of model variables only, the so called feedback variables, that an approximate Jacobian is used and that a subperiod method is introduced. This leads to significantly smaller Jacobians and less calculations for solving a model. The feedback variables are determined by an ordering algorithm.
The paper describes the modification to Newton's method by describing an Extended Feedback Jacobian, introducing an approximate 'Shift' Jacobian to reduce calculation time and a subperiod method to reduce storage space for the Jacobian matrix. The method has been implemented at the CPB and used for its GAMMA model. This paper also reports the results of some experiments with Multimod mark III. These experiments show faster convergence than the Fair-Taylor method and significantly smaller matrices than the Stacked-Time method.

Abstract in Dutch

Dit stuk beschrijft een nieuwe benadering voor het oplossen van modellen met rationele verwachtingen. In plaats van het oplossen van het model voor achtereenvolgende periodes, zoals in de Fair-Taylor methode, maakt de hier beschreven methode gebruik van het idee van de Stacked-Time methode om het model voor alle periodes simultaan op te lossen.
Het nieuwe aan de methode is dat deze slechts wordt toegepast op een kleine deelverzameling modelvariabelen, de zogenaamde feedback variabelen, dat gebruik wordt gemaakt van een benaderde 'Shift' Jacobiaan en dat een methode voor deelperiodes wordt beschreven. Dit leidt tot significant kleinere Jacobianen en minder berekeningen voor het oplossen van het model. De feedback variabelen worden bepaald met behulp van een volgorde algorithme.
Dit stuk beschrijft de aanpassing aan Newton's methode door de introductie van een Extended Feedback Jacobian, de introductie van een benaderde 'Shift' Jacobiaan voor het beperken van de rekentijd en een deelperiode methode voor het beperken van de benodigde opslagcapaciteit voor de Jacobiaan. De methode is geïmplementeerd op het CPB en wordt gebruikt voor het GAMMA model. Ook wordt verslag gedaan van de resultaten van een aantal experimenten met Multimod mark III. Deze experimenten laten snellere convergentie dan de Fair-Taylor methode en significant kleinere matrices dan de Stacked-Time methode zien.

Contents

Summary

This paper describes a new approach to solving models containing rational expectations. It avoids the problems of the Fair-Taylor method, which are slow convergence and often no convergence. It also avoids the storage requirements of the full Stacked-Time method. Instead of solving the model for each period consecutively as in the Fair-Taylor method, the method in this paper uses the idea of the Stacked-Time method to solve the model for all periods simultaneously. However, this approach is applied to a small subset of model variables only, the feedback variables. This leads to significantly smaller Jacobians and less calculations to solve the model. The set of feedback variables is determined by an ordering algorithm. This paper gives an overview of the ordering algorithm used and describes the extensions to the algorithm for lead variables.

As a starting point for the method the Extended Feedback Jacobian is described. This matrix contains the partial derivatives of the feedback variables for the full simulation period and is very sparse. The block structure of this matrix is explained.

To reduce calculation time a ‘Shift’ Jacobian is introduced. This matrix has the structure of the Extended Feedback Jacobian, however only a small part of the partial derivatives are actually calculated. The remaining derivatives are approximated. The appendix contains proof that the Newton method with the ‘Shift’ Jacobian converges q-linearly to the solution.

For more complicated and more heavily nonlinear models a subperiod method is introduced. Solving the model for overlapping subperiods using ‘Shift’ Jacobians makes the method applicable for a larger class of models.

A small example is given in which the ‘Shift’ Jacobian for a small model is derived numerically. The results of some experiments with Multimod mark III and GAMMA are also included.

Finally some suggestions for improvement and further research and some concluding remarks are made.

1 Introduction[1]

This paper describes a new approach to solving models with rational expectations. It avoids the problems of the Fair-Taylor method, which are slow convergence and often no convergence, (see Fair and Taylor (1983) for a description). It also avoids the storage requirements of the full Stacked-Time method (see Armstrong et al. (1995) for a description), which, in recent articles (see Dixon et al. (2005)), is still mentioned as the major problem of this method.

Before the model is solved it is ordered and the result of this ordering is a feedback set containing a (usually) small subset of the endogenous variables of the model. The ordering algorithm used requires the models to be normalised. The Newton method is applied to these feedback variables only, significantly reducing the size of the Jacobian matrix. The ordering method and the extension for lead variables are described in section 2 and appendix A.

The starting point of the method presented here is a new structure for the Jacobian matrix. It is defined for all feedback variables in all periods. Its structure is described in section 3. Derivatives of feedback variables, needed for the Jacobian matrix, are calculated somewhat differently than direct derivatives. Appendix C discusses this topic.

To reduce calculation time an approximation of the Jacobian can be used, the ‘Shift’ Jacobian. Only a fraction of the derivatives are actually calculated and these are used to approximate the remaining derivatives. Section 4 discusses this approximate Jacobian and appendix B proves that a Newton method with the Shift Jacobian converges q-linearly to the solution and derives the condition for which this property holds.

To reduce storage space the Newton process can be applied to subperiods of the full simulation path. This creates a second iterative stage over the extended Newton method. This process is described in section 5. This section also presents a pseudo code algorithm of the subperiod method.

Some practical examples are given in section 6. First some implementation issues are discussed. Then a small numerical example is given. Finally some tests are performed on IMF’s Multimod mark III and on CPB’s GAMMA models.

Finally some suggestions for improvement and some short concluding remarks are made.

[1] I wish to thank B.H. Hasselman, A.A. van der Giessen and D.A.G. Draper for their helpful comments.

2 Ordering the model

2.1 The standard Ordering method

The ordering method described here works for normalised models. In normalised models there is a clear relationship between left-hand side endogenous variables and right-hand side endogenous variables:

$$x_t = F\left(x_{t-h},\ldots,x_{t-1},x_t,x_{t+1},\ldots,x_{t+u};z_{t-i},\ldots,z_{t-1},z_t,z_{t+1},\ldots,z_{t+j}\right)$$

with x_k the endogenous variables with maximum lag h and maximum lead u and z_j the exogenous variables. The ordering method ignores the exogenous variables.

Ordering a model can be seen as a method for compressing the model into a set of feedback variables or a compressed form of the model. A Newton type method only needs derivatives of these variables for the Jacobian matrix to solve the complete model $F(x) = x$.

The ordering algorithm presented here splits a model in four parts:

$$\text{Model}\begin{cases}\text{Set of Recursive prologue variables, } X_p\\ \text{Simultaneous block}\begin{cases}\text{Set of Recursive simultaneous block equations, } X_s\\ \text{Set of Simultaneous feedback variables } X_{fb}\end{cases}\\ \text{Set of Recursive epilogue variables, } X_e\end{cases}$$

The prologue X_p contains those variables that depend only on lagged or exogenous variables or on previously calculated prologue variables. The recursive part of the simultaneous block X_s is ordered in such a way that, given values for X_{fb}, all X_s can be calculated recursively. Finally the epilogue X_e can be calculated recursively once X_p, X_s and X_{fb} have been calculated.

Don and Gallo (1987) and Levy and Low (1988) discuss this algorithm for compressing a model. Since the method presented in this paper is applied to the feedback set it is helpful to explain the ordering method in some detail. This is done in appendix A.

2.2 Extending the ordering for forward looking variables

The method mentioned in the previous section orders the model based on the current period structure of the model. The rules do not ensure that leads are put into the feedback set whenever possible or useful. To achieve this, a second stage is added to the ordering process.

First the rules as described in the previous section are applied. Using the ordering from this phase the model is ordered a second time, starting again with the initial incidence matrix M. Let the $N \times N$ incidence matrix of model $F(x) \in \mathbb{R}^N$ be defined such that

$$M_{i,j} = \begin{cases} 0 & \textbf{if } \frac{\partial f_i}{\partial x_j} = 0, \forall x_j \in \mathbb{R} \qquad i,j \rightarrow 1 \ldots N \\ 1 & \textbf{otherwise} \end{cases}$$

This time we start with an initial feedback set $\hat{X}_{fb}$which is created by the following rule:

Repeat

if x_i occurs in $F(x)$ with a lead **and** $x_i \in X_s$ **or** $x_i \in X_{fb}$ **or** $x_i \in X_e$ **then** add x_i to $\hat{X}_{fb}$ (2.1)

delete row i and column i

until no new x_i is found.

Finally $\hat{X}_{fb}$ is copied to X_{fb}, the sets X_p, X_s, X_e are cleared and a second ordering round is executed starting at the rule given in A.1.

This approach is fully heuristic and based on the idea that leads of prologue variables do not influence the Newton process. These can be calculated immediately. However leads of variables in the simultaneous set will influence the Newton process and have to be included in the feedback set. Leads in epilogue variables may influence the simultaneous set in the next period, so they also have to be in the feedback set.

Clearly this is far from a formal method of using lead information in the ordering of a model and more research needs to be done on this topic. The extension presented here doesn't take interdependencies between lead variables into account. Use of these interdependencies may lead to a smaller set of feedback variables and thus to smaller sizes of the resulting Jacobian matrices. Later in this paper some ideas about improving the ordering method will be given. A summary of the extended ordering algorithm is given in figure 2.1.

Figure 2.1 The Extended Ordering Algorithm

for *phase* = 1 **to** 2 **do**

repeat

if $\sum_j M_{i,j} = 0$ **then** add x_i to the end of prologue X_p and delete row i and column i (A.1)

until no new x_i are found.

repeat

if $\sum_i M_{i,j} = 0$ **then** add x_j to the start of epilogue X_e and delete row j and column j (A.2)

until no new x_j are found.

repeat

Apply compression rules from figure A.1

until $M = 0$

rebuild M

delete X_p, X_e and X_{fb} from M

repeat

if $\sum_j M_{i,j} = 0$ **then** add x_i to recursive simultaneous set X_s and delete row i and column i (A.8)

until $M = 0$

if *phase* $= 1$ **then**

rebuild M

repeat

if x_i occurs in $F(x)$ with a lead **and** $x_i \in X_s$ **or** $x_i \in X_{fb}$ **or** $x_i \in X_e$ **then**

add x_i to $\hat{X}_{fb}$ and delete row i and column i (2.1)

endif

until all leads have been checked

$X_p = \emptyset \quad X_e = \emptyset \quad X_s = \emptyset$

$X_{fb} = \hat{X}_{fb}$

endif

endfor

3 Modified Newton method: the Extended Feedback Jacobian

In this section the modified Newton method is described. As was mentioned in the previous section the Newton process is applied to the feedback variables only. So the model under consideration is actually a condensed version of N (feedback)equations of the original model of M equations, with $M \gg N$.

A (feedback) model is a series of expressions $f^i(x), 1 \leqslant i \leqslant N, x \in \mathbb{R}^N$ generating a result x^i for each period t of the simulation path.

The standard expression for the variation of a function $f^i(x)$ in period t is:

$$\mathrm{d}\, f_t^i(x) = \frac{\partial\, f_t^i(x)}{\partial x_t^1}\mathrm{d}x_t^1 + \frac{\partial\, f_t^i(x)}{\partial x_t^2}\mathrm{d}x_t^2 \cdots + \frac{\partial\, f_t^i(x)}{\partial x_t^N}\mathrm{d}x_t^N \tag{3.1}$$

This expression uses only variations in period t to derive the variation in f_t^i.[2] In models with rational expectations we would like to use information about the dynamics of the model, hidden in the intertemporal derivatives. Therefore equation 3.1 has to be extended for all periods of the simulation path:

$$\mathrm{d}\, f_t^i(x) = \sum_{j=1}^{T}\sum_{p=1}^{N} \frac{\partial\, f_t^i(x)}{\partial x_j^p}\mathrm{d}x_j^p, \qquad (1 \leqslant t \leqslant T, 1 \leqslant i \leqslant N) \tag{3.2}$$

For a system of N equations $F(x)$ the expression is:

$$\mathrm{d}F(x) = J(x)\mathrm{d}x \qquad F, x \in \mathbb{R}^{NT}, J \in \mathbb{R}^{NT} \times \mathbb{R}^{NT} \tag{3.3}$$

If vector x^T looks like $(x_1^1\, x_1^2 \ldots x_1^N\, x_2^1 \ldots x_t^i\, x_t^{i+1} \ldots x_T^N)$ then $J(x)$ has the following structure:

$$J(x) = \begin{pmatrix}
\frac{\partial f_1^1(x)}{\partial x_1^1} & \frac{\partial f_1^1(x)}{\partial x_1^2} & \cdots & \frac{\partial f_1^1(x)}{\partial x_t^j} & \frac{\partial f_1^1(x)}{\partial x_t^{j+1}} & \cdots & \frac{\partial f_1^1(x)}{\partial x_T^N} \\
\frac{\partial f_1^2(x)}{\partial x_1^1} & \ddots & \cdots & \vdots & \vdots & \cdots & \frac{\partial f_1^2(x)}{\partial x_T^N} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \cdots & \vdots \\
\frac{\partial f_t^i(x)}{\partial x_1^1} & \frac{\partial f_t^i(x)}{\partial x_1^2} & \cdots & \frac{\partial f_t^i(x)}{\partial x_t^i} & \frac{\partial f_t^i(x)}{\partial x_t^{i+1}} & \cdots & \frac{\partial f_t^i(x)}{\partial x_T^N} \\
\frac{\partial f_t^{i+1}(x)}{\partial x_1^1} & \vdots & \cdots & \vdots & \ddots & \cdots & \frac{\partial f_t^{i+1}(x)}{\partial x_T^N} \\
\vdots & \vdots & \cdots & \vdots & \vdots & \ddots & \vdots \\
\frac{\partial f_T^N(x)}{\partial x_1^1} & \multicolumn{5}{c}{\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots} & \frac{\partial f_T^N(x)}{\partial x_T^N}
\end{pmatrix} \tag{3.4}$$

Using 3.3 the Newton method for solving $x = F(x)$ can be written as[3]:

$$x_{k+1} = x_k - \lambda \left(J(x_k) - I\right)^{-1}\left(F(x_k) - x_k\right) \tag{3.5}$$

Here $J(x_k)$, the Jacobian, has the shape of 3.4 and λ is a relaxation factor.

[2] For ease of notation $\frac{\partial f(x)}{\partial x}$ is written instead of the full feedback expression. Appendix C on page 45 explains how these derivatives are calculated for feedback variables.

[3] Here $\mathrm{d}F(x_k) = -(G(x_k) - x_k)$ when solving for $F(x) = G(x) - x = 0$ and $\mathrm{d}x_k = (J_G(x_k) - I)^{-1}\mathrm{d}F(x_k)$.

The size of this $J(x_k)$ is much larger than the size of the corresponding Jacobian in the single period method. However the matrix is very sparse. If the maximum lead is u then clearly $\frac{\partial f_p^i(x)}{\partial x_t^j} = 0$ for all $t > p+u$.

The matrix $J(x_k)$ has the following block structure:

$$J(x) = \begin{pmatrix} D_1 & E_2^1 & \dots & E_T^1 \\ L_1^2 & D_2 & \dots & E_T^2 \\ \vdots & \vdots & \ddots & \vdots \\ L_1^T & L_2^T & \dots & D_T \end{pmatrix} \tag{3.6}$$

Where each D_i, E_i^j and L_i^j is an $N \times N$ submatrix. Here

$$D_i = \begin{pmatrix} \frac{\partial f_i^1(x)}{\partial x_i^1} & \frac{\partial f_i^1(x)}{\partial x_i^2} & \dots & \frac{\partial f_i^1(x)}{\partial x_i^N} \\ \frac{\partial f_i^2(x)}{\partial x_i^1} & \frac{\partial f_i^2(x)}{\partial x_i^2} & \dots & \frac{\partial f_i^2(x)}{\partial x_i^N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_i^N(x)}{\partial x_i^1} & \frac{\partial f_i^N(x)}{\partial x_i^2} & \dots & \frac{\partial f_i^N(x)}{\partial x_i^N} \end{pmatrix} \tag{3.7}$$

is the matrix of current period derivatives, which is identical to the single period Jacobian in period i from the original Newton method.

$$E_i^j = \begin{pmatrix} \frac{\partial f_j^1(x)}{\partial x_i^1} & \frac{\partial f_j^1(x)}{\partial x_i^2} & \dots & \frac{\partial f_j^1(x)}{\partial x_i^N} \\ \frac{\partial f_j^2(x)}{\partial x_i^1} & \frac{\partial f_j^2(x)}{\partial x_i^2} & \dots & \frac{\partial f_j^2(x)}{\partial x_i^N} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_j^N(x)}{\partial x_i^1} & \frac{\partial f_j^N(x)}{\partial x_i^2} & \dots & \frac{\partial f_j^N(x)}{\partial x_i^N} \end{pmatrix} \qquad i > j \tag{3.8}$$

is the matrix of lead derivatives, the effect of a change in period i on period j where $i > j$. Finally L_i^j has the structure of E_i^j, only here $i < j$, the matrix of lag derivatives. If the maximum lead in the model is u then

$$E_i^j = 0, \qquad \text{for } i > j+u \tag{3.9}$$

If lag effects die out after l periods then

$$L_i^j = 0, \qquad \text{for } j > i+l \tag{3.10}$$

In many econometric models $J(x)$ is very sparse and has a band structure. For example if $u = 1$

(most common) and $l = 1$ then $J(x)$ has the following structure

$$J(x) = \begin{pmatrix} D_1 & E_2^1 & 0 & \dots & \dots & 0 \\ L_1^2 & D_2 & E_3^2 & \ddots & \dots & 0 \\ 0 & L_2^3 & D_3 & E_4^3 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & 0 \\ \vdots & \dots & \ddots & L_{T-2}^{T-1} & D_{T-1} & E_T^{T-1} \\ 0 & \dots & \dots & 0 & L_{T-1}^T & D_T \end{pmatrix}$$

The standard Newton method requires recalculation of $J(x_n)$ at each step. However, there are several Quasi Newton methods, which postpone the recalculation of $J(x_n)$. One of these methods can be used here. In practice recalculation of $J(x_n)$ is postponed until $\frac{\|x_n - x_{n-1}\|}{\|x_{n-1} - x_{n-2}\|} > \eta$ with η some threshold for speed of convergence. In this way the method turns into

$$x_n = x_{n-1} - \big(J(x_k) - I\big)^{-1}(F(x_{n-1}) - x_{n-1}) \qquad k \leqslant n-1 \tag{3.11}$$

which is known to converge q-linearly in a δ-neighbourhood of x^* (See Kelley (1995), section 5.4.1)

$$\|x_n - x^*\| = K_J \Big[\|x_k - x^*\| + \|J(x_k) - J(x^*)\|\Big] \|x_{n-1} - x^*\| \tag{3.12}$$

Here K_J is a constant defined as

$$K_J = (1 + 2\gamma)\Big[\gamma \|F'(x^*)^{-1}\| + 16\|F'(x^*)^{-1}\|^2 \|F'(x^*)\| + 4\|F'(x^*)^{-1}\|\Big]$$

with γ the Lipschitz constant of $J(x)$.

4 Reducing the number of calculations: the 'Shift Jacobian'

Similar to the Quasi Newton methods, which do not recalculate $J(x)$ at each iteration, there is the option of not explicitly calculating each $\frac{\partial f_t^i(x)}{\partial x_p^j}$. If u is the maximum lead in the model then calculating derivatives for the first $u+1$ periods captures the lag and lead effects in the model. Setting

$$\frac{\partial f_t^i(x)}{\partial x_p^j} = 0 \qquad [t+u < p \leqslant T] \tag{4.1}$$

$$\frac{\partial f_t^i(x)}{\partial x_p^j} = \frac{\partial f_{t-1}^i(x)}{\partial x_{p-1}^j} = \frac{\partial f_{t-(p-u-1)}^i(x)}{\partial x_{u+1}^j} \qquad [u+1 < p \leqslant t+u] \tag{4.2}$$

reduces the number of derivatives from T^2N^2 to $(u+1)TN^2$. Using 4.2 implies that derivatives for $t > u+1$ will be approximated by those for $t = u+1$. The reuse of earlier derivatives can be seen as a shift in diagonal direction of the blocks D_i, E_i^j and L_i^j from 3.6. For $u = 1$ this looks like

$$\hat{J}(x) = \begin{pmatrix} D_1 & E_2^1 & 0 & \dots & \dots & 0 \\ L_1^2 & D_2 & \searrow & \ddots & \dots & 0 \\ L_1^3 & L_2^3 & \searrow & \searrow & \ddots & \vdots \\ \vdots & \vdots & \searrow & \searrow & \searrow & 0 \\ \vdots & \vdots & \searrow & L_2^3 & D_2 & E_2^1 \\ L_1^T & L_2^T & \dots & L_3^4 & L_2^3 & D_2 \end{pmatrix} \tag{4.3}$$

Because of the way the matrix is constructed it will be called the 'Shift Jacobian'. Using this approximation of $J(x)$ we can write the Newton method as:

$$x_{n+1} = x_n - \big(J(x_n) + \Delta(x_n) - I\big)^{-1}\big(F(x_n) - x_n\big) \tag{4.4}$$

where $\Delta(x_n)$ is the error matrix in point x_n, the difference between the true Jacobian $J(x_n)$ and the 'Shift Jacobian'. If for example $u = 1$ then $\Delta(x)$ looks like

$$\Delta(x) = \hat{J}(x) - J(x) = \begin{pmatrix} 0 & 0 & 0 & \dots & \dots & 0 \\ 0 & 0 & E_2^1 - E_3^2 & \ddots & \dots & 0 \\ 0 & 0 & D_2 - D_3 & E_2^1 - E_4^3 & \ddots & \vdots \\ \vdots & \vdots & \dots & \ddots & \ddots & 0 \\ \vdots & \vdots & \dots & L_2^3 - L_{T-2}^{T-1} & D_2 - D_{T-1} & E_2^1 - E_T^{T-1} \\ 0 & 0 & \dots & \dots & L_2^3 - L_{T-1}^T & D_2 - D_T \end{pmatrix} \tag{4.5}$$

If $\|\Delta(x)\|$ is such that $(J(x) + \Delta(x))^{-1}$ is an approximate inverse of $J(x)$ then this method can be shown to converge to x^*.

Theorem 4.1. *Assume that $F(x)$ has a solution and that $J(x)+\Delta(x)$ is nonsingular. Furthermore assume that $\mathscr{B}(\delta)=\{x \mid \|x-x^*\|<\delta\}$ and that $J(x)$ is Lipschitz continuous on $\mathscr{B}(\delta)$ with Lipschitz constant γ, implying that $\|J(x_i)-J(x_j)\| \leqslant \gamma\|x_i-x_j\|$. Finally assume that $\big(J(x_n)+\Delta(x_n)\big)^{-1}$ is an approximate inverse of $J(x_n)$. Then there exist $K>0$, $\delta>0$ and $\delta_1 \in (0,1)$ such that for $x_n \in \mathscr{B}(\delta)$*

$$x_{n+1}=x_n-\big(J(x_n)+\Delta(x_n)-I\big)^{-1}\big(F(x_n)-x_n\big)$$

is defined and satisfies

$$\|x_{n+1}-x^*\| \leqslant \big(K\delta+4\delta_1\kappa(J(x^*)-I)\big)\|x_n-x^*\|$$

where $\kappa(A)$ is the condition number of matrix A with respect to $\|.\|$.

This theorem shows that for sufficiently small δ and δ_1 the Newton method with the Shift Jacobian converges q-linearly to x^* in a neighbourhood $\mathscr{B}(\delta)$ of x^*. Proof of this theorem is given in Appendix B.

Theorem 4.2. *Assume that $J(x)+\Delta(x)$ is nonsingular. If there exists a $\delta_1 \in (0,1)$ such that*

$$\|\Delta(x_n)\| \leqslant \frac{\delta_1\|J(x_n)\|}{1-\delta_1}$$

which implies

$$\|I-\big(J(x_n)+\Delta(x_n)\big)^{-1}J(x_n)\| \leqslant \delta_1 < 1$$

then $\big(J(x_n)+\Delta(x_n)\big)^{-1}$ is an approximate inverse of $J(x_n)$.

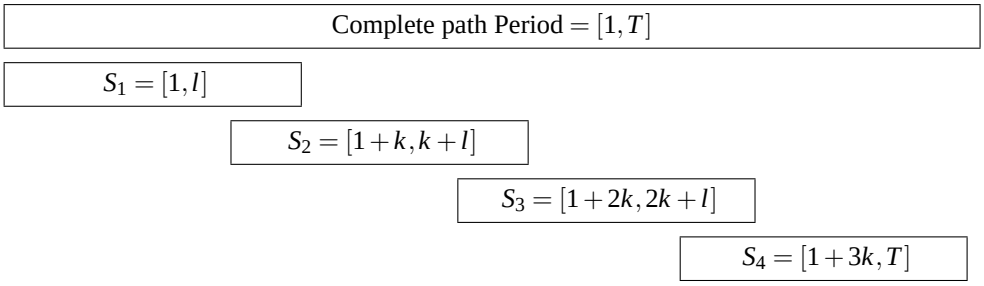
This theorem shows that the upper bound on $\|\Delta(x_n)\|$ is relative to $\|J(x_n)\|$. Proof of this theorem is also given in Appendix B. So if the partial derivatives of $F(x)$ are relatively stable over time, then the Shift Jacobian can definitely be used, since $\|\Delta(x)\|$ will be small in this case. For those models which do not satisfy the criterion for $\|\Delta(x)\|$ the next chapter will deal with another modification, which leads to smaller sizes of $J(x)$ and usually a smaller $\|\Delta(x)\|$.

5 Reducing storage space: the Subperiod method

The last two sections dealt with applying the Stacked-Time method to feedback variables and using an approximate Jacobian. This section uses some of the ideas on Type III iterations of the Fair-Taylor method (see Fair and Taylor (1983), section 2.1) to make the method suitable for more complicated and more heavily nonlinear models. Instead of calculating $J(x)$ for the complete simulation path $1 \dots T$, separate Jacobians for overlapping subperiods might be calculated. Then each subperiod i can be solved using the Newton method with a Jacobian $J_i(x_i)$. Since the length of a subperiod will be smaller than T, the size of the subperiod Jacobians $J_i(x_i)$ will be significantly smaller than the full Jacobian $J(x)$ for the complete period $1 \dots T$.

The solution process for the complete simulation path $1 \dots T$ is an iteration over the subperiods until convergence is reached in all subperiods.

As an example the following diagram shows the structure of the simulation path when it is divided into 4 subperiods with equal length l and stepping through the simulation path with step size k.



Here S_n is used to denote subperiod n. First the model is solved for S_1 using Jacobian $J_1(x_1)$. This is repeated for $S_2 \dots S_4$ using Jacobians $J_2(x_2) \dots J_4(x_4)$. Then one pass, or subperiod iteration, through the complete simulation path $1 \dots T$ has been made. These subperiod iterations continue until in all subperiods convergence is reached.

Using the overlap of $l-k$ periods between the subperiods ensures that results in subperiod $i-1$ and $i+1$ will influence subperiod i. The iterations over the subperiods ensure that a shock in any period will have an influence on all periods $1 \dots T$.

The parameters l and k can be freely chosen to define different lengths of subperiods and different step sizes. Using subperiod length l and step size k from subperiod to subperiod ($k < l$) defines $\frac{T-l}{k} + 1$ subperiods. If this expression is non-integer then the size of $J_s(x_s)$ for the last subperiod will be smaller, since the length of the remaining subperiod is smaller than l. The dimensions of the Newton method for subperiod s are $x_s \in \mathbb{R}^{Nl}, J_s(x_s) \in \mathbb{R}^{Nl} \times \mathbb{R}^{Nl}$.

Use of the subperiod method means that the original problem of solving the complete simulation path using an $NT \times NT$ Jacobian matrix is replaced by iteratively solving $\frac{T-l}{k} + 1$ subperiods using $Nl \times Nl$ Jacobian matrices.

This way, not only storage space is saved, but also execution time. In practice it appears that it is possible to reuse the submatrices in following subperiods, reducing storage space and execution time even more. For each subperiod i the matrix $J_{i-1}(x)$ is reused until $\|x_t - x_{t-1}\| > \eta \|x_{t-1} - x_{t-2}\|$ with η being some threshold for convergence speed. Choosing l and k such that $\frac{T-l}{k}$ is integer allows the Jacobian to be used for all subperiods, since all subperiod Jacobians have the same dimension.

It is also possible to combine the results of the previous section with the subperiod approach. For each subperiod Jacobian only the first $u+1$ derivatives are calculated. These values are shifted in diagonal direction as described in the previous section. Using the Shift Jacobian leads to $(u+1)lN^2$ evaluations of derivatives.

A pseudo code algorithm given in figure 5.1 illustrates the working of the subperiod method. Since the use of smaller subperiods leads to a smaller size of $J_s(x_n)$ and a relatively larger part of the derivatives are actually calculated ($\frac{u+1}{l} > \frac{u+1}{T}$) it is likely that $\|\Delta_s(x_n)\| \leqslant \|\Delta(x_n)\|$. So if a model converges q-linearly for the full-period method it will also converge q-linearly within the subperiods. If a model does not converge in the full-period method it is possible that it will reach convergence with the subperiod method.

It is difficult to determine optimal sizes for l and k. A smaller l leads to smaller Jacobians, but more subperiods. Therefore it may take longer to reach full convergence. A smaller k leads to a larger overlap between subperiods, which means that each subperiod is more 'solidly connected' to its neighbouring subperiods, but it also leads to more subperiods. As will be seen in section 6 the optimal values for l and k will depend on the model and the structure of the impulses given to the data.

Figure 5.1 The Subperiod Algorithm

$actjac = false$
initialise x
$totconv =$ convergence$\big(x_T, F_T(x_T)\big)$ // Check convergence for $x_{1..T}$
while not $totconv$ **and not** maximum iterations reached **do**
    $totconv = true$
    for $i = 1$ **to** T **step** k **do** // $k =$ subperiod step
        if $i + l - 1 \leqslant T$ **then** // $l =$ subperiod length; $k \leqslant l$ ensures overlapping
            set subperiod s to $[i, i + l - 1]$
        else
            set subperiod s to $[i, T]$ // Note $T - i + 1 < l$ when $\frac{T-l}{k}$ is non-integer
        endif
        if not convergence$(x_s, F_s(x_s))$ **then**
            $totconv = false$
            if not $actjac$ **then**
                calculate $J_s(x_s)$ or load $J_s(x_s)$ from file
                $actjac = true$
                $conv = false$
            endif
        else
            $conv = true$
        endif
        while not $conv$ **and not** maximum iterations reached **do**
            calculate $x_n = x_{n-1} - (J_s(x_{n-1}) - I)^{-1}(F_s(x_{n-1}) - x_{n-1})$
            $conv =$ convergence (x_n, x_{n-1})
            if $\|x_n - x_{n-1}\| > \eta \|x_{n-1} - x_{n-2}\|$ **then**
                step back to x_{n-1}
                calculate $J_s(x_{n-1})$ or load $J_s(x_{n-1})$ from file
            endif
        endwhile
    endfor
endwhile

6 Practical use

6.1 Some implementation issues

The method described in this paper has been implemented in ISIS, a proprietary software package developed by the CPB in cooperation with the University of Aarhus, Denmark, for solving large scale normalised econometric models and manipulating data.

For solving for dx in the linear system d$F = J(x)$dx, necessary to calculate the Newton steps, some basic Sparse Linear Algebra methods are needed. In this implementation the RGMRES Krylov-method is used (see Greenbaum (1997), section 2.4 for details). Although RGMRES needs more working space than alternative methods such as Bi-CGSTAB or CGS it appears to be the most robust method (see also Pauletto (1997)). When using Krylov-methods, preconditioning appears to be effective (see Kelley (1995) and Greenbaum (1997)). As a preconditioner an Incomplete LU-factorisation (ILU) is used. Alternatives could be Jacobi or SSOR preconditioning schemes, but these are less generally applicable. The actual implementations used are those contained in the NAG-library (mark 18).

The Jacobian and its ILU are stored on file. Thus they can be reused in several runs or, in the subperiod method, they can be loaded at each new subperiod step instead of recalculating them. Clearly this may save a lot of execution time for large models.

6.2 Small numerical example

In this section the Jacobian of a small model is derived numerically. The model is a closed economy 3-generation OLG model of consumption and saving derived from intertemporal optimisation of consumer utility. The equations of the model are

$$
\begin{aligned}
a^i &= w_{-1} - \left(\frac{1+r}{1+\theta}\right)^{\frac{1}{\gamma-1}} c^i \\
c^i &= (1+r)a^i + w - a^j_{+1} \\
a^j &= (1+r)a^i_{-1} + w_{-1} - \left(\frac{1+r}{1+\theta}\right)^{\frac{1}{\gamma-1}} c^j \\
c^j &= (1+r)a^j + w \\
r &= \beta B(n(a^i + a^j))^{\beta-1}
\end{aligned}
$$

where c^i_t and c^j_t are consumption of generations i and j in period t, a^i_t and a^j_t the assets of generations i and j at the beginning of period t and r the interest rate. The time index t has been omitted from the model equations. Lags are indicated by negative subscripts and leads by signed positive subscripts. The first generation has not been modelled explicitly, but can be calculated as $c^h = w - a^i_{+1}$ and $a^h = 0$. The parameter values used are

$$w = 1 \qquad \theta = 0.1 \qquad \gamma = 0.5 \qquad B = 0.2 \qquad \beta = 0.6 \qquad n = 100$$

Finally $a_0^i = 0.063$ and $a_T^j = 0.065$.

This model is split into two parts by the ordering method: 3 recursive simultaneous equations and 2 feedback equations. The 2 feedback variables are a^i and a^j. The model is solved for 4 periods ($T = 4$). The resulting shift Jacobian is[4]:

$$J(x) = \begin{pmatrix} & a_1^i & a_1^j & a_2^i & a_2^j & a_3^i & a_3^j & a_4^i & a_4^j \\ a_1^i & -1.008 & -0.003 & 0 & 0.991 & 0 & 0 & 0 & 0 \\ a_1^j & -0.009 & -1.014 & 0 & 0 & 0 & 0 & 0 & 0 \\ a_2^i & 0 & 0 & -1.008 & -0.003 & 0 & 0.991 & 0 & 0 \\ a_2^j & -1.023 & -0.003 & -0.009 & -0.009 & 0 & 0 & 0 & 0 \\ a_3^i & 0 & 0 & 0 & 0 & -1.008 & -0.003 & 0 & 0.991 \\ a_3^j & 0 & 0 & -1.023 & -0.003 & -0.009 & -0.009 & 0 & 0 \\ a_4^i & 0 & 0 & 0 & 0 & 0 & 0 & -1.008 & -0.003 \\ a_4^j & 0 & 0 & 0 & 0 & -1.023 & -0.003 & -0.009 & -0.009 \end{pmatrix}$$

In the structure of 4.3 this matrix consists of the following basic submatrices:

$$D_1 = \begin{pmatrix} & a_1^i & a_1^j \\ a_1^i & -1.008 & -0.003 \\ a_1^j & -0.009 & -1.014 \end{pmatrix} \qquad E_2^1 = \begin{pmatrix} & a_2^i & a_2^j \\ a_1^i & 0 & 0.991 \\ a_1^j & 0 & 0 \end{pmatrix}$$

$$L_1^2 = \begin{pmatrix} & a_1^i & a_1^j \\ a_2^i & 0 & 0 \\ a_2^j & -1.023 & -0.003 \end{pmatrix} \qquad D_2 = \begin{pmatrix} & a_2^i & a_2^j \\ a_2^i & -1.008 & -0.003 \\ a_2^j & -0.009 & -0.009 \end{pmatrix}$$

The submatrices D_2, E_2^1 and L_1^2 are shifted along the diagonal to construct $J(x)$. An interesting difference between D_1 and D_2 can be noticed. On the one hand in the matrix D_1 it can be seen that $\partial a_1^j / \partial a_1^j = -1.014$; on the other hand in the matrix D_2 we have $\partial a_2^j / \partial a_2^j = -0.009$. This is caused by the absence of a lag effect in period 1. Changes in period 2 also have an effect on values in period 1 through the lead variables of period 1 changing the lag values for period 2 (see Appendix C for a description). This effect is measured in E_2^1 and is only present for changes in periods > 1.

Using this sparse 8×8 system $J(x)$ the original 20×20 system can be solved. Using the feedback Jacobian the system contains 25 nonzeroes. If this model were to be solved using a Stacked-Time method with the complete model a Jacobian with 58 nonzeroes would be the result. To calculate the Shift Jacobian 16 passes through the model are needed (to calculate D_1, D_2, E_2^1 and L_1^2). In the full Stacked-Time method this would be 80 (5 model variables for 4 periods; each shock to a variable requires 4 passes through the model to calculate all

[4] Note that a_t^i and a_t^j are not part of the matrices. They are merely included for clarity.

derivatives). So, for this model, using the Shift Jacobian on feedback variables saves 56% of storage space and 80% of model iterations for the Jacobian.

Note that the same Jacobian matrix can be used to solve a much longer simulation path with the subperiod method (for example for $T = 200$ the model is solved in 3 subperiod iterations using this matrix).

6.3 MULTIMOD mark III

Multimod is a country model containing rational expectations and implicit equations (see Laxton et al. (1998) for details). In order to solve Multimod mark III with the new method the model (and the steady state model) had to be converted into a normalised system of equations. This has been done rather crudely, by replacing $f(x) = g(x)$ by $x = x + \alpha(g(x) - f(x)), (\alpha \neq 0)$. Clearly this leads to the same solution as the original model, but it has the drawback that each x treated in this way becomes a feedback variable, since they are self-dependent. Therefore this transformation leads to many more feedback variables than are strictly necessary. However for Multimod, the objective was not to get a minimal feedback set, but simply to see if the model could be solved using the Shift Jacobian and the subperiod method.

The ordering method decomposes the model in the following way

MULTIMOD Mark III
- 39 Prologue equations
- 499 Simultaneous equations
  - 348 Recursive
  - 151 Feedback variables
- 127 Epilogue equations

Using the normalised version two policy simulations were solved for 50 periods:

1. a temporary 1% increase in Government spending (CA_G) of Canada in the first period
2. a permanent 1% increase of the Money Target (CA_MT) of Canada in all periods

The policy simulations were solved using Shift Jacobians in 5 different ways:

1. the full-period method,
2. the subperiod method with subperiod length 20 (3 subperiods, $k = 15$),
3. the subperiod method with subperiod length 10 (9 subperiods, $k = 5$),
4. the subperiod method with subperiod length 5 (16 subperiods, $k = 3$),
5. the subperiod method with subperiod length 3 (25 subperiods, $k = 2$).

The Shift Jacobians are calculated using a maximum lead u of 1.[5] The resulting sizes of the

[5] In the model the actual maximum lead is 10 for the short term interest rates. But the model solves well using $u = 1$. This enables the use of very short subperiods.

Shift Jacobians are given in table 6.1.

Table 6.1 Jacobian sizes

T_{sub}	Matrix size	# Nonzeroes	% Density
3	205209	13010	6.34 %
5	570025	35570	6.24 %
10	2280100	132246	5.80 %
20	9120400	456932	5.01 %
50	57002500	2399805	4.21 %

Matrix size is the total number of elements of the Jacobian for this period ($= N^2 T_{sub}^2$, where N is the number of feedback variables). A large part of the nonzeroes were introduced by the normalisation process. Undoubtedly this could be done more sensibly, which would almost certainly lead to a substantial reduction in the amount of non zeroes.

The results of the solution runs are presented in table 6.2.

Table 6.2 1% increase of CA_MT for all periods

T_{sub}	Subperiod iter	Model iter	Avg. Newt. stp.	# Jacobians	# Jac. iter	Total model iter
3	17	3072	3	4	906	6696
5	16	6256	5	3	1510	10786
10	30	9299	5	2	3020	15339
20	36	9852	5	1	6040	15892
50	1	652	12	1	15100	15752
Fair-Taylor	335[a]			305[b]	151	89138

[a] Number of Fair-Taylor rounds
[b] Number of single period Jacobians needed

In this table

- *subperiod iter* is the number of subperiod iterations needed to reach full-period convergence. This is the number of times the total period 1..50 is passed using the subperiods,
- *model iter* is the total number of passes through the model for all the Newton steps (excluding Jacobian calculation). Here each iteration is one pass for 1 period, so calculating values for period 1..50 are 50 model iterations,
- *avg Newt. stp.* is the average number of newton steps in each subperiod needed to solve that subperiod,
- *# Jacobians* is the number of Shift Jacobians calculated during the run,
- *# Jac. iter* is the number of model iterations needed to calculated a Shift Jacobian for T_{sub},

- *Total model iter* is the total number of model iterations needed to solve the simulation path. This equals the number of iterations for the Jacobian times the number of Jacobians needed plus the model iterations for all the Newton steps.

For purpose of reference the last row of the table shows the results for this model run using the Fair-Taylor method. Only the comparable items have been reported. Here the column *subperiod iter* reports the number of rounds over the path $1 \ldots 50$ needed to solve the simulation path. This is comparable to subperiods with $T_{sub} = 1$. *# Jacobians* reports the number of single period Jacobians needed and *# Jac. iter* shows the number of model iterations needed to calculate the single period Jacobian ($= N$). The last column, *Total model iter*, is completely comparable, since this also reports the total number of model iterations needed to solve the path.

As can be seen in table 6.2 all runs using Shift Jacobian are between 5 and 13 times faster than the Fair-Taylor method. In terms of model iterations for the Newton steps the full-period method is superior. This can be expected since iterating over subperiods means it will take at least one pass through the complete path to spread the effects over all periods. In the full-period method this can be done in one Newton step.

In terms of total model iterations the smallest subperiod ($T = 3$) is superior. Although 17 rounds for the full period and 4 Jacobians are needed this still costs less than half of the iterations needed to calculate a Jacobian for the full period. This is caused by the fact that, although $T = 3$ defines 25 different subperiods, only 4 Jacobians are needed to solve all these subperiods. Clearly the Shift Jacobians are good approximations of the true Jacobians for these subperiods.

Another interesting result is that increasing the size of the subperiod leads to larger amounts of Newton steps needed to solve each subperiod. The explanation for this can be found in the size of $||\Delta(x)||$ for each subperiod. Since Multimod is strongly nonlinear, increasing the subperiod length will lead to a larger $||\Delta(x)||$, since more elements of the Jacobian are approximated, and this leads to decreasing convergence speed. This is the reason why, using the full-period Jacobian, 12 Newton steps are needed to solve the period.

This analysis compares several single runs. For these the smallest subperiod Shift Jacobian is superior to all others. If, however, the model user intends to run a number of policy simulations, using the same model (to inspect the effects of different exogenous impulses), the full-period Shift Jacobian is superior. In the first run this Jacobian can be stored, to be reused in all other runs. As table 6.2 shows: if the Jacobian doesn't have to be calculated the full-period Shift Jacobian is superior to all subperiod Jacobians (see column *model iter*).
In table 6.3 again the smallest subperiod method is superior to all other variations. Inspecting the solution results reveals that an increase in government spending in the first period has (decreasing) effects only on the first 22 periods. This means that the model is converged for most of the subperiods in the smallest subperiod case. Only a few iterations on the first subperiods are necessary to reach full convergence. Besides this, the argument sketched above still holds:

Table 6.3 1% increase of CA_G in the first period

T_{sub}	Subperiod iter	Model iter	Avg Newt. stp.	# Jacobians	# Jac. iter	Total model iter
3	2	32	<1	1	906	938
5	2	48	<1	1	1510	1558
10	2	55	<1	1	3020	3075
20	2	64	<1	1	6040	6104
50	1	52	1	1	15100	15152
Fair-Taylor	355[a]			203[b]	151	82702

[a] Number of Fair-Taylor rounds
[b] Number of single period Jacobians needed

smaller Jacobians are a better approximation of the real Jacobians in the case of Multimod mark III. Table 6.3 demonstrates that the approximate Jacobians aren't too bad, since the full-period method reaches convergence in one newton step. Finally the table shows that for this model run the differences with the Fair-Taylor method are spectacular. Using Shift Jacobians is between 5 and 88 times faster than the Fair-Taylor method.

6.4 GAMMA

The CPB developed the GAMMA model for ageing and pension studies (See Draper and Westerhout (2002) for a short description of the model). GAMMA models 85 generations of households (in an OLG-structure), firms, a public sector and different pension systems. Each household uses a lifecycle approach for planning consumption and asset holdings.

The model is made up of 25241 equations and is solved for 200 periods. The model contains 913 variables with leads and the maximum lead $u = 1$. The ordering method splits the model as follows

$$\text{GAMMA} \begin{cases} 8047 & \text{Prologue equations} \\ 1442 & \text{Simultaneous equations} \begin{cases} 948 & \text{Recursive} \\ 494 & \text{Feedback variables} \end{cases} \\ 15752 & \text{Epilogue equations} \end{cases}$$

Table 6.4 shows the sizes of the Shift Jacobians for different (sub-)period lengths.
As the table shows, the Shift Jacobians are very sparse. Since the maximum lead is 1, $2T_{sub}N^2$ model iterations are needed to calculate the Jacobian. Since the number of nonzeroes and the amount of fill-in is very small the ILU doesn't take much execution time. As a test with different sizes of subperiods GAMMA is solved with a 1% increase in period 20 of the size of the generation of 35 year old consumers. Using this impulse the model was solved for a period of

Table 6.4 Jacobian sizes

T_{sub}	Matrix size	# Nonzeroes	% Density
100	2440360000	432963	0.02 %
150	5490810000	1034239	0.02 %
200	9761440000	1639589	0.02 %

200 years. Table 6.5 shows the results.

Table 6.5 1% increase of the size of a generation in period 20

T_{sub}	Nr. subperiod iter	Model iter	Avg. Newt	# Jacobians
100	6	2100	3	1
150	2	750	3	1
200	1	400	2	1

No results for the Fair-Taylor method are given, because the model couldn't be solved using Fair-Taylor. This example shows that for GAMMA the full-period method is superior to the subperiod method. This is caused by the fact that in GAMMA impulses tend to have long lasting effects. The longer the period of the Jacobian, the more efficiently the Newton method will deal with this. GAMMA is not very nonlinear so $\|\Delta(x)\|$ will be small even for Jacobians over long periods.

7 Suggestions for improvement

7.1 The Ordering rules

Improving on the ordering algorithm for rational expectation models is subject of further study. The idea is that the feedback sets derived by the current (heuristic) extension of the algorithm might possibly be smaller if an intertemporal ordering algorithm is used. This implies a smaller size of the Jacobian matrices. Although the precise definition of the ordering rules are not quite clear yet, a sketch of the idea is given.

By analogy with the extended Jacobian, the incidence matrix M can be extended to capture all periods of the path, creating an intertemporal incidence matrix.

$$M \in \mathbb{R}^{MT} \times \mathbb{R}^{MT} \qquad M_{i_p,j_q} = \begin{cases} 0 & \frac{\partial f_p^i(x)}{\partial x_q^j} = 0 \\ 1 & \text{otherwise} \end{cases}$$

In this definition of M self dependent variables are still found on the diagonal. Only now, not only variable i's row and column are deleted, but the rows and columns of i for all periods.

To find intertemporal dependencies, it has to be checked if variable i in period t depends on i in future periods. So if

$$\sum_{k=t+1}^{MT} M_{i_t,i_k} > 0$$

then variable i is an intertemporal feedback and should be added to the feedback set.

This is just a rough sketch of the ideas on modifying the ordering algorithm. The rules for prologue and epilogue sets and the substitution rules would also need to be modified.

7.2 The Relaxation factor

In the current implementation the relaxation mechanism is simple. Instead of calculating a new Jacobian when $\|x_n - x_{n-1}\| > \eta \|x_{n-1} - x_{n-2}\|$, a smaller Newton step can be tried using a relaxation factor $\lambda < 1$. If, using this relaxation factor, there is still no convergence, λ can be set to a smaller value and the (smaller) step can be tried. There is a large amount of theory on how to decrease λ (see for instance Dennis, Jr. and Schnabel (1996) on line search algorithms). The method for decreasing λ in the current implementation is very simple: $\lambda_k = \max(\frac{\lambda_{k-1}}{2}, \lambda_{low})$. When convergence has been reached using λ in this way, the reverse operation is performed until convergence is reached with $\lambda = 1$.

Dennis, Jr. and Schnabel (1996) give more sophisticated algorithms of selecting values for λ, and it would be interesting to see if a reduction in the number of Newton steps would result if one of these algorithms was used.

8 Conclusion

This paper has developed the method of using Extended Feedback Jacobians, which appears to be a good (excellent) alternative to the full Stacked-Time method and the Fair-Taylor method in solving dynamic economic models with perfect foresight. It has also shown that approximate 'Shift' Jacobians can be used for the full period or iteratively for subperiods. All the models used in this paper converge in both the full-period method and the subperiod method using Shift Jacobians. Which method is superior (full-period or subperiod) depends largely on the model involved.

Proof has been given that the full-period method with the approximate Shift Jacobian converges q-linearly if the derivatives of the model are relatively stable over time. But even when they are not, as is the case with Multimod mark III, the method still performs well. Finally some suggestions for further improvement have been made. The most important improvement would be the modification of the ordering algorithm, since reducing the number of feedback variables implies less calculations and less storage space needed for the Jacobian matrices.

Appendix A The standard Ordering method

This section contains a formal describtion of the standard ordering method used on the CPB for models without rational expectations. As descibed in section 2, the aim of the ordering procedure is to split a normalised simultaneous model in four parts:

$$\text{Model}\begin{cases}\text{Set of Recursive prologue variables, } X_p\\ \text{Simultaneous block}\begin{cases}\text{Set of Recursive simultaneous block equations, } X_s\\ \text{Set of Simultaneous feedback variables } X_{fb}\end{cases}\\ \text{Set of Recursive epilogue variables, } X_e\end{cases}$$

Let the $N \times N$ incidence matrix of model $F(x) \in \mathbb{R}^N$ be defined such that

$$M_{i,j} = \begin{cases} 0 & \textbf{if } \frac{\partial f_i}{\partial x_j} = 0, \forall x_j \in \mathbb{R} \qquad i,j \to 1 \ldots N \\ 1 & \textbf{otherwise} \end{cases}$$

The prologue variables can now be determined by

Repeat

if $\sum_j M_{i,j} = 0$ **then** add x_i to the end of prologue X_p and delete row i and column i (A.1)

until no new x_i are found.

For the epilogue set a similar rule can be followed.

Repeat

if $\sum_i M_{i,j} = 0$ **then** add x_j to the start of epilogue X_e and delete row j and column j (A.2)

until no new x_j are found

The prologue set and epilogue set are ordered. The remaining equations form the simultaneous block. The following phase is the selection of feedback variables. Once this is done, the rest of the simultaneous block, or X_s, can be ordered in the last phase.

The following rules are performed to select the smallest possible set of feedback variables such that the remaining part of the model can be calculated recursively. A first selection of feedback variables can be done by selecting the self dependent variables, occurring on the diagonal of M.

Repeat

if $M_{i,i} = 1$ **then** add x_i to feedback set X_{fb} and delete row i and column i (A.3)

until no new x_i are found.

Removing these variables may have changed the structure of the simultaneous block so that a number of equations can now be ordered recursively.

Repeat

if $\sum_j M_{i,j} = 0$ **then** delete row i and column i (A.4)

until no new x_i are found.

If no more equations remain a minimal feedback set has been found to which a Newton algorithm can be applied. However, if there are still equations remaining a substitution is performed on M. In all equations determined by just one variable the left-hand side variable is replaced by the right-hand side variable for all occurrences of the left-hand side variable, thus removing a variable from the structure matrix M. All variables x_i used in just one equation are replaced by the left-hand side variable of that equation, also removing a variable from M.

Repeat

if $\sum_j M_{i,j} = 1 \wedge M_{i,p} = 1$ **then** $M_{i,k} = M_{i,k} \oplus M_{p,k} \qquad k \rightarrow 1 \ldots N$ (A.5)
delete row p and column p

until no new x_i is found.

Here $\oplus$ is the boolean-addition operator. And

Repeat

if $\sum_i M_{i,j} = 1 \wedge M_{p,j} = 1$ **then** $M_{p,k} = M_{p,k} \oplus M_{j,k} \qquad k \rightarrow 1 \ldots N$ (A.6)
delete row j and column j

until no new x_j are found.

These substitutions may lead to the emergence of new self dependent variables. Therefore the process from A.3 onward is repeated until no more variables are deleted or all variables have been processed. After application of all these rules, it is possible that the matrix M is not yet empty. In this case a heuristic rule is applied to select the variable that is most likely to occur in the largest number of feedback loops.

find i:

$\max_i \left((\sum_j M_{i,j})(\sum_j M_{j,i}) \right)$ and add x_i to feedback set X_{fb} (A.7)

and then continue with the substitution rules above until M is empty.

Now the feedback set has been generated. An overview of the compression rules is given in figure A.1.

Figure A.1 The compression rules

repeat
  repeat
    if $M_{i,i} = 1$ **then** add x_i to feedback set X_{fb} and delete row i and column i (A.3)
  until no new x_i are found.
  repeat
    if $\sum_j M_{i,j} = 0$ **then** delete row i and column i (A.4)
  until no new x_i are found.
  repeat
    if $\sum_j M_{i,j} = 1 \wedge M_{i,p} = 1$ **then** $M_{i,k} = M_{i,k} \oplus M_{p,k} \qquad k \rightarrow 1 \ldots N$ (A.5)
      delete row p and column p
    endif
  until no new x_i is found.
  repeat
    if $\sum_i M_{i,j} = 1 \wedge M_{p,j} = 1$ **then** $M_{p,k} = M_{p,k} \oplus M_{j,k} \qquad k \rightarrow 1 \ldots N$ (A.6)
      delete row j and column j
    endif
  until no new x_j are found.
until no new x_i are found in this repeat-round.
if $M \neq 0$ **then**
    find i for which $\max_i \Big(\big(\sum_j M_{i,j}\big) \big(\sum_j M_{j,i}\big) \Big)$ and add x_i to feedback set X_{fb} (A.7)
endif

The last step is to order the recursive set of simultaneous equations, X_s. To do this the incidence matrix M is regenerated. Then the following rules are applied to it:

foreach $x_i \in X_p$ delete row i and column i
foreach $x_i \in X_e$ delete row i and column i
foreach $x_i \in X_{fb}$ delete row i and column i
Repeat

if $\sum_j M_{i,j} = 0$ **then** add x_i to recursive simultaneous set X_s (A.8)
delete row i and column i

until M is empty.

When all feedback variables in X_{fb} have been selected by A.3 (the selfloop rule) then the feedback set is a minimal feedback set (see Levy and Low (1988) for a graph theoretical formal proof).

An interesting extension to the ordering method described in this section is given in Hasselman (2004), discussing a method to efficiently reduce the size of the feedback set by detecting and removing redundant feedback variables.

Appendix B Proofs of Theorems

B.1 Proof of Local convergence for the 'Shift Jacobian'

In this section the proof for local convergence of the Newton method with the Shift Jacobian is given. For this purpose the Newton method for solving $F(x) = x$

$$x_{n+1} = x_n - \big(J(x_n) - I\big)^{-1}(F(x_n) - x_n) \tag{B.1}$$

is rewritten as

$$x_{n+1} = x_n - \tilde{J}(x_n)^{-1}\tilde{F}(x_n) \tag{B.2}$$

by setting

$$\tilde{F}(x_n) = F(x_n) - x_n$$

and

$$\tilde{F}'(x_n) = \tilde{J}(x_n) = J(x_n) - I$$

so that $\tilde{F}(x^*) = 0$ as in the standard Newton method.

Using the Shift Jacobian as an approximation to $J(x_n)$ B.2 turns into

$$x_{n+1} = x_n - \big(\tilde{J}(x_n) + \Delta(x_n)\big)^{-1}\tilde{F}(x_n) \tag{B.3}$$

Let x^N_{n+1} be the result of the pure Newton step with the true Jacobian $\tilde{J}(x_n)$, then

$$x_{n+1} = x^N_{n+1} + \big[\tilde{J}(x_n)^{-1} - \big(\tilde{J}(x_n) + \Delta(x_n)\big)^{-1}\big]\tilde{F}(x_n) \tag{B.4}$$

and

$$\|x_{n+1} - x^*\| \leqslant \|x^N_{n+1} - x^*\| + \|\tilde{J}(x_n)^{-1} - \big(\tilde{J}(x_n) + \Delta(x_n)\big)^{-1}\|\|\tilde{F}(x_n)\| \tag{B.5}$$

From the proofs of local convergence of the Newton method (See Kelley (1995), section 5.1 and Dennis, Jr. and Schnabel (1996), section 5.2) it is known that

$$\|x^N_{n+1} - x^*\| \leqslant K\|x_n - x^*\|^2$$

for each $x_n \in \mathscr{B}(\delta)$, where constant $K = \gamma\|F'(x^*)^{-1}\|$ with γ the Lipschitz constant.

One of the basic Lemma's from Kelley (1995) (Lemma 4.3.1) states that for each $x \in \mathscr{B}(\delta)$

$$\|F(x)\| \leqslant 2\|F'(x^*)\|\|x - x^*\|$$

Substituting these results in B.5 gives

$$\|x_{n+1} - x^*\| \leqslant K\|x_n - x^*\|^2 + 2\|\tilde{J}(x_n)^{-1} - \big(\tilde{J}(x_n) + \Delta(x_n)\big)^{-1}\|\|\tilde{J}(x^*)\|\|x_n - x^*\| \tag{B.6}$$

Let $\|\Delta(x_n)\|$ be bounded so that

$$\|I - \left(\tilde{J}(x_n) + \Delta(x_n)\right)^{-1}\tilde{J}(x_n)\| \leqslant \delta_1 < 1 \tag{B.7}$$

This means that $\left(\tilde{J}(x_n) + \Delta(x_n)\right)^{-1}$ is an approximate inverse of $\tilde{J}(x_n)$. B.6 can be rewritten as

$$\|x_{n+1} - x^*\| \leqslant K\|x_n - x^*\|^2 + 2\|\tilde{J}(x_n)^{-1}\left(I - \left(\tilde{J}(x_n) + \Delta(x_n)\right)^{-1}\tilde{J}(x_n)\right)\|\|\tilde{J}(x^*)\|\|x_n - x^*\|$$

Now using B.7

$$\|x_{n+1} - x^*\| \leqslant K\|x_n - x^*\|^2 + 2\delta_1\|\tilde{J}(x_n)^{-1}\|\|\tilde{J}(x^*)\|\|x_n - x^*\| \tag{B.8}$$

The Lemma of Kelley (1995) (Lemma 4.3.1) also states that for each $x \in \mathscr{B}(\delta)$

$$\|F'(x)^{-1}\| \leqslant 2\|F'(x^*)^{-1}\|$$

So we get

$$\|x_{n+1} - x^*\| \leqslant K\|x_n - x^*\|^2 + 4\delta_1\|\tilde{J}(x^*)^{-1}\|\|\tilde{J}(x^*)\|\|x_n - x^*\| \tag{B.9}$$

Since $\kappa(A) = \|A^{-1}\|\|A\|$, the condition number relative to $\|.\|$ this equals

$$\|x_{n+1} - x^*\| \leqslant K\|x_n - x^*\|^2 + 4\delta_1\kappa(\tilde{J}(x^*))\|x_n - x^*\| \tag{B.10}$$

and

$$\|x_{n+1} - x^*\| \leqslant \left[K\delta + 4\delta_1\kappa(\tilde{J}(x^*))\right]\|x_n - x^*\| \tag{B.11}$$

which in terms of the original model $F(x_n)$ gives:

$$\|x_{n+1} - x^*\| \leqslant \left[K\delta + 4\delta_1\kappa(J(x^*) - I)\right]\|x_n - x^*\| \tag{B.12}$$

with $\kappa(J(x^*) - I)$ being the condition number of $J(x^*) - I$. With δ and δ_1 small enough the iteration process converges q-linearly to x^*. This concludes the proof. □

B.2 Proof of the Upper Bound for $\|\Delta(x_n)\|$

This section gives the proof for the upper bound on $\|\Delta(x_n)\|$ such that $(J(x_n) + \Delta(x_n))^{-1}$ is an approximate inverse of $J(x_n)$.

The necessary restriction on $\|\Delta(x_n)\|$ for B.12 to be satisfied can be derived using the Banach Lemma (see Kelley (1995), Theorem 1.2.1)

$$\|A - B^{-1}\| = \frac{\|A\|\|I - BA\|}{1 - \|I - BA\|}$$

where B is an approximate inverse of A, and the definition of an approximate inverse (see Kelley (1995), definition 1.2.1)

$$\|I - BA\| \leqslant \delta_1 < 1$$

Taking A to be $J(x_n)$ and B to be $\big(J(x_n)+\Delta(x_n)\big)^{-1}$ the definition becomes

$$\|I-\big(J(x_n)+\Delta(x_n)\big)^{-1}J(x_n)\| \leqslant \delta_1 < 1 \tag{B.13}$$

and the Banach Lemma evaluates to

$$\|J(x_n)-\big(J(x_n)+\Delta(x_n)\big)\| = \frac{\|J(x_n)\|\|I-\big(J(x_n)+\Delta(x_n)\big)^{-1}J(x_n)\|}{1-\|I-\big(J(x_n)+\Delta(x_n)\big)^{-1}J(x_n)\|} \tag{B.14}$$

Using B.13 this results in an upper bound for $\|\Delta(x_n)\|$

$$\|\Delta(x_n)\| \leqslant \frac{\delta_1\|J(x_n)\|}{1-\delta_1} \tag{B.15}$$

which completes the proof. □

Appendix C Derivatives of feedback variables

C.1 Period-by-period case

This section shows how derivatives of the feedback variables (for the Jacobians) are calculated. The ordering algorithm splits the model in several sections:

$$F(x)\begin{cases} F_p(x) & \text{prologue set} \\ F_s(x) & \text{recursive part of the simultaneous set} \\ F_{fb}(x) & \text{feedback part of the simultaneous set} \\ F_e(x) & \text{epilogue set} \end{cases} \tag{C.1}$$

For Newton type methods the Jacobian is calculated for the feedback variables only. Calculating the partial derivative $\frac{\partial f^i_{fb}(x)}{\partial x_j}$ where x_j is a feedback variable, would ignore the effects of the other equations in the recursive simultaneous set $F_s(x)$. This set, together with $F_{fb}(x)$, forms the simultaneous block of the model. To ensure all the simultaneous effects are taken into account in calculating the derivatives the following term is calculated instead of a direct derivative of function f^i_{fb}

$$\frac{\partial f^i_{fb}(F_s(x))}{\partial x_j} = f^{i'}_{fb}(F_s(x))\frac{\partial F_s(x)}{\partial x_j} \tag{C.2}$$

In words: first the effects of the change in x_j on all functions of F_s are calculated and these are then used to calculate the total effect on function f^i_{fb}.

For the implementation this means that in order to calculate the difference approximation $\nabla_i f_{fb}(x)$ we first give an impulse Δx_i to x_i, calculate $F_s(x+\Delta x_i e_i)$ and then use the values of F_{fb} to calculate the differences:

$$\nabla_i F_{fb}(x) = \frac{F_{fb}(F_s(x+\Delta x_i e_i)) - F_{fb}(F_s(x))}{\Delta x_i} \tag{C.3}$$

C.2 Intertemporal case

For the method presented in this paper not only current period derivatives are required, but also derivatives of lags and leads. To capture the effects of a change in lags/leads on the current period it is not sufficient to calculate $F_{fb}(F_s(x))$, since this only takes current period effects into account.

Suppose we want to capture the effect of a change in x^i_t (with x^i a feedback variable) on function values in period p. Then there are three cases to consider

- $t < p$, a lag impulse. To fully capture all the lag effects resulting from this impulse the complete model has to be calculated from period $t-u$ to p (u being the maximum lead)

- $t = p$, a current period impulse. Here we also have to calculate the model from period $t-u$ to p, since x_t^i may influence $F_{t-u}(x)$
- $t > p$, a lead impulse. If $t > p+u$ we have no effect otherwise we only calculate $F_p(x)$ to capture the effects.

The three cases can be captured by the following general expression:

$$\nabla_{i,t} F_{fb,p}(x) = \frac{F_{fb,p}(F(x + \Delta x_t^i e_t^i)) - F_{fb,p}(F(x))}{\Delta x_t^i} \tag{C.4}$$

with $F(x) \in \mathbb{R}^{NT}$ the vector of function values for all periods of the path.

References

Armstrong, J., R. Black, D. Laxton and D. Rose, 1995, A robust method for simulating Forward-looking models, Part 2. Technical Report No. 72, Bank of Canada.

Dennis, Jr., J.E. and R.B. Schnabel, 1996, *Numerical methods for unconstained optimization and nonlinear equations*, Classics in Applied Mathematics, SIAM.

Dixon, P., K. Pearson, M. Picton and M. Rimmer, 2005, Rational expectations for large CGE models: A practical algorithm and a policy application, *Economic Modelling*, vol. 22, pp. 1001–1019.

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

Draper, N. and E. Westerhout, 2002, Ageing, sustainability and the interest rate: the GAMMA model, *CPB-Report*, vol. 2002/4, pp. 38–41.

Fair, R. and J. Taylor, 1983, Solution and Maximum likelihood estimation of dynamic nonlinear rational expectations models, *Econometrica*, vol. 51, pp. 1169–1185.

Greenbaum, A., 1997, *Iterative methods for solving linear systems*, Frontiers in Applied Mathematics, SIAM.

Hasselman, B.H., 2004, An efficient method for detecting redundant feedback vertices, CPB Discussion Paper 29.

Kelley, C.T., 1995, *Iterative methods for linear and nonlinear equations*, Frontiers in Applied Mathematics, SIAM.

Laxton, D., P. Isard, H. Faruqee, E. Prasad and B. Turtelboom, 1998, Multimod mark III, The core dynamics and steady state models, IMF Occasional Papers 164, IMF.

Levy, H. and D.W. Low, 1988, A contraction algorithm for finding small cycle cutsets, *Journal of Algorithms*, pp. 470–493.

Pauletto, G., 1997, *Computational solution of Large-Scale Macroeconometric models*, vol. 7 of *Advances in Computational Economics*, Kluwer Academic Publishers.