ANALYSES IN MACROECONOMIC MODELLING

Advances in Computational Economics

VOLUME 12

The titles published in this series are listed at the end of this volume.

Analyses in Macroeconomic Modelling

edited by

Andrew Hughes Hallett
University of Strathclyde

and

Peter McAdam
University of Kent at Canterbury

Springer-Science+Business Media, LLC

Library of Congress Cataloging-in-Publication Data
Analyses in macroeconomic modelling / edited by Andrew Hughes Hallett and Peter McAdam.
p. cm. -- (Advances in computational economics; v. 12)
Includes bibliographical references and index.
ISBN 978-1-4613-7378-0 ISBN 978-1-4615-5219-2 (eBook)
DOI 10.1007/978-1-4615-5219-2
1. Macroeconomics--Mathematical models. 2. Rational expectations (Economic theory) -- Mathematical models. I. Hughes Hallett, Andrew. II. McAdam, Peter. III. Series.
HB 172.5.A5 1999
339' .01' 5195 -- dc21 99-44507
CIP

Originally published by Kluwer Academic Publishers in 1999
Softcover reprint of the hardcover 1st edition 1999

Printed on acid-free paper.

TABLE OF CONTENTS

PART FOUR: LONG RUN AND CLOSURES IN MACRO MODELS

LIST OF CONTRIBUTORS

Gary S. Anderson
Federal Reserve Board of Governors

Andrew J. Hughes Hallett
University of Strathclyde

Michel Juillard
University Paris 8 and CEPREMAP

Douglas Laxton
International Monetary Fund

Peter McAdam
University of Kent at Canterbury

Pierre Malgrange
CEPREMAP

Scott Moss
Manchester Metropolitan University

Werner Roeger
European Commission

Steven Symansky
International Monetary Fund

Jan in't Veld
European Commission

Stephen Hall
Imperial College of Science Technology and Medicine

Sean Holly
University of Cambridge

David A Kendrick
University of Texas

Coung Le Van
CEPREMAP

Warwick McKibbin
Australian National University and the Brookings Institution

P. Ruben Mercado
University of Texas

Hope Pioro
Bank of Canada

Esther-Mirjam Sent
University of Notre Dame

Paul Turner
University of Sheffield

PART ONE

SOLUTION METHODS

CHAPTER 1

SOLUTION METHODS AND NON-LINEAR FORWARD-LOOKING MODELS[1]

Michel Juillard, Douglas Laxton, Peter McAdam and Hope Pioro

1. INTRODUCTION

In this paper, we compare the performance of two leading algorithms now regularly used to solve large forward-looking models. In particular we relate traditional Fair-Taylor 'extended path' algorithms to the newer breed of stacked Newton Raphson ones.[2] As a testing ground for these solution methods we use the IMF's world econometric model,MULTIMOD (Mason et al,1990) which may be considered typical of many current forward-looking large macro-models.[3]

The organisation of the paper is as follows: in section II we review the structure of both types of algorithms and review their general convergence properties. Section III extends the discussion to forward-looking looking models. In section IV we compare the performance of the Fair-Taylor algorithm and a new state-of-the-art Newton Raphson algorithm called NEWSTACK that is available in portable TROLL. Section VI provides some illustrative simulation times from NEWSTACK that were obtained from several different macroeconomic models and computer platforms. Finally, in section VII we offer conclusions.

2. TRADITIONAL SOLUTION METHODS

In this section, we illustrate the mechanics and convergence properties of our chosen solution methods. These results are already well known and so merely motivate and support the remainder of the paper. We first deal with first order then discuss

[1] We thank Ralph Bryant, Don Coletti, Peter Hollinger, Andrew Hughes Hallett, Ben Hunt, Tiff Macklem, Guy Meredith, Susanna Mursula, Steve Symansky, Bob Tetlow, Jakob Toftgard and Jan in't Veld for helpful comments and for providing simulation analyses on several models. The usual disclaimer applies.

[2] This is not to suggest that they are the *only* methods of solving forward-looking models only that they would appear to be the most popular - viable alternatives include Penalty function methods (Holly and Zarrop,1983) and shooting methods (Lipton et al,1982).

[3] MULTIMOD is an annual estimated econometric model containing the G7 countries as well as distinct other country blocks. Each country incorporates 53 equations (of which 34 are identities) and for which there are roughly 19 exogenous variables including, principally, monetary target, government expenditure, debt target, oil price and population etc. A full description of MULTIMOD's properties and simulation characteristics is given in Masson et al (1990) and model vintages in Helliwell et al (1990) and Mason et al (1988).Exercises in cross model comparisons may be found in Bryant et al (1988,1993) and Mitchell et al (1995) ,amongst others. Finally note that for these exercises we use the basic production vintage of the MULTIMOD model - MULTAR. With some exceptions - such as the ERM members' (cubic) monetary reaction function and the obvious case of price deflators and log-linear functional forms - the model is highly linear and so should, in principle, be relatively straight forward to solve.

Newton-type methods. We also deal with the necessary extensions to these algorithms to solve models with lead variables separately in section three.

In linear systems, solutions methods merely require the substitution of exogenous (and pre-determined) variables into the reduced form. To illustrate given a structural estimated model,

$$AY + BX = U$$

we can solve for the reduced form - assuming that the A matrix is square and of full rank:

$$Y = \Pi X + V$$

where, $\Pi = -(A)^{-1} B$; $V= (A^{-1}) U$.

In non-linear systems, however, the impact and dynamic multipliers embodied Π are base and perturbation dependent and so yield no unique reduced form. Such systems therefore are solved iteratively with the initial search based on a series of "first-guesses" or "starts" for each endogenous variable, usually their lagged values.

2.1 First Order Algorithms

In a first-order solution (Gauss-Seidel, Jacobi or variants) the iterative solution is of the form:

$$y^s = Gy^{s-1} + b_s$$

In Gauss-Seidel (GS) we progress by solving equations sequentially (based on first-guesses for the endogenous variables Y^{s-1} or Y^0 in the case of the first iteration) with an exogenous variable set. In other words:

$$Y^{s1t} = \Sigma^n_{j=2} b_{ij} Y^{s-1}{}_{jt} + b_{1t}$$

$$Y^{s2t} = b_{21} Y^s{}_{1t} + \Sigma^n_{j=3} b_{ij} Y^{s-1}{}_{jt} + b_{2t}$$

$$Y^{s3t} = b_{31} Y^s{}_{1t} + b_{32} Y^s{}_{1t} + \Sigma^n_{j=4} b_{ij} Y^{s-1}{}_{jt} + b_{3t}$$

.
.
.

$$Y^s{}_t = LY^s{}_t + UY^{s-1}{}_t + b_t$$

collecting terms,

$$Y^s_t = (I-L)^{-1}UY^{s-1}_t + (I-L)^{-1}b_t$$

The iteration pattern therefore depends on values already solved for (indicated L for lower) and values for which there is not yet a new solution because they are solved further down in the model (indicated U for upper).This last equation can be re-written as:

$$Y^s_t = GY^{s-1}_t + k_t \qquad (1)$$

where $G = (I-L)^{-1}U$ and $k = (I-L)^{-1}b_t$.

This iterative process terminates only when the values of the endogenous variables in successive iterations converge to a pre-set tolerance(δ):

$$\max_j \left| (y^s - y^{s-1})/y^{s-1} \right| < \delta$$

for all j,j=1,2,3,..,n,subject to a maximum iterations setting.

Popular extensions to this basic GS scheme include Successive Over-relaxation (SOR) and Fast-Gauss-Seidel (FGS) in which the Gauss-Seidel result are further damped or extrapolated according to the iteration matrices:

$$G^{SOR} = (I-\alpha L)^{-1}(\alpha U + (1-\alpha)I)$$

$$G^{FGS} = \gamma(I-\alpha L)^{-1}(\alpha U + (1-\alpha)I) + (1-\gamma)I \qquad (2)$$

Note that variations are nested within (2) - with (α=1,γ=1) retrieving (pure) Gauss-Seidel,($\alpha\neq1$,γ=1) retrieving Gauss-Seidel plus SOR and (α=1,$\gamma\neq1$) yielding Fast Gauss-Seidel with no SOR.[4]

The convergence properties for these schemes are reasonably well know. Equation (1) converges if the spectral radius (i.e., the absolute value of the largest eigenvalue) of the iteration matrix, G, is less than unity - see Young (1971). The FGS case converges for some non-zero γ if the real parts of the eigenvalues of G are all less than unity - see Hughes-Hallett (1981). In the SOR case there are limited convergence results; it is convergent if the for some $\alpha\neq0$ (where we require $0<\alpha<2$ for convergence) if all the roots of (I-B) are less than unity in real parts.[5] However the *general* properties of SOR are well known to programmers and modellers alike (see Dorn and McCracken

---

[4] There are also the much less popular Jacobi and Jacobi Over-Relaxation Methods; in the case of the Jacobi iteration pattern we use only past iteration values in the current iteration update whilst GS uses (where possible) the value of current iterations in solving forward. Gauss-Seidel tends to be considered the faster of the two (Fair,1984). In the context of our previous discussion this yields :
$Y^s_t = GY^{s-1}_t + k_t$; where G = U and k = b_t.

[5] Note an $\alpha\approx0$ setting converges super quick but is a meaningless one since the simulation tends to the (uncooked) baseline as the new iterated values are 'damped away'.

(1972) for a simple example and Hughes-Hallett (1981) for SOR searches on a range of vintage US models) with the iteration number-SOR space a unique minimum.

Equivalent results hold for non-linear models: the iteration matrix,G^{s-1},and the forcing function, k , will be base and iteration dependent:

$$Y^s_t = G^{s-1}Y^{s-1}_t + k_s$$

Convergence requires that the spectral radius of the iteration matrix evaluated at the true Solution ,G^{YSol}, is less than unity.

2.2 Newton Raphson Algorithms

Given a non-linear model where y and x represent endogenous and exogenous variable sets respectively and f, the model's functional form,

$$f_i(y,x) = 0 \quad i=1,2,...,n$$

then the Newton Raphson (NR) solution is based on an expansion around a starting solution y^{s-1}:

$$y^s = y^{s-1} - (F^{s-1})^{-1} f(y^{s-1},x)$$

Where F, the Jacobian, is the matrix of partial derivatives evaluated at the present iteration :

$$F^{s-1} = [\partial f/\partial y]_{y(s-1)}$$

The convergence results for NR methods are relatively straight forward; if the form of the functional form f(.) is continuously differentiable over a convex set D containing the unique true solution,Y^{Sol}, where $f(Y^{Sol})$ in non-singular and $f(Y^{Sol,x})=0$ then there exists around Y^{Sol} an open set C such that

$$F^{s-1} = [\partial f/\partial y]_{y(s-1)}$$

converges from any starting values in the set of C. Furthermore if

$$\| F(Y)-F(Y^{Sol}) \| \leq d \| (Y-Y^{Sol}) \|$$

holds with d>0 the rate of convergence becomes:

$$\| Y^s - Y^{Sol} \| \leq \delta \| Y^{s-1} - Y^{Sol}) \|^{1+q}$$

Where s is the current iteration number. Thus convergence is quadratic (as opposed to linear as under First-Order systems) with q=1 and $\delta>0$. Indeed quadratic convergence rates is the norm if C is sufficiently small when

$$f_i(y,x) = 0$$

is twice differentiable around Y^{Sol}.

Thus convergence to Y^{Sol} is *guaranteed* once the iterates (or indeed any arbitrary starts) are within some open set C - without the need for damping or acceleration parameters.[6]

However despite these seemingly trivial convergence requirements traditional NR solutions are computational problematic because of the need to evaluate, invert and update F, the Jacobian (which is of the order of the model) at each iteration and that problem worsens if we have inappropriate first-guesses[7] or if the Jacobian is near singular around the solution. Therefore, unless the system can be decomposed or the algorithm takes advantage of the sparseness of the Jacobian the solution remains unambiguously inefficient for large systems. We now present a simple example of how such sparseness in the Jacobian manifests itself.

2.3 A Simple Example Of Sparse Jacobians.

Consider the following traditional closed-economy Income-Expenditure model:

Consumption Function.
$C_t = a*Y^d + b*C_{t-1}$
Disposable Income Identity.
$Y^d_t = Y_t - T_t$
National Income Identity
$Y_t = C_t + G_t + I_t$
Non-Linear Tax Yield
$T_t = (RY)_t$
Tax Rate
$R_t = c + d*Y_t$

We can build up the (first-iteration) Jacobian matrix, F_o :

$$\begin{matrix} 1 & -a & 0 & 0 & 0 \\ 0 & 1 & -1 & 1 & 0 \\ -1 & 0 & 1 & 0 & 0 \\ 0 & 0 & -R_o & 1 & -Y_o \\ 0 & 0 & -d & 0 & 1 \end{matrix}$$

where the subscript $_o$ indicates starting (or lagged) values.

Thus it can be seen that over 50% (13 out of 25) of the elements in the Jacobian are sparse (i.e., zero). Indeed sparseness is very much a feature of identified

---

[6] In practise damping is useful in the armoury of NR users and of the form:
$y^s = y^{s-1} - \alpha(F^{s-1})^{-1} f(y^{s-1},x)$, where $0<\alpha\leq 1$.

[7] i.e., $Y^{s-1} \notin c$.

macro-models (and certainly not just a construct of this particular example) since typically not all of the endogenous variables appear in every single equation in a system.

Here of course continual evaluation, inversion and updating of the Jacobian is negligible given its trivial dimensions (5*5) however this would not carry over to ,say, a 100+ equation model.

For Newton techniques to be tractable therefore invariably implies that programmers minimise their computational burden.

NR techniques can be simplified by for example rank updating schemes (which avoid the need for continual Jacobian inversion) or only re-evaluating the Jacobian at discreet pre-set points. In the last case the more the steps to re-evaluation the more we tend to traditional NR methods, the fewer (Limit$_{\to\infty}$) the more the initial Jacobian matrix is used throughout all subsequent calculations.[8]

Furthermore NR methods are often based on models re-ordered into recursive and simultaneous blocks because if the simultaneous block is small then computational savings are made (relative to the formerly ordered model) from solving the equation system on a recursive basis - assuming those blocks are themselves easy to solve.

However, NR algorithms which exploit the sparseness and block structure of the Jacobian matrix hold out perhaps the most promising means of accelerating NR methods. Although there is nothing new about such methods - see for example Duff(1977),Duff et al(1986),Press et al(1992) - they have become relatively more well known and more successful recently with the common implementation of a new breed of Stacked-Newton algorithms - see for example Laffargue (1990),Boucekkine(1995) and Juillard (1996).However we reserve a discussion of those techniques until section III which deals specifically with algorithms for solving models with lead variables.

2.4 A General Comparison Of FO and NR Methods.

FO methods require an explicit normalisation and in turn are sensitive to equation ordering; simulations may converge slowly or not at all (even when a well-defined stable solution exists), depending on whatever equation ordering call exists. Moreover, in case of convergence difficulties there are practical difficulties in arranging an alternative successful ordering.[9] Normalisation itself is also important : models can be written in any number of (algebraically equivalent) ways but may fail to solve depending on whatever normalisation is employed.

NR methods (which require no explicit normalisation) are sensitive to the choice of starting values and so, where $Y^{s-1} \notin C$, convergence (in non-linear models) is certain to fail. In most contemporary macro-models this does not - it should be admitted - present too much of a practical problem since invariably 'starts' are derived from

---

[8] This however will reduce the rate of convergence from Superlinear to linear ; a clear trade off therefore.

[9] Individual modellers will of course follow their own practises such as calling the least embedded equations first,solving for flows before stocks, solving the (uncovered interest parity) exchange rate equation first etc. However there exists a considerable literature on reordering techniques – for example Stewart(1962),Don and Gallo(1987),Hughes Hallett and Piscitelli (1998) and Hughes Hallett and Fisher (1990) investigate others.

lagged or steady state values.[10] However the overwhelming problem with Newton methods remains that - as they stand - they are rarely computationally feasible since the solution method requires evaluation, inversion and updating of the Jacobian, F, *at each iteration* : for models of ≥ 100 equations the computational burden involved is non-trivial. In addition, that problem worsens if we have inappropriate first-guesses or if the Jacobian is near singular around the solution. Therefore *workable* Newton methods require some decomposition of the Jacobian.

3. EXTENSIONS TO MODELS WITH 'RATIONAL' EXPECTATIONS (RE).

Normal First Order or Newton methods have to be changed to accommodate RE models otherwise they would treat the lead as exogenous in the same way as FT Type 1 solution.

3.1 First Order Techniques For RE Models.

For First-Order Techniques invariably the Fair-Taylor algorithm is used - we fix the expectational terms at their baseline values and solve the system as in the conventional case period by period with normal Gauss-Seidel iterative techniques (or indeed any solution method),[11] having done this 'Type 1' or 'inner loop' we then update the lead terms, for each variable in turn. We then return to the inner or Type 1 loop with the expectational terms updated and perform another iteration - until the tolerance between successive iteration on both loops respects the pre-set convergence criteria - which may be set differently across loops. We therefor have a two-part scheme - an inner or Type1 loop, which solves for the current, and lagged parts of the model with fixed expectations and an outer or Type 2 loop, which solves the model consistently. When the outer or Type 2 loop has converged the model has consistently solved subject to iteration tolerance.

To illustrate ,consider the model :

$$Y_t = BY_{t-1} + CY^e_{t+1|t} + U_t$$

which stacked over time yields:

---

[10] Although of course that in itself is no guarantee that these values are within C either.

[11] In these exercises we have chosen to solve the type 1 loop of the FT runs with Newton matrix inversion methods since in the authors' experience this has proved more robust at solving and developing this particular model. Solving the model with Gauss methods was examined in Poiro et al (1996) using different software (Fisher's (1990) SLIM) and a vintage of the model (MULTAQ) but with far less successful results in terms of generating convergent scenarios for the FO methods.

$$\begin{bmatrix} I & -C & 0 & 0 \\ -B & I & -C & 0 \\ 0 & -B & I & -C \\ 0 & 0 & -B & I \end{bmatrix} * \begin{bmatrix} Y_t \\ . \\ . \\ Y_T \end{bmatrix} = \begin{bmatrix} U_t \\ . \\ . \\ U_T \end{bmatrix} + \begin{bmatrix} B \\ 0 \\ 0 \\ 0 \end{bmatrix} * Y_0 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ C \end{bmatrix} * Y_{T+1}$$

which can be written more compactly as:

$$HY_t = b_t$$

with b_t containing predetermined variables such as lags, residuals, exogenous terms and perhaps forward terms if expectations are fixed. A solution only exists if H^{-1} is defined with the matrix H itself being unit block diagonal with upper and lower triangular sub-matrices of U^{s-1} and L^{s-1}. The lower triangle can by solved by any solution method keeping expectations fixed with the upper block triangle solved by setting expectational terms equal to (or as a weighted average of solved and baseline estimates of) their forward solution from the lower triangle.

Often modellers have sought to take advantage of this splitting of the solution procedure with the most popular method the so-called incomplete inner iterations method.[12] This may be explained as follows : since the solutions of the inner or Type 1 loop will be updated with every outer or Type 2 loop the practise choose is to avoid these extra and unnecessary calculations - we can therefore set the convergence criteria for the inner or Type 1 loop looser relative to the outer (Rational Expectations) loop. Alternatively we may require the Type 1 loop to have tighter or the same convergence criteria as the Type 2.[13] Alternatively rather than loosen convergence criteria (cc) we could vary the maximum number of iterations allowed over each loop.

3.2 Newton Techniques For RE Models

Fundamentally we have two approaches for solving RE models with Newton Techniques. First we simply use Newton for calculating the inner or type 1 loop (rather than First Order methods) and so yielding essentially the same type of analysis of section II.I,[14] the other is to use a single loop NR method in which we endogenous leads. In this paper we compare FT methods (with Newton on the inner) and a single-loop Newton method. We have found elsewhere (Poiro et al (1996) , Juillard et al (1998)) that first order methods have performed particularly poorly in the case of Multimod.[15]

As suggested operational NR techniques often require some decomposition of the Jacobian matrix to reduce the computational burden to manageable levels; for

---

[12] Fisher (1992) and Fisher et al (1986) discuss many more types of splitting - according to their own classification the method presented here is their (preferred) c method.

[13] The (TROLL) FT macro that we use was developed by Faust, Tyron and Gagnon at the Federal Reserve Board Of Governors and interestingly would seem to have CC_Type1 < CC_Type2 as its default.

[14] Although the convergence requirements follow that given in section II.II rather than II.1 .

[15] We have also found that another Newton based one loop procedure - discussed in Armstrong et al(1995) - performed badly and therefore excluded it from our analysis.

example in the MULTAR vintage of MULTIMOD with 466 equations and ,say, 120 simulation periods we would have a Jacobian of dimension 56,852 by 56,852 (i.e.,[n*(T+2)]*[n*(T+2)].[16] However as is shown in Laffargue(1990) the structure of this matrix is such that its triangularization can be handled recursively and so there is no need to store the entire Jacobian at any one time; the matrix stored need only in fact be of order 55,920 by 112 (ie,n*T by TNLV - where TNLV = Total Number Of Lead Variables).

In this paper we concentrate on an extension of such techniques based on Laffargue (1990), Boucekkine (1995) and Juillard (1996). In accordance with modern programming parlance we shall call this algorithm *NEWSTACK* although it has also been variously called 'LBJ' - after the aforementioned authors - in Juillard et al(1998) and the 'Stacked-Time Simulator' in Hollinger (1996).

3.3 A New NR Method For RE Models: NewStack

We can write the equations of a model as

$$f(z_t) = \begin{bmatrix} g_1(y_{t-1}, y_t, y_{t+1}, x_t, \theta) \\ \vdots \\ g_n(y_{t-1}, y_{t+1}, x_t, \theta \end{bmatrix}$$

for t-1,2,..,T and $z_t=[y'_{t-1},y'_t,y'_{t+1}]'$. That is to say as a stacked non-normalised equation set over time.

Solving all periods simultaneously we can build up the vector of endogenous variables:

$$Y' = [y'_0,y'_1,\ldots,y'_T,y'_{T+1}]'.$$

We know the initial and terminal points:

$$f_o = (y_o - y_o^*) = f_{T+1} = (y_{T+1} - y_{T+1}^*) = 0$$

and so the entire system forms a (T+2)*n equation system:

---

[16] The problem here is essentially a two-point boundary problem accommodating initial and terminal conditions hence the 2 in 'T+2'.

$$F(Y) = \begin{bmatrix} f_0(y_0) \\ f_1(z_1) \\ . \\ . \\ . \\ f_T(Z_T) \\ f_{T_{+1}}(Y_{T_{+1}}) \end{bmatrix} = 0$$

Now recall the general NR structure:

$$y^t = y^{t-1} - (F^{t-1})^{-1} f(y^{t-1},x)$$

or

$$\Delta y^t = - (F^{t-1})^{-1} f(y^{t-1},x)$$

$$(F^{t-1}) \Delta y^t = - f(y^{t-1},x)$$

which here in full matrix form becomes:

$$\begin{bmatrix} I & 0 & 0 & 0 & 0 & 0 & 0 \\ L_1 & C_1 & F_1 & 0 & 0 & 0 & 0 \\ 0 & . & . & . & 0 & 0 & 0 \\ 0 & 0 & L_t & C_t & F_t & 0 & 0 \\ 0 & 0 & 0 & . & . & . & 0 \\ 0 & 0 & 0 & 0 & L_T & C_T & F_T \\ 0 & 0 & 0 & 0 & 0 & 0 & I \end{bmatrix} * \Delta Y = - \begin{bmatrix} 0 \\ f_1(z_1) \\ . \\ f_t(z_t) \\ . \\ f_T(z_T) \\ 0 \end{bmatrix}$$

where L,C and F are the partial Jacobian for lagged, contemporaneous and forward variables ,i.e.,

$F_t = \partial f_t(z_t)/\partial y_{t+1}$
etc.

The approach of Newstack is to remove elements below or above the main diagonal (here we remove below but the solution is invariant to whether you end up

with an upper or lower block Jacobian) and so the solution can proceed recursively either backwards or forwards.

Consider the first period solution :

$$L_1 \Delta Y_o + C_1 \Delta Y_1 + F_1 \Delta Y_2 = -f_1(z_1) \qquad (3)$$

Given $\Delta y_o=0$ this reduces to :

$$\Delta Y_1 + M_1 \Delta Y_2 = d_1$$

where,$M_1=(C_1)^{-1}*F_1$ and $d_1=-(C_1)^{-1}*f_1(z_1)$

For the second period:

$$L_t \Delta Y_1 + C_t \Delta Y_t + F_t \Delta Y_{t+1} = -f_t(z_t)$$

ΔY_1 can be retrieved from (3) and substitution Y_{t-1} for Y_1 yields:

$$\Delta Y_t + M_t \Delta Y_{t+1} = D_t$$

where $M_t=(C_t-L_tM_{t-1})^{-1}*F_t$ and $d_t=-(C_t - L_t M_{t-1})^{-1}*[f_1(z_1) + L_td_{t-1}]$

We can do this for all subsequent periods but clearly this M and d pattern is emerging. Thus the system can be recomposed as

$$\begin{bmatrix} I & 0 & 0 & 0 & 0 \\ 0 & I & M_1 & 0 & 0 \\ 0 & 0 & . & . & 0 \\ 0 & 0 & 0 & I & M_T \\ 0 & 0 & 0 & 0 & I \end{bmatrix} * \Delta Y = \begin{bmatrix} 0 \\ d_1 \\ . \\ d_T \\ 0 \end{bmatrix}$$

And the value of ΔY can be retrieved through backward substitution:

$$\Delta Y_t = d_t - M_t \Delta Y_{t+1}$$

And so in this approach it is only the M_t and d_t block, for t=1,2,..,T which require storage. And further storage reduction can be achieved by taking into account the sparse columns of the Jacobians using conventional sparse matrix techniques - see for example Duff et al(1986),Press et al(1992).

4. THE SIMULATION FRAMEWORK : THE SIMULATIONS UNDERTAKEN

In these exercises we run the following deterministic simulations on the US country :

(A) A permanent increased in Government Expenditure of 5% of real baseline GDP.

(B) A permanent 5% decreases in the Debt target.

(C) A permanent 5% increases in the monetary target.

(D) A permanent 50% increases in World Oil Production.[17]

Whilst these scenarios are by no means exhaustive they are certainly representative; for a wider selection see Poiro et al (1996). Furthermore to test for robustness and any non-linearities involved (in the timing and accuracy dimension) we also doubled both the size of these shocks (from 5% to 10% etc) and the simulation horizon (from T to 2T).Specifically, we perform these shocks for the US economy.[18]

Finally in these exercise - as in Armstrong et al (1995) - we set a common convergence setting of:[19]

$$\max_j \left| (y^s - y^{s-1})/(y^{s-1}+5) \right| < \delta.$$

Where δ, the convergence tolerance, is changed at various junctures to accommodate incomplete inner iterations. Such a rule may be justified as a mixture of relative and absolute convergence criteria since the '5' in the denominator prevents division by zero for variables which may take values around zero (e.g., the trade balance) - it makes the convergence criteria approximate an absolute one for small iterated values of y and a relative one for larger values.

Specifically (and to accommodate incomplete-inner-iterations) we follow the following codes for the FT simulations : we start off with three types (A,B,C) where we set a common Type 2 convergence criteria of 0.001 which matches that of Newstack. As we move from A to B however we progressively tighten the type 1 convergence criteria from ten times looser (0.01) to ten times tighter (0.0001) the type 2 one. Thereafter (D to F) we perform the same pattern although with tighter type 2 convergence criteria at 0.0001. The reason for the latter scenario is our prior knowledge that Newstack is

---

[17] Note that this is a shock to oil production rather than oil prices - the former is exogenous whilst the latter is endogenous. For Oil Price Shocks see Juillard et al(1998).

[18] All in all this involves running 32 simulations. These jobs were performed on batch mode overnight on a Pentium PC with 32mb of Ram and a 200Mhz clock at Strathclyde University. Any further information on these simulation can be directed to *p.mcadam@ukc.ac.uk*. For all the simulations we used the TROLL simulation software,Hollinger and Spivakovsky(1996).

[19] In TROLL parlance:CONOPT STOP 200 CONCR 0.001 GAMMA 5;.