# An Application of Nonstationary Iterative Methods for Solving a Multi-Country Model with Rational Expectations*

Manfred Gilli
Department of Econometrics
University of Geneva

Giorgio Pauletto
Department of Econometrics
University of Geneva

**Abstract**

In this paper we present an implementation of a Newton method based on iterative Krylov subspace methods such as GMRES, QMR and BiCGSTAB for solving large nonlinear macroeconometric models. These methods are tested for the solution of the model MULTIMOD and the computational costs of the different techniques are compared together with a sparse direct method.

## 1 Introduction

Traditionally in the practice of solving large macroeconometric models two kinds of solution algorithms have been used. The most popular one are probably first-order iterative techniques and related methods like Gauss-Seidel. One evident reason for this is their ease of implementation. A second reason is that their computational complexity is in general quite low. This is due

*Paper presented at the *Second International Conference on Computing in Economics and Finance*, Geneva, Switzerland, 26–28 June 1996. This paper is available at URL `http://www.unige.ch/ses/metri/pauletto/ce96.ps`

in particular to the fact that Gauss-Seidel exploits the sparse structure of the system of equations. The convergence of these methods depends on the particular quantification of the equations and their ordering. Convergence is not guaranteed and its speed is linear.

Newton-type methods constitute a second group of techniques commonly used to solve models. These methods use the information about derivatives of the equations. The major advantage is then a quadratic convergence and the fact that it is not necessary to normalize the equations and that the ordering does not influence the convergence rate. The computational cost comprises the evaluation of the derivatives forming the Jacobian and the solution of the linear system. If the linear system is solved using a classical direct method based on LU or QR decomposition the complexity of the whole method is $O(n^3)$. This promises interesting savings in computations if the size $n$ can be reduced. A common technique consists then in applying the Newton method only to a subset of equations, for instance the equations formed by the spike variables.

This leads to a hybrid method, i.e. a first-order iterative method where a subsystem of equations only is solved with a Newton method. The hybrid method can also be viewed as a block method, where the first block constitutes a recursive system and the second block (in general much smaller) is solved by a Newton method.

However such a method brings back the problem of convergence for the outer loop. Moreover, for macroeconometric models, it happens that the block of spike variables is also in most cases recursive which then results in carrying out unnecessary computations (see Gilli, Pauletto and Garbely [15] p. 150).

Thus the hybrid or block method tries to take advantage from both, the sparse structure of the system under consideration and the desirable convergence properties of Newton-type algorithms. However, as explained above, this approach relapses into the convergence problem in the framework of a block method.

This suggests that the sparsity should be exploited when solving the linear system in the Newton method. This can be achieved by using sparse direct solvers. Nonstationary methods have been more recently developed. They use information that changes from iteration to iteration contrarily to the stationary methods such as Gauss-Seidel. These methods are computationally attractive as the operations involved can be easily executed on sparse matrices and also require few storage. They also generally show a better convergence speed than stationary iterative methods.

In this paper we present a Newton method using nonstationary iterative methods for solving and simulating large macroeconometric models. First we present the framework of the problem and recall the principles of nonlinear first-order techniques and Newton-like methods for solving nonlinear systems of equations. In a second section nonstationary solution algorithms for linear systems are presented. Finally, the method proposed is illustrated by solving the forward looking model MULTIMOD.

# 2 The Problem

The econometric models we consider here for solution are represented by a system of $n$ linear and nonlinear equations

$$h(y, z) = 0 \tag{1}$$

where $h : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ is differentiable in the neighborhood of the solution $y^* \in \mathbb{R}^n$ and $z \in \mathbb{R}^m$ are the lagged and the exogenous variables. In practice, the Jacobian matrix $\partial h/\partial y'$ of an econometric model can often be put into a blockrecursive form, as shown in Figure 1, where the dark shadings indicate interdependent blocks and the light shadings the existence of nonzero elements. The solution of the model then concerns a sequence of recursive

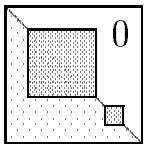


Figure 1: Blockrecursive pattern of a Jacobian matrix.

equations and interdependent submodels. Clearly only the solution of the latter constitutes a problem and will be discussed in the paper.

Let us briefly recall the principles of the two main techniques for the solution of such systems. First-order iterative techniques such as Gauss-Seidel and Jacobi require the normalization of the model, i.e. rewriting the system (1) in the following way

$$y_i = g_i(y_1, \ldots, y_{i-1}, y_{i+1}, \ldots, y_n, z) \;, \quad i = 1, \ldots, n \;. \tag{2}$$

In the Gauss-Seidel case, the generic iteration $k$ can be written

$$y_i^{k+1} = g_i(y_1^{k+1}, \ldots, y_{i-1}^{k+1}, y_{i+1}^k, \ldots, y_n^k, z) \ , \quad i = 1, \ldots, n \ . \tag{3}$$

where the $i-1$ updated components of the vector $y^{k+1}$ are used as soon as they are available. In the Jacobi case the components of the vector $y$ in the right-hand side are updated at once at the end of each iteration. First-order iterative algorithms can then be summarized to the three statements given in Algorithm 1.

**Algorithm 1** *Nonlinear First-order Method*

**while** ( not_converged )
1. $y^0 = y$
2. Evaluate all equations
3. not_converged $=$ any$(|(y - y^0)/y^0| > \eta)$

**end**

The difference between Jacobi and Gauss-Seidel appears in statement 2. Jacobi uses different arrays for $y$ and $y^0$, i.e. $y_i = g_i(y_1^0, \ldots, y_{i-1}^0, y_{i+1}^0, \ldots, y_n^0, z)$, whereas Gauss-Seidel overwrites $y^0$ with the computed values for $y$ and therefore the same array is used in $y_i = g_i(y_1, \ldots, y_{i-1}, y_{i+1}^0, \ldots, y_n^0, z)$. Statement 3 specifies the termination criterion for the iterations.

When solving the system with Newton methods, the general step $k$ in the iterative process can be written as

$$y^{k+1} = y^k - \underbrace{\left[\frac{\partial h(y^k, z)}{\partial y'}\right]^{-1}}_{J^k} h(y^k, z) \quad . \tag{4}$$

Two types of problems occur at each iteration (4): first the evaluation of the Jacobian matrix $J^k = h'(y^k, z)$ and the function $h(y^k, z)$ and second the solution of the linear system $(J^k)^{-1}h(y^k, z)$. The six statements given in Algorithm 2 schematize the Newton algorithm.

**Algorithm 2** *Newton Method*

**while** ( not_converged )
1. $y^0 = y$
2. **evaluate** $b = -h(y^0, z)$
3. **evaluate** $J = \partial h(y^0, z)/\partial y'$
4. **solve** $J\ s = b$
5. $y = y^0 + s$
6. not_converged $= \text{any}(|s/y^0| > \eta)$

**end**

Let us compare the computational complexity of the two algorithms for a single iteration. Statements 1 and 2 are of the same complexity for both algorithms and statement 3 in the first-order iterative algorithm has the same complexity as statement 5 and 6 together in the Newton algorithm.

The computational complexity of the Newton algorithm for one iteration is therefore exactly superior by the computations required to execute statements 3 and 4 which is the numerical evaluation of the Jacobian matrix and the solution of a linear system. This seems an overwhelming disadvantage for this method.

Looking however closer to the particular problem of solving macroeconometric models we may discover that despite this seemingly big disadvantage Newton-like methods are very likely to have an overall computational complexity which is inferior. Indeed, macroeconometric models are used for policy simulations, sometimes for forecasting which can be done in a deterministic way or better by means of stochastic simulations. This implies that one has to solve many times a system of equations the logical structure of which does not change. As a consequence some intermediate results of the computations in the solution algorithm remain identical for all simulations. These results are therefore computed only once and their computational cost can be considered as an overhead.

Much of the computations in statement 3 of the Newton algorithm have to be executed only once: this concerns the symbolic derivation which define the elements in the Jacobian matrix and the evaluation of those elements which are constants (in the Jacobian matrix).

In statement 4 we solve a sparse linear system and savings in the computations which have to be repeated for every simulation can be made by performing only an evaluation of already determined expressions for solving the

system.

Iterative methods form an important class of solution techniques for solving large systems of linear equations. They can be an interesting alternative to direct methods because they take into account the sparsity of the system. Iterative methods may be divided into two classes: stationary and nonstationary. The former rely on invariant information from an iteration to another, whereas the latter modify their search by using the results of previous iterations.

The computational complexity of a Newton method combined with an efficient solution for the linear system is in general much smaller than the work needed to iterate a larger number of times with a nonlinear first-order iterative method which moreover may fail to converge much easier than a Newton method.

# 3 Solution of the Linear System

Although efficient nonstationary iterative methods for the solution of large and sparse linear systems are by now available, these methods are not widely used in economic modeling.

Such methods have been more recently developed and use information that changes from iteration to iteration contrarily to the stationary methods such as Jacobi or Gauss-Seidel. These methods are computationally attractive as the operations involved can be easily executed on sparse matrices and also require few storage. They also generally show a better convergence speed than stationary iterative methods. Presentations of nonstationary iterative methods can be found for instance in Freund, Golub and Nachtigal [12], Barrett et al. [4] and Axelsson [3].

We first have to present some algorithms that solve particular systems, such as symmetric positive definite ones, from which were derived the nonstationary iterative methods for solving the general linear systems we are interested in.

## 3.1 Conjugate Gradient

The first and perhaps best known of the nonstationary methods is the *Conjugate Gradient* (CG) method proposed by Hestenes and Stiefel in [19]. This technique solves symmetric positive definite systems $Ax = b$ by using only

matrix-vector products, inner products and vector updates. The method may also be interpreted as arising from the minimization of the quadratic function $q(x) = \frac{1}{2}x'Ax - x'b$ where $A$ is the symmetric positive definite matrix and $b$ the right-hand side of the system. As the first order conditions for the minimization of $q(x)$ give the original system, the two approaches are equivalent.

The idea of the CG method is to update the iterates $x^{(i)}$ in the direction $p^{(i)}$ and to compute the residuals $r^{(i)} = b - Ax^{(i)}$ in such a way to ensure that we achieve the largest decrease in terms of the objective function $q$ and furthermore that the direction vectors $p^{(i)}$ are $A$-orthogonal.

The largest decrease in $q$ at $x^{(0)}$ is obtained by choosing an update in the direction $-\nabla q(x^{(0)}) = b - Ax^{(0)}$. We see that the direction of maximum decrease is the residual of $x^{(0)}$ defined by $r^{(0)} = b - Ax^{(0)}$. We can look for the optimum step length in the direction $r^{(0)}$ by solving the line search problem

$$\min_{\alpha} q(x^{(0)} + \alpha r^{(0)}) \ .$$

As the derivative with respect to $\alpha$ is

$$\begin{aligned} \nabla_{\alpha} q(x^{(0)} + \alpha r^{(0)}) &= x^{(0)\prime}Ar^{(0)} + \alpha r^{(0)}Ar^{(0)} - b'r^{(0)} \\ &= (x^{(0)\prime}A - b')r^{(0)} + \alpha r^{(0)\prime}Ar^{(0)} \\ &= r^{(0)\prime}r^{(0)} + \alpha r^{(0)\prime}Ar^{(0)} \ , \end{aligned}$$

we have that the optimal $\alpha$ is

$$\alpha_0 = -\frac{r^{(0)\prime}r^{(0)}}{r^{(0)\prime}Ar^{(0)}} \ .$$

The method described up to now is just a steepest descent with exact line search on $q$. To avoid the convergence problems which are likely to arise with this technique, it is further imposed that the update directions $p^{(i)}$ be $A$-orthogonal (or conjugate with respect to $A$). In other words that we have

$$p^{(i)\prime}Ap^{(j)} = 0 \qquad i \neq j \ . \tag{5}$$

It is therefore natural to choose a direction $p^{(i)}$ that is closest to $r^{(i-1)}$ and satisfies equation (5). It is possible to show that one can find explicit formulas for such a $p^{(i)}$, see e.g. Golub and Van Loan [17, pp. 520–523]. These solutions can be expressed in a computationally efficient way involving only one matrix-vector multiplication per iteration.

The CG method can be formalized as follows.

**Algorithm 3** *Conjugate Gradient*

Compute $r^{(0)} = b - Ax^{(0)}$ for some initial guess $x^{(0)}$
**for** $i = 1, 2, \ldots$ **until** convergence
    $\rho_{i-1} = r^{(i-1)\prime} r^{(i-1)}$
    **if** $i = 1$ **then**
        $p^{(1)} = r^{(0)}$
    **else**
        $\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$
        $p^{(i)} = r^{(i-1)} + \beta_{i-1} p^{(i-1)}$
    **end**
    $q^{(i)} = Ap^{(i)}$
    $\alpha_i = \rho_{i-1}/(p^{(i)\prime} q^{(i)})$
    $x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$
    $r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$
**end**

In the conjugate gradient method, the $i$-th iterate $x^{(i)}$ can be shown to be the vector minimizing $(x^{(i)} - x^*)' A (x^{(i)} - x^*)$, where $Ax^* = b$, among all $x^{(i)}$ in the affine subspace $x^{(0)} + \text{span}\{r^{(0)}, Ar^{(0)}, \ldots, A^{m-1} r^{(0)}\}$. This subspace is called the *Krylov subspace* associated to $A$ and $r^{(0)}$.

### 3.1.1 Convergence of the CG Method

In exact arithmetic the CG method yields the solution in at most $n$ iterations, see Luenberger [21, p. 248, Theorem 2]. In particular we have the following relation for the error in the $k$-th CG iteration

$$\|x^{(k)} - x^*\|_2 \leq 2\sqrt{\kappa} \left( \frac{\sqrt{\kappa} - 1}{\sqrt{\kappa} + 1} \right)^k \|x^{(0)} - x^*\|_2 \ ,$$

where $\kappa = \kappa_2(A)$, the condition number of $A$ in the two norm. However in finite precision and with a large $\kappa$ the method may fail to converge.

## 3.2 Preconditioning

The convergence speed of the CG method is as just explained linked to the condition number of the matrix $A$. To improve the convergence speed of the CG-type methods, the matrix $A$ is often *preconditioned*, that is transformed into $\hat{A} = SAS'$, where $S$ is a nonsingular matrix. The system solved is then

$\hat{A}\hat{x} = \hat{b}$ where $\hat{x} = (S')^{-1}x$ and $\hat{b} = Sb$. The matrix $S$ is chosen so that the condition number of matrix $\hat{A}$ is smaller than the condition number of the original matrix $A$ and, hence, speeds up the convergence.

In order to avoid the explicit computation of $\hat{A}$ as well as not to destroy the sparsity pattern of $A$, the methods are usually formalized to use directly the original matrix $A$. We can build a *preconditioner*

$$M = (S'S)^{-1}$$

and apply the preconditioning step by solving the system $M\tilde{r} = r$. Since $\kappa_2(S'\hat{A}(S')^{-1}) = \kappa_2(S'SA) = \kappa_2(M^{-1}A)$, one actually does not form $M$ from $S$ but rather directly chooses a matrix $M$. The choice of $M$ is constrained to be a symmetric positive definite matrix.

The preconditioned version of the CG is described in the following algorithm.

**Algorithm 4** *Preconditioned Conjugate Gradient*

Compute $r^{(0)} = b - Ax^{(0)}$ for some initial guess $x^{(0)}$
**for** $i = 1, 2, \ldots$ **until** convergence
    Solve $M\tilde{r}^{(i-1)} = r^{(i-1)}$
    $\rho_{i-1} = r^{(i-1)\prime}\tilde{r}^{(i-1)}$
    **if** $i = 1$ **then**
        $p^{(1)} = \tilde{r}^{(0)}$
    **else**
        $\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$
        $p^{(i)} = \tilde{r}^{(i-1)} + \beta_{i-1}p^{(i-1)}$
    **end**
    $q^{(i)} = Ap^{(i)}$
    $\alpha_i = \rho_{i-1}/(p^{(i)\prime}q^{(i)})$
    $x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$
    $r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$
**end**

As the preconditioning speeds up the convergence, the question of how to choose a good preconditioner naturally arises. There are two conflicting goals in the choice of $M$. First, $M$ should reduce the condition number of the system solved as much as possible. To achieve this we would like to pick an $M$ as "close" as possible to matrix $A$. Second, since the system $M\tilde{r} = r$ has to be solved at each iteration of the algorithm, this system should be as "easy" to solve as possible. Clearly, the preconditioner will be chosen between the two extreme cases $M = A$ and $M = I$. When $M = I$, one

obtains the unpreconditioned version of the method and when $M = A$ the complete system is solved in the preconditioning step. A possibility is to take $M = \mathrm{diag}(a_{11}, \ldots, a_{nn})$. This is not useful if the system is normalized, as it is sometimes the case for macroeconometric systems.

Other preconditioning methods do not construct explicitly $M$. Some authors, for instance Dubois, Greenbaum and Rodrigue [8] and Adams [1], suggest to take a given number of steps of an iterative method such as Jacobi. We can note that taking one step of Jacobi amounts to do a diagonal scaling $M = \mathrm{diag}(a_{11}, \ldots, a_{nn})$ as mentioned before.

Another common approach is to perform an *incomplete LU factorization* (ILU) of the matrix $A$. This method is similar to the LU factorization except that it respects the pattern of nonzero elements of $A$ in the lower triangular part of $L$ and the upper triangular part of $U$. In other terms we apply the following algorithm:

**Algorithm 5** *Incomplete LU factorization*

```
Set L = I_n        The identity matrix of order n
for k = 1, ..., n
    for i = k + 1, ..., n
        if a_ki = 0 then        Respect the sparsity pattern of A
            l_ik = 0
        else
            l_ik = a_ik / a_kk
            for j = k + 1, ..., n
                if a_ij ≠ 0 then        Respect the sparsity pattern of A
                    a_ij = a_ij - l_ik a_kj        Gaussian elimination
                end
            end
        end
    end
end
Set U = upper triangular part of A
```

This factorization can be written as $A = LU + R$ where $R$ is a matrix containing the elements that would "fill-in" $L$ and $U$ and is not actually computed. The approximate system $LU\tilde{r} = r$ is then solved using forward and backward substitution in the preconditioning step of the nonstationary method used.

A more detailed analysis of preconditioning and other incomplete factoriza-

tions can be found in Axelsson [3].

## 3.3 Conjugate Gradient Normal Equations

In order to deal with nonsymmetric systems it is necessary either to convert the original system into a symmetric positive definite equivalent one, or to generalize the CG method. The next sections discuss such possibilities.

The first approach, and perhaps the easiest, is to transform $Ax = b$ into a symmetric positive definite system by multiplying the original system by $A'$. As $A$ is assumed to be nonsingular, $A'A$ is symmetric positive definite and the CG algorithm can be applied to $A'Ax = A'b$. This method is known as the *Conjugate Gradient Normal Equation* (CGNE) method.

A somewhat similar approach is to solve $AA'y = b$ by the CG method and then to compute $x = A'y$. The difference of the two approaches is discussed by Golub and Ortega [16, pp. 397ff].

Besides the computation of the matrix-matrix and matrix-vector products, these methods have the disadvantage of increasing the condition number of the system solved since $\kappa_2(A'A) = (\kappa_2(A))^2$. This in turn increases the number of iterations of the method, see Barrett et al. [4, p. 16] and Golub and Van Loan [17]. However since the transformation and the coding are easy to implement the method might be appealing in certain circumstances.

## 3.4 Generalized Minimal Residual

Paige and Saunders [23] proposed a variant of the CG method that minimizes the residual $r = b - Ax$ in the 2-norm. It only requires the system to be symmetric and not positive definite. It can also be extended to unsymmetric systems if some more information is kept from step to step. This method is called GMRES, which stands for Generalized Minimal Residual, and was introduced by Saad and Schultz [24].

The difficult part is not to loose the orthogonality property of the direction vectors $p^{(i)}$. To achieve this goal, all previously generated vectors have to be kept in order to build a set of orthogonal directions, for instance by a modified Gram-Schmidt orthogonalization process.

However, this method necessitates the storage and the computation of an increasing amount of information. Thus, in practice, the algorithm is very limited due to its prohibitive cost.

To overcome these difficulties, the method may be "restarted" after a chosen number of iterations $m$; the information is erased and the current intermediate results are used as a new starting point. The choice of $m$ is of critical importance for the restarted version of the method, usually referred as GMRES($m$).

The pseudo-code for this method is given hereafter.

**Algorithm 6** *Preconditioned GMRES($m$)*

Choose an initial guess $x^{(0)}$ and initialize an $(m+1) \times m$ matrix $\bar{H}_m$ to $h_{ij} = 0$
**for** $k = 1, 2, \ldots$ **until** convergence
  Solve for $r^{(k-1)}$  $Mr^{(k-1)} = b - Ax^{(k-1)}$
  $\beta = \|r^{(k-1)}\|_2$ ;  $v^{(1)} = r^{(k-1)}/\beta$ ;  $q = m$
  **for** $j = 1, \ldots, m$
    Solve for $w$  $Mw = Av^{(j)}$
    **for** $i = 1, \ldots, j$  *Orthonormal basis by modified Gram-Schmidt*
      $h_{ij} = w' v^{(i)}$
      $w = w - h_{ij} v^{(i)}$
    **end**
    $h_{j+1,j} = \|w\|_2$
    **if** $h_{j+1,j}$ is sufficiently small **then**
      $q = j$
      **exit** from loop on $j$
    **end**
    $v^{(j+1)} = w / h_{j+1,j}$
  **end**
  $V_m = [v^{(1)} \ \ldots \ v^{(q)}]$
  $y_m = \mathrm{argmin}_y \|\beta e_1 - \bar{H}_m y\|_2$  *Use the method given below to compute $y_m$*
  $x^{(k)} = x^{(k-1)} + V_m y_m$  *Update the approximate solution*
**end**

To solve the least-squares problem involving the upper Hessenberg matrix $\bar{H}_m$, apply Givens rotations to triangularize $\bar{H}_m$
$d = \beta e_1$  $e_1$ *is* $[1\ 0 \ldots 0]'$
**for** $i = 1, \ldots, q$  *Compute the sine and cosine values of the rotation*
  **if** $h_{ii} = 0$ **then**
    $c = 1$ ;  $s = 0$
  else
    **if** $|h_{i+1,i}| > |h_{ii}|$ **then**
      $t = -h_{ii}/h_{i+1,i}$ ;  $s = 1/\sqrt{1+t^2}$ ;  $c = s\,t$
    **else**
      $t = -h_{i+1,i}/h_{ii}$ ;  $c = 1/\sqrt{1+t^2}$ ;  $s = c\,t$
    **end**
  **end**
  $t = c\,d_i$ ;  $d_{i+1} = -s\,d_i$ ;  $d_i = t$
  $h_{ij} = c\,h_{ij} - s\,h_{i+1,j}$ ;  $h_{i+1,j} = 0$
  **for** $j = i+1, \ldots, m$  *Apply rotation to zero the subdiagonal of $\bar{H}_m$*
    $t_1 = h_{ij}$ ;  $t_2 = h_{i+1,j}$
    $h_{ij} = c\,t_1 - s\,t_2$ ;  $h_{i+1,j} = s\,t_1 + c\,t_2$
  **end**
**end**
Solve the triangular system $\bar{H}_m y_m = d$ by back substitution.

Another issue with GMRES is the use of the modified Gram-Schmidt method

which is fast but not very reliable, see Golub and Van Loan [17, p. 219]. For ill-conditioned systems, a Householder orthogonalization process is certainly a better alternative, even if it leads to an increase in the complexity of the algorithm.

### 3.4.1 Convergence of GMRES

The convergence properties of GMRES($m$) are given in the original paper by Saad and Schultz [24] introducing the method. A recent result by Strikwerda and Stodder [25] gives a necessary and sufficient condition for GMRES($m$) to converge.

**Theorem 1** *A necessary and sufficient condition for GMRES($m$) to converge is that the set of vectors*

$$\mathcal{V}_m = \{v | v' A^j v = 0 \quad for \quad 1 \leq j \leq m\}$$

*contains only the vector 0.*

In particular it follows that for a symmetric or skew-symmetric matrix $A$, GMRES(2) converges.

Another important result stated in [25] is that if GMRES($m$) converges, it does so with a geometric rate of convergence:

**Theorem 2** *If $r^{(k)}$ is the residual after $k$ steps of GMRES($m$), then*

$$\|r^{(k)}\|_2^2 \leq (1 - \rho_m)^k \|r^{(0)}\|_2^2$$

*where*

$$\rho_m = \min_{\|v\|=1} \left( \frac{\sum_{j=1}^{m} (v^{(1)\prime} A v^{(j)})^2}{\sum_{j=1}^{m} \|A v^{(j)}\|_2^2} \right)$$

*and the vectors $v^{(j)}$ are the unit vectors generated by GMRES($m$).*

Similar conditions and rate of convergence estimate are also given for the preconditioned version of GMRES($m$).

## 3.5 BiConjugate Gradient Method

The BiConjugate Gradient method (BiCG) takes a different approach based upon generating two mutually orthogonal sequences of residual vectors $\{\tilde{r}^{(i)}\}$

and $\{r^{(j)}\}$ and $A$-orthogonal sequences of direction vectors $\{\tilde{p}^{(i)}\}$ and $\{p^{(j)}\}$. The interpretation in terms of the minimization of the residuals $r^{(i)}$ is lost. The updates for the residuals and for the direction vectors are similar to those of the CG method, but are performed not only using $A$ but also $A'$. The scalars $\alpha_i$ and $\beta_i$ ensure the bi-orthogonality conditions $\tilde{r}^{(i)}r^{(j)} = \tilde{p}^{(i)}Ap^{(j)} = 0$ if $i \neq j$.

The algorithm for the Preconditioned BiConjugate Gradient method is given hereafter.

**Algorithm 7** *Preconditioned BiConjugate Gradient*

Compute $r^{(0)} = b - Ax^{(0)}$ for some initial guess $x^{(0)}$
Set $\tilde{r}^{(0)} = r^{(0)}$
**for** $i = 1, 2, \ldots$ **until** convergence
    Solve $Mz^{(i-1)} = r^{(i-1)}$
    Solve $M'\tilde{z}^{(i-1)} = \tilde{r}^{(i-1)}$
    $\rho_{i-1} = z^{(i-1)\prime}\tilde{r}^{(i-1)}$
    **if** $\rho_{i-1} = 0$ **then** the method fails
    if $i = 1$ **then**
        $p^{(i)} = z^{(i-1)}$
        $\tilde{p}^{(i)} = \tilde{z}^{(i-1)}$
    **else**
        $\beta_{i-1} = \rho_{i-1}/\rho_{i-2}$
        $p^{(i)} = z^{(i-1)} + \beta_{i-1}p^{(i-1)}$
        $\tilde{p}^{(i)} = \tilde{z}^{(i-1)} + \beta_{i-1}\tilde{p}^{(i-1)}$
    **end**
    $q^{(i)} = Ap^{(i)}$
    $\tilde{q}^{(i)} = A'\tilde{p}^{(i)}$
    $\alpha_i = \rho_{i-1}/(\tilde{p}^{(i)\prime}q^{(i)})$
    $x^{(i)} = x^{(i-1)} + \alpha_i p^{(i)}$
    $r^{(i)} = r^{(i-1)} - \alpha_i q^{(i)}$
    $\tilde{r}^{(i)} = \tilde{r}^{(i-1)} - \alpha_i \tilde{q}^{(i)}$
**end**

The disadvantages of the method are the potential erratic behavior of the norm of the residuals $r_i$ and unstable behavior if $\rho_i$ is very small, i.e. the vectors $r^{(i)}$ and $\tilde{r}^{(i)}$ are nearly orthogonal. Another potential breakdown situation is when $\tilde{p}^{(i)\prime}q^{(i)}$ is zero or close to zero.

### 3.5.1 Convergence of BiCG

The convergence of BiCG may be irregular, but when the norm of the residual is significantly reduced, the method is expected to be comparable to GMRES. The breakdown cases may be avoided by sophisticated strategies see Barrett et al. [4] and references therein. Few other convergence results are known for this method.

## 3.6 BiConjugate Gradient Stabilized Method

A version of the BiCG method which tries to "smooth" the convergence was introduced by van der Vorst [26]. This more sophisticated method is called *BiConjugate Gradient Stabilized* method (BiCGSTAB) and its algorithm is formalized as follows.

**Algorithm 8** *BiConjugate Gradient Stabilized*

Compute $r^{(0)} = b - Ax^{(0)}$ for some initial guess $x^{(0)}$
Set $\tilde{r} = r^{(0)}$
**for** $i = 1, 2, \ldots$ **until** convergence
    $\rho_{i-1} = \tilde{r}' r^{(i-1)}$
    **if** $\rho_{i-1} = 0$ **then** the method fails
    **if** $i = 1$ **then**
        $p^{(i)} = r^{(i-1)}$
    else
        $\beta_{i-1} = (\rho_{i-1}/\rho_{i-2})(\alpha_{i-1}/w_{i-1})$
        $p^{(i)} = r^{(i-1)} + \beta_{i-1}(p^{(i-1)} - w_{i-1}v^{(i-1)})$
    **end**
    Solve $M\hat{p} = p^{(i)}$
    $v^{(i)} = A\hat{p}$
    $\alpha_i = \rho_{i-1}/\tilde{r}' v^{(i)}$
    $s = r^{(i-1)} - \alpha_i v^{(i)}$
    **if** $\|s\|$ is small enough **then**
        $x^{(i)} = x^{(i-1)} + \alpha_i \hat{p}$
        **stop**
    **end**
    Solve $M\hat{s} = s$
    $t = A\hat{s}$
    $w_i = (t's)/(t't)$
    $x^{(i)} = x^{(i-1)} + \alpha_i \hat{p} + w_i \hat{s}$
    $r^{(i)} = s - w_i t$
    For continuation it is necessary that $w_i \neq 0$
**end**

The method is more costly in terms of required operations than BiCG, but does not involve the transpose of matrix $A$ in the computations; this can sometimes be an advantage. The other main advantages of BiCGSTAB is to avoid the irregular convergence pattern of BiCG and to usually show a better convergence speed.

## 3.7 Implementation of Nonstationary Iterative Methods

The codes for conjugate gradient type methods are easy to implement, but the interested user should first check the NETLIB repository. It contains the

package SLAP 2.0 that solves sparse and large linear systems using preconditioned iterative methods.

We used the MATLAB programs distributed with the Templates book [4] as a basis and modified this code for our experiments.

# 4 Practical Results

The presence of forward looking variables also creates a need for new solution techniques for simulating such models, see for example Fair and Taylor [9], Hall [18], Fisher and Hughes-Hallett [11], Laffargue [20], Boucekkine [5] and Bruaset [6].

In the following we first introduce a formulation of rational expectation (RE) models that will be used to analyze the structure of such models; we then briefly present the model MULTIMOD we used for our numerical experiments; we finally give some results for the different methods that were implemented.

## 4.1 Formulation of RE Models

Using the conventional notation, see for instance Fisher [10], a dynamic model with rational expectations is written

$$f_i(y_t, y_{t-1}, \ldots, y_{t-r}, y_{t+1|t-1}, \ldots, y_{t+h|t-1}, z_t) = 0 \qquad i = 1, \ldots, n \ , \quad (6)$$

where $y_{t+j|t-1}$ is the expectation of $y_{t+j}$ conditional on the information available at the end of period $t-1$, and $z_t$ represents the exogenous and random variables. For consistent expectations the forward expectations $y_{t+j|t-1}$ have to coincide with the next period's forecast when solving the model conditional on the information available at the end of period $t-1$. These expectations are therefore linked forward in time and to solve model (6) for each $y_t$ conditional on some start period 0 requires each $y_{t+j|0}$, for $j = 1, 2, \ldots, T-t$, and a terminal condition $y_{T+j|0}$, $j = 1, \ldots, h$.

To simplify notation we will now consider that the system of equations contains one lag and one lead, that is, we set $r = 1$ and $h = 1$ in (6). In order to discuss the structure of our system of equations it is also convenient to resort to a linearization. System (6) becomes therefore

$$D^t y_t + E^{t-1} y_{t-1} + A^{t+1} y_{t+1|t-1} = z_t \quad (7)$$

where we have $D^t = \frac{\partial f}{\partial y_t'}$, $E^{t-1} = -\frac{\partial f}{\partial y_{t-1}'}$ and $A^{t+1} = -\frac{\partial f}{\partial y_{t+1|t-1}'}$. Stacking up the system for period $t+1$ to period $t+T$ we get

$$\begin{bmatrix} E^{t+0} & \boxed{\begin{matrix} D^{t+1} & A^{t+2} & & & \\ E^{t+1} & D^{t+2} & A^{t+3} & & \\ & \ddots & \ddots & \ddots & \\ & & E^{t+T-2} & D^{t+T-1} & A^{t+T} \\ & & & E^{t+T-1} & D^{t+T} \end{matrix}} & A^{t+T+1} \end{bmatrix} \begin{bmatrix} y_{t+0} \\ y_{t+1} \\ y_{t+2} \\ \vdots \\ y_{t+T-1} \\ y_{t+T} \\ y_{t+T+1} \end{bmatrix} = \begin{bmatrix} z_{t+1} \\ z_{t+2} \\ \vdots \\ z_{t+T-1} \\ z_{t+T} \end{bmatrix} \tag{8}$$

and a consistent expectations solution to (8) is then obtained by solving

$$Jy = b$$

where $J$ is the boxed matrix in equation (8), $y = [y_{t+1} \ \ldots \ y_{t+T}]'$ and

$$b = \begin{bmatrix} z_{t+1} \\ \vdots \\ z_{t+T} \end{bmatrix} + \begin{bmatrix} -E^{t+0} \\ \vdots \\ 0 \end{bmatrix} y_{t+0} + \begin{bmatrix} 0 \\ \vdots \\ -A^{t+T+1} \end{bmatrix} y_{t+T+1}$$

are the stacked vectors of endogenous respectively exogenous variables which contain the initial condition $y_{t+0}$ and the terminal conditions $y_{t+T+1}$.

This system can then be either solved with a block iterative method or a Newton-like method. The computation of the Newton step requires the solution of a linear system for which we may consider different solution techniques. We concentrate on the use of nonstationary methods such as the ones presented in Section 3 and report results on numerical experiments with different nonstationary iterative solvers that are applied to find the step in Newton's method.

## 4.2 MULTIMOD

MULTIMOD (Masson, Symanski and Meredith [22]) is a multi-region econometric model developed by the International Monetary Fund in Washington. The model is available upon request and is therefore widely used for academic research.

MULTIMOD is a forward looking dynamic annual model and describes the economic behavior of the whole world decomposed into eight industrial zones and the rest of the world. The industrial zones correspond to the G7 and a zone called "small industrial countries" (SI), which collects the rest of the OECD. The rest of the world comprises two developing zones, i.e. high-income oil exporters (HO) and other developing countries (DC). The model mainly distinguishes three goods, which are oil, primary commodities and manufactures. The short-run behavior of the agents is described by error correction mechanisms with respect to their long-run theory based equilibrium. Forward looking behavior is modeled in real wealth, interest rates and in the price level determination.

The specification of the models for industrialized countries is the same and consists of 51 equations each. These equations explain aggregate demand, taxes and expenditures of the government, money and interest rates, prices and supply, and international balances and accounts.

The system is now stacked and the size of the nontrivial system to solve is $T \times 413$, where $T$ is the number of times the model is stacked. Figure 2 shows the pattern of the stacked model for $T = 10$.

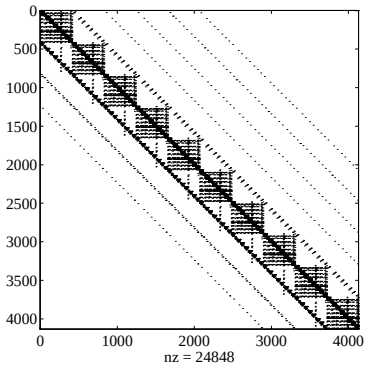


Figure 2: Incidence matrix of the stacked system for $T = 10$.

## 4.3 Application of Nonstationary Methods

The nonstationary solvers suited for nonsymmetric problems suggested in the literature we chose to experiment are BiCGSTAB, QMR and GMRES($m$). The QMR method (Quasi-Minimal Residual) introduced by Freund and Nachtigal [13] has not been presented in Section 3 since the method presents

potentials for failures without sophisticated look-ahead procedures. We tried a version of QMR without look-ahead strategies for our application. BiCGSTAB proposed by van der Vorst [26] is also designed to solve large and sparse nonsymmetric linear systems and usually displays robust features and small computational cost. Finally, we chose to experiment the behavior in our framework of GMRES($m$) originally presented by Saad and Schultz [24].

For all these methods, it is known that preconditioning can greatly influence the convergence. Therefore, following some authors (e.g. Concus [7], Axelsson [2] and Bruaset [6]), we applied a preconditioner based upon the block structure of our problem.

The block preconditioner we used is built on the LU factorization of the first block of our stacked system. If we dropped the leads and lags of the model, the Jacobian matrix would be block diagonal, i.e.

$$\begin{bmatrix} D^{t+1} & & & \\ & D^{t+2} & & \\ & & \ddots & \\ & & & D^{t+T} \end{bmatrix}.$$

The tradeoff between the cost of applying the preconditioner and the expected gain in convergence speed can be improved by using a same matrix $D$ along the diagonal. This is simplification is reasonable when the matrices display little change in their structure and values. We therefore selected $D^{t+1}$ for the diagonal block, computed its sparse LU factorization with partial pivoting and used it to perform the preconditioning steps in the iterative methods. Since the factorization is stored, only the forward and back substitutions are carried out for applying this block preconditioner.

The values in the Jacobian matrix change at each step of the Newton method and therefore a new LU factorization of $D^{t+1}$ is computed at each iteration. Another possibility involving a cheaper cost would have been to keep the factorization fixed during the whole solution process.

For our experiment we shocked the variable of government expenditures of Canada by 1% of the canadian GDP for the first year of simulation.

We report the results of the linear solvers in the classical Newton method. The figures reported are the average number of Mflops (millions of floating point operations) used to solve the linear system of size $T \times 413$ arising in the Newton iteration. The number of Newton steps needed to converge is 2 for $T$ less than 20 and 3 for $T = 30$.

Table 1 presents the figures for the solver BiCGSTAB. The column labeled "size" contains the number of equations in the systems solved and the one named "nnz" shows the number of nonzero entries in the corresponding matrices. This information is not repeated in the other tables for keeping them more compact.

| | | | tolerance | | |
|---|---|---|---|---|---|
| $T$ | size | nnz | $10^{-4}$ | $10^{-8}$ | $10^{-12}$ |
| 7 | 2891 | 17201 | 53 | 71 | 85 |
| 8 | 3304 | 19750 | 67 | 90 | 105 |
| 10 | 4130 | 24848 | 100 | 140 | 165 |
| 15 | 6195 | 37593 | 230 | 320 | 385 |
| 20 | 8260 | 50338 | 400 | 555 | 670 |
| 30 | 12390 | 75821 | 970 | 1366 | 1600 |
| 50 | 20650 | 126783 | * | * | * |

The symbol * indicates a failure to converge.

Table 1: Average number of Mflops for BiCGSTAB.

We remark that the increase in the number of flops is less than the increase in the logarithm of the tolerance criterion. This seems to indicate that for our case the rate of convergence of BiCGSTAB is, as usually expected, more than linear. The work to solve the linear system increases with the size of the set of equations; doubling the number of equations leads to approximately a fourfold increase in the number of flops.

Table 2 summarizes the results obtained with the QMR method. We choose to report only the flop count corresponding to a solution with a tolerance of $10^{-4}$. As for BiCGSTAB, the numbers reported are the average Mflops counts of the successive linear solutions arising in the Newton steps.

The increase of floating point operations is again linear in the size of the problem. The QMR method seems however about twice as expensive as BiCGSTAB to reach the same tolerance level. The computational burden of QMR consists of about 14 level-1 BLAS and 4 level-2 BLAS operations, whereas BiCGSTAB uses 10 level-1 BLAS and 4 level-2 BLAS operations. This apparently indicates a better convergence behavior of BiCGSTAB compared to QMR.

We present a summary of the results obtained with the GMRES($m$) technique in Table 3.

As in the previous methods, the convergence displays the expected superlinear convergence behavior. Another interesting feature of GMRES($m$) is

| $T$ | tol=$10^{-4}$ |
|---|---|
| 7 | 91 |
| 8 | 120 |
| 10 | 190 |
| 15 | 460 |
| 20 | 855 |
| 30 | 2100 |
| 50 | 9900* |

*Average of the first two Newton steps; failure to converge in the third step.

Table 2: Average number of Mflops for QMR.

| | $m = 10$ | | | $m = 20$ | | | $m = 30$ | | |
|---|---|---|---|---|---|---|---|---|---|
| | tolerance | | | tolerance | | | tolerance | | |
| $T$ | $10^{-4}$ | $10^{-8}$ | $10^{-12}$ | $10^{-4}$ | $10^{-8}$ | $10^{-12}$ | $10^{-4}$ | $10^{-8}$ | $10^{-12}$ |
| 7 | 76 | 130 | 170 | 74 | 107 | 135 | 76 | 109 | 145 |
| 8 | 117 | 170 | 220 | 103 | 125 | 190 | 110 | 150 | 200 |
| 10 | 185 | 275 | 355 | 175 | 250 | 320 | 165 | 240 | 310 |
| 15 | 415 | 600 | 785 | 430 | 620 | 810 | 460 | 660 | 855 |
| 20 | 725 | 1060 | 1350 | 705 | 990 | 1300 | 770 | 1085 | 1450 |
| 30 | 2566 | 3466 | 4666 | 2100 | 2900 | 3766 | 2166 | 3000 | 3833 |
| 50 | * | * | * | * | * | * | * | * | * |

The symbol $*$ indicates a failure to converge.

Table 3: Average number of Mflops for GMRES($m$).

the possibility to tune the restart parameter $m$. We know that the storage requirements increase with $m$ and that the larger $m$ becomes, the more likely the method converges, see Saad and Schultz [24, p. 867]. Each iteration uses approximately $2m + 2$ level-1 BLAS and 2 level-2 BLAS operations.

To back up with evidence the fact that the convergence will take place for sufficiently large $m$, we ran a simulation of our model with $T = 50$, tol=$10^{-4}$ and $m = 50$. In this case the solver converged with an average count of 9900 Mflops.

It is also interesting to notice that long restarts, i.e. large values of $m$, do not in general generate much heavier computations and that the increase in convergence may even lead to diminish the global computational cost. An operation count per iteration is given by Saad and Schultz [24], which clearly shows this feature of GMRES($m$).

Even though this last method is not cheaper than BiCGSTAB in terms of flops, the possibility to overcome nonconvergent cases by using larger values of $m$ certainly favors GMRES($m$).

Finally, we used the sparse LU solver provided in MATLAB. For general nonsymmetric matrices, a strategy that this method implements is to reorder the columns according to their minimum degree in order to minimize the fill-in. On the other hand, a sparse partial pivoting technique proposed by Gilbert and Peierls [14] is used to prevent losses in the stability of the method. We present some results obtained with the method just described in Table 4.

| $T$ | Mflops |
|---|---|
| 7 | 91 |
| 8 | 160 |
| 10 | 2950 |
| 15 | 7300[a] |
| 20 | * |
| 30 | * |
| 50 | * |

The symbol * indicates that the memory capacity (40 Mbytes) has been exceeded.

[a]Out of memory in the solution of second system.

Table 4: Average number of Mflops for MATLAB's sparse LU.

The direct solver obtains an excellent error norm for the computed solution, which in general is less than the machine precision for our hardware (i.e. about $2 \cdot 10^{-16}$). A drawback, however, is the steep increase in the number of arithmetic operations when the size of the system increases. This results in our situation favors the nonstationary iterative solvers for systems larger than $T = 10$, i.e. 4130 equation with about 25000 nonzero elements. Another major disadvantage of the sparse solver is that memory requirements became so quickly a constraint that we were not able to experiment with matrices of order larger than approximately 38000 with 40 Mbytes of memory. We may mention, however, that no special tuning of the parameters in the sparse method was performed for our experiments. Careful control of such parameters may probably allow to obtain better performance results.

# 5 Concluding Remarks

A class of models generating challenging problems in economics is the class of macroeconometric models containing forward looking variables. We presented an experiment of recent solution methods such as Newton combined with nonstationary linear techniques to solve a real model of the world economy MULTIMOD developed by the IMF in Washington. The special structure of such models gives rise to interesting possibilities to overcome the computational difficulties in the simulation of RE models. The savings obtained by using efficient solution techniques allow the user to investigate larger models in less time.

The recent nonstationary iterative methods proposed in the scientific computing literature are certainly an alternative to sparse direct methods for solving large and sparse linear systems such as the ones arising with forward looking macroeconomic models. Sparse direct methods have however the advantage of allowing to monitor the stability of the process and to reuse the structural information for instance in several Newton steps.

# References


[1] L. Adams. M-Step Preconditioned Conjugate Gradient Methods. *SIAM J. Sci. Stat. Comput.*, 6:452–463, 1985.

[2] O. Axelsson. Incomplete Block Matrix Factorization Preconditioning Methods. The Ultimate Answer? *J. Comput. Appl. Math.*, 12:3–18, 1985.

[3] O. Axelsson. *Iterative Solution Methods.* Oxford University Press, Oxford, UK, 1994.

[4] R. Barrett et al. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods.* SIAM, Philadelphia,PA, 1994.

[5] R. Boucekkine. An Alternative Methodology for Solving Nonlinear Forward-looking Models. *Journal of Economic Dynamics and Control*, 19:711–734, 1995.

[6] A. M. Bruaset. Efficient Solutions of Linear Equations Arising in a Nonlinear Economic Model. In M. Gilli, editor, *Computational Economics: Models, Methods and Econometrics*, Advances in Computational Economics. Kluwer Academic Press, Boston, MA, 1995.

[7] P. Concus, G. Golub, and G. Meurant. Block Preconditioning for the Conjugate Gradient Method. *SIAM J. Sci. Stat. Comput.*, 6:220–252, 1985.

[8] P. Dubois, A. Greenbaum, and G. Rodrigue. Approximating the Inverse of a Matrix for Use in Iterative Algorithms on Vector Processors. *Computing*, 22:257–268, 1979.

[9] R. C. Fair and J. B. Taylor. Solution and Maximum Likelihood Estimation of Dynamic Nonlinear Rational Expectations Models. *Econometrica*, 51(4):1169–1185, 1983.

[10] P. Fisher. *Rational Expectations in Macroeconomic Models.* Kluwer Academic Publishers, Dordrecht, 1992.

[11] P. G. Fisher and A. J. Hughes-Hallett. An Efficient Solution Strategy for Solving Dynamic Nonlinear Rational Expectations Models. *Journal of Economic Dynamics and Control*, 12:635–657, 1988.

[12] R. W. Freund, G. H. Golub, and N. M. Nachtigal. Iterative Solution of Linear Systems. *Acta Numerica*, pages 1–44, 1991.

[13] R. W. Freund and N. M. Nachtigal. QMR: A Quasi-mininal Residual Method for Non-Hermitian Linear Systems. *Numer. Math.*, 60:315–339, 1991.

[14] J. R. Gilbert and Peierls. Sparse Partial Pivoting in Time Proportional to Arithmetic Operations. *SIAM J. Sci. Statist. Comput.*, 9:862–874, 1988.

[15] M. Gilli, M. Garbely, and G. Pauletto. Equation Reordering for Iterative Processes — A Comment. *Computer Science in Economics and Management*, 5:147–153, 1992.

[16] G. H. Golub and J. M. Ortega. *Scientific Computing: An Introduction with Parallel Computing.* Academic Press, San Diego, CA, 1993.

[17] G. H. Golub and C. F. Van Loan. *Matrix Computations.* Johns Hopkins, Baltimore, 1989.

[18] S. G. Hall. On the Solution of Large Economic Models with Consistent Expectations. *Bulletin of Economic Research*, 37:157–161, 1985.

[19] M. R. Hestenes and E. Stiefel. Method of Conjugate Gradients for Solving Linear Systems. *J. Res. Nat. Bur. Stand.*, 49:409–436, 1952.

[20] J.-P. Laffargue. Résolution d'un modèle macroéconométrique avec anticipations rationnelles. *Annales d'Economie et Statistique*, 17:97–119, 1990.

[21] D. G. Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley, Reading, MA, second edition, 1989.

[22] P. Masson, S. Symanski, and G. Meredith. MULTIMOD Mark II: A Revised and Extended Model. Occasional Paper 71, International Monetary Fund, Washington D.C., July 1990.

[23] L. Paige and M. Saunders. Solution of Sparse Indefinite Systems of Linear Equations. *SIAM J. Numer. Anal.*, 12:617–629, 1975.

[24] Y. Saad and M. Schultz. GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM J. Sci. Stat. Comput.*, 7:856–869, 1986.

[25] J. C. Strikwerda and S. C. Stodder. Convergence Results for GMRES(m). Department of Computer Sciences, University of Wisconsin, August 1995.

[26] H. van der Vorst. BiCGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems. *SIAM J. Sci. Stat. Comput.*, 13:631–644, 1992.

Manfred Gilli & Giorgio Pauletto, "undated". "An Application of Nonstationary Iterative Methods for Solving a Multi-Country Model with Rational Expectations," Computing in Economics and Finance 1996 _045, Society for Computational Economics. Handle: RePEc:sce:scecf6:_045. as. HTML HTML with abstract plain text plain text with abstract BibTeX RIS (EndNote, RefMan, ProCite) ReDIF JSON. An Application of Nonstationary Iterative Methods for Solving a Multi-Country Model with Rational Expectations. Manfred Gilli. () and Giorgio Pauletto.Â  Abstract: In this paper we present an implementation of a Newton method based on iterative Krylov subspace methods such as GMRES, QMR and BiCGSTAB for solving large nonlinear macroeconometric models. These methods are tested for the solution of the model MULTIMOD and the computational costs of the different techniques are compared together with a sparse direct method. References: View references in EconPapers View complete reference list from CitEc Citations: View citations in EconPapers (1) Track citations by RSS feed. Keywords. Endogenous Variable Outer Loop Rational Expectation Time Block Multiple Time Period. These keywords were added by machine and not by the authors. This process is experimental and the keywords may be updated as the learning algorithm improves. This is a preview of subscription content, log in to check access.Â  Faust J., Tryon R. (1996) Block Distributed Methods for Solving Multi-Country Econometric Models. In: Gilli M. (eds) Computational Economic Systems. Advances in Computational Economics, vol 5. Springer, Dordrecht.