Board of Governors of the Federal Reserve System

International Finance Discussion Papers

Number 516

July 1995

BLOCK DISTRIBUTED METHODS FOR SOLVING MULTI-COUNTRY ECONOMETRIC MODELS

Jon Faust and Ralph Tryon

Abstract

This paper examines variations on a baseline Fair-Taylor algorithm used to solve multi-country, rational expectations models. One notable feature of these variations is the ability to exploit small-scale distributed processing using a network of workstations or PCs. Using four processors to solve MX-4 (152 endogenous variables), the largest speedup factor relative to Fair-Taylor is 59; for RE-7 (978 endogenous variables) the maximum speedup factor is 12.

Block Distributed Methods for Solving Multi-country Econometric Models

Jon Faust and Ralph Tryon[1]

1 Introduction

Despite advances in computer speed, solving large macroeconometric models with rational expectations can be very time consuming. Using the Fair-Taylor algorithm (Fair and Taylor, 1983), solving a model of 500 or more equations can take over an hour on a fast workstation. This paper demonstrates the potential efficiency gains from two variations on a basic Fair-Taylor algorithm in solving multi-country, rational expectations macroeconometric models. Both variations come from considering how better to exploit the block structure of the models.

The first variation breaks the models into their constituent country blocks and solves the full model by repeatedly solving the country blocks in an iterative scheme. This approach represents a return to the methodology used to solve the Project LINK models in the 1970s (Klein, 1983). One advantage of this approach is that it allowed the country models to be solved on separate computers, and it is the possibility of distributed processing that leads us back to this method. The second variation alters Fair-Taylor's period-by-period blocking by using Newton's method to solve multiple time periods simultaneously in a single block. This variation is motivated by recent theoretical and simulation work by Armstrong *et al.* (1994), Boucekkine (1994), Juillard (1994), Laffargue (1990).

This paper describes how these two variations can be combined to provide a family of solution algorithms and documents the benefits that may come from using such methods. For example, using four processors, the maximum observed speedup

[1] The authors are, respectively, a staff economist and a section chief at the International Finance Division of the Board of Governors of the Federal Reserve System. The authors thank Ray Board, Neil Ericsson, and Jaime Marquez for helpful comments. The views in this paper are solely the responsibility of the authors and should not be interpreted as reflecting the views of the Board of Governors of the Federal Reserve System or of any other person associated with the Federal Reserve System.

factor relative to Fair-Taylor is 59. Section 2 describes the solution of rational expectations models and lays out a general classification of mixed first-order iterative and Newton algorithms. Section 3 discusses how we implement distributed versions of the algorithms. Section 4 provides simulation results, and section 5 concludes.

2 Solving Rational Expectations Models

Rational expectations macroeconometric models involve equations of the general form

$$0 = g_j(Ey_{t+f|t}, \ldots, Ey_{t+1|t}, y_t, \ldots, y_{t-l}|z) + \varepsilon_t, \tag{1}$$

$j = 1, \ldots, M$, where y_t is $(M \times 1)$, z is a vector of exogenous variables, and $Ey_{t+j|t}$ stands for the expectation of y_{t+j} given the information available at t, as defined below.

The desired solution is a y_t, $t = 1, \ldots, T$, that satisfies (1) with ε_t set to zero for $t = 1, \ldots, T$, for fixed z and fixed initial and terminal values, y_{1-j}, $j = 1, \ldots, l$, and y_{T+j}, $j = 1, \ldots, f$. Also, at the solution the expectations variables are *model consistent* in the sense that $Ey_{t+f|t} = y_{t+f}$ for all t and f. (Clearly, E does not stand for the mathematical expectation).[2] Thus, the goal is to solve the system

$$0 = g_j(y_{t+f}, \ldots, y_{t+1}, y_t, \ldots, y_{t-l}|Z), \tag{2}$$

$j = 1, \ldots, M$, $t = 1, \ldots, T$, where Z includes z and the initial and terminal values.

The two models for which we report results are MX-4 (Gagnon, 1989) and RE-7 developed and maintained in the International Finance Division of the Federal Reserve Board.[3] These models are similar in structure to multi-country models such as MULTIMOD (Masson *et al.*, 1990) and Taylor's (1993) model. MX-4 is made up of four roughly symmetric country models. We take each country block as

[2] Issues of existence and uniqueness of a solution are often difficult to explore in these models. After linearizing the model around some Y^*, one can check whether the model satisfies Blanchard and Kahn's (1980) conditions for existence and local uniqueness of a solution.

[3] Ralph Tryon and Joseph Gagnon developed RE-7, which is a rational expectations version of the Multi-Country Model developed at the Board. See Edison *et al.* (1987) and Stevens *et al.* (1984) .

a separate model, and make each endogenous variable in the full model endogenous in one country model.[4] There are 39 endogenous variables in each country block and 9 forward looking variables; the maximum lead and lag (l and f in (2)) are each 3. Nine endogenous variables in each country block link it to the world by occurring (as exogenous variables) in some other country block. RE-7 has 7 country blocks, with the U.S. model slightly larger than the others. On average, the country blocks have 140 endogenous variables, 8 forward looking variables, and 6 endogenous variables occurring as exogenous variables in other country blocks. The maximum lead and lag in a typical country block are 3 and 11, respectively.

2.1 Solution Methods

This section describes a family of mixed first-order iterative and Newton algorithms. The discussion draws on Varga (1962) and Ortega and Rheinboldt (1970); for theoretical properties of these algorithms, see those sources. Fisher and Hughes Hallett (1988), Fisher (1992), and Armstrong *et al.* (1994) discuss these methods in the context of macroeconometric models.

Consider the equation system $G(Y) = 0$ formed by combining the $Q = M \cdot T$ equations (2), $j = 1, \dots, M$, $t = 1, \dots, T$, with the Z variables suppressed for expositional clarity. The algorithms we consider involve reordering the elements of Y and G and partitioning them into blocks. To do so, fix a one-to-one mapping between endogenous variables y_j and equations g_j, or more simply, suppose that we have normalized each equation on a different y_j. Whenever we re-order the elements of Y, the equations are similarly re-ordered.

Having fixed an order for the elements of Y, partition G and Y into n blocks so that the system can be written

$$0 = G_j(Y_1, \dots, Y_n) \qquad j = 1, \dots, n,$$

where for each j, G_j and Y_j have the same number of elements.

[4] It is generally clear in which country block each equation belongs. Where to put the exchange rate equations, however, is not so clear. In our models, the exchange rate equations all determine bilateral rates versus the U.S., and each is placed in the non-U.S. country block.

The simplest block first-order iterative scheme is block Jacobi. This method involves selecting a Y^0 and repeatedly solving

$$0 = G_j(Y_1^i, \ldots, Y_{j-1}^i, \mathbf{Y_j^{i+1}}, Y_{j+1}^i, \ldots, Y_n^i) \qquad j = 1, \ldots, n, \tag{3}$$

for Y_j^{i+1}, where i is the iteration count. Under Jacobi, Y_{j-1}^{i+1} is available, but unused, in calculating Y_j^{i+1}. The familiar Gauss-Seidel algorithm uses any updated values as soon as they become available, leading to iterations of the form

$$0 = G_j(Y_1^{i+1}, \ldots, Y_{j-1}^{i+1}, \mathbf{Y_j^{i+1}}, Y_{j+1}^i, \ldots, Y_n^i) \qquad j = 1, \ldots, n. \tag{4}$$

While we limit our exposition to the Jacobi and Gauss-Seidel first-order iterative schemes, there are several simple variations—Jacobi overrelaxation, and successive overrelaxation, for example—that often add significantly to computational efficiency (Fisher, 1992). Such schemes could be substituted for Jacobi and Gauss-Seidel in what follows.

Since each block G_j is in general nonlinear, evaluating G_j on each iteration itself requires some iterative method. Thus, the general form of the block Gauss-Seidel and block Jacobi algorithms in which the blocks are solved by an algorithm X is:

Block Gauss-Seidel/X and Block Jacobi/X Algorithms

```
1.  Set $Y^0$; set $i = 0$
2.  Loop:
3.          For $j = 1, \ldots, n$
4.                  Solve [(3) or (4)] for $Y_j^{i+1}$ using algorithm X
5.          End for
6.          Increment $i$
7.  Continue loop while $d(Y^i, Y^{i-1}) \geq \varepsilon$
```

Using equation (3) in step 4 gives block Jacobi; equation (4) gives block Gauss-Seidel. In step 7, d is some metric on the closeness of the successive values. The loop in steps 2–7 is called the *outer loop*, and the algorithm X iterations implicit in step 4 comprise the *inner loop*.

The Fair-Taylor algorithm can be interpreted as block Gauss-Seidel with the X algorithm simple Gauss-Seidel. To see this, arrange Y as $Y = (y_1', \ldots, y_T')'$ and treat each time period as a block; thus, $Y_j = y_j$. On each execution of the outer loop the value of y_t^{i+1}, $t = 1, \ldots, T$ is solved by Gauss-Seidel, conditional on the value of y_s^{i+1}, $s = 1, \ldots, t-1$ and conditional on the previous iteration value of y_s^i, $s = t+1, \ldots, T$.

The remainder of this section lays out our variations on the Fair-Taylor algorithm. While we consider algorithms like Fair-Taylor that have only an inner and outer loop, we also consider block Jacobi/X and block Gauss-Seidel/X algorithms in which the X algorithm is itself block Jacobi or block Gauss-Seidel, requiring a further loop. This nesting of loops could continue, of course, but the algorithms we consider are nested at most two deep, and will be denoted *outer/middle/inner*, where *outer* and *middle* and *inner* give the type of iteration in the relevant loop.

Independent of the level of nesting, in all the algorithms considered, the inner most loop is Newton iterations.[5] Thus, for example, our baseline Fair-Taylor algorithm is a Gauss-Seidel/Newton algorithm.[6]

2.2 Variation One: Country blocks

Our first variation is motivated by the desire to exploit small-scale distributed processing on widely available hardware such as a network of workstations or PCs. Distributed processing is most simply applied to an algorithm with sub-problems that have independent computational blocks. When the algorithm comes to these sub-problems, the independent computational blocks can be sent to different processors, solved, and the answers collected and passed to the next sub-problem. Obviously, distributed processing will save time only so long as the gains from simultaneous

[5] As applied to equation block j, Newton's method involves iterations of the form, $Y_j^{i+1} = Y_j^i - H_j^{-1}(Y_j^i)G(Y_j^i)$ where $H_j(Y^i) = \frac{\partial G_j(Y_j^i)}{\partial Y_j}$, and the arguments of G_j other than Y_j are fixed and suppressed.

[6] Don and Gallo (1987) have emphasized that the efficiency of Newton algorithms can be greatly increased by taking account of the structure of the Jacobian. How best to do this is a topic of ongoing research. See, e.g., Hughes Hallett and Fisher, 1990; Gilli, 1992; Gilli *et al.*, 1992. Our algorithms do not exploit this research.

evaluation of blocks outweigh the overhead involved in managing the multiple processors, communicating the problems to the processors, and collecting the answers.

Block Jacobi algorithms offer an obvious opportunity for distribution: in steps 3–5, the n equation blocks are independent and can be solved simultaneously. In contrast, the blocks in Gauss-Seidel must be solved sequentially. For distribution to be efficient, we want solution of the n blocks to be computationally intensive relative to the amount of information that must be communicated among the blocks between iterations—that is we want to structure the problem so that it is *coarse grained*, with a good *computation-to-communication* ratio.[7]

An obvious choice in the case of multi-country models is to order and block Y by country.[8] The country-oriented block Jacobi and Gauss-Seidel algorithms will be called Jacobi(C)/X and Gauss(C)/X, where C indicates that the model is blocked by country, and X is the algorithm used to solve the country blocks.

Intuitively, the Jacobi form of the algorithm involves solving each country model separately, conditional on the values of variables for its trading partners. This solution results in new values for each country's trading partners, which form the basis of a new iteration. The procedure is continued until convergence. This procedure is likely to deliver the high computation-to-communication ratio needed efficiently to exploit of a small number of processors. The country blocks are large enough to be computationally intensive to solve, and the number of variables linking the models is small, limiting the amount of information that must be communicated among the blocks between outer-loop iterations.

Although our main motivation for considering country blocking is the opportunity it affords for distributed processing, country blocking may be computationally efficient on a single processor.[9] The underlying reason for this is that the dimension of each country model is small relative to the full model, and the channels of interac-

[7] Wilson (1993) provides a useful introduction to the jargon of distributed computing.

[8] Each equation j of (2) is assigned to one country block and that equation for all time periods is included in the block.

[9] This point has been illustrated, for example, in the context of solving multi-country models from general equilibrium theory (Mansur and Whalley, 1982; van der Laan, 1985).

tion between countries are few and are generally weak relative to interactions within countries. More precisely, when blocked by country, the Jacobian of the model is nearly block diagonal and the non-zero off-block-diagonal terms are small relative to those within the blocks.[10]

2.3 Variation Two: Time blocks

In the Jacobi(C)/X and Gauss(C)/X algorithms, the inner loop uses algorithm X to repeatedly solve the country models. While a natural starting point would be to use Fair-Taylor for the X algorithm, the country models in MX-4 and RE-7 are relatively small, leading us to consider other approaches. The approach we use is motivated by work of Armstrong *et al.* (1994), Boucekkine (1994), Juillard (1994), and Laffargue (1990), which provides theoretical and simulation evidence that when the number of equations times the number of time periods in a block is not too large, the entire equation block can efficiently be solved as one simultaneous system using Newton's method.

We consider an array of options between Fair-Taylor and Newton's method as applied to the entire block. As noted above, in our baseline Fair-Taylor algorithm, the outer loop involves block Gauss-Seidel iterations treating each time period as a separate block and solving the time blocks using Newton's method. It is a straightforward extension to alter the algorithm's time blocking by partitioning Y into b blocks of T/b periods each—the first block includes the equations for $(y_1', \ldots, y_{T/b}')'$, and so on.[11] Of course, taking $b = T$ gives the baseline Fair-Taylor algorithm, and setting $b = 1$ eliminates the need for outer loop iterations entirely by using Newton's method to solve all time periods simultaneously. The idea of considering intermediate cases between $b = 1$ and $b = T$ is due to Peter Hollinger.[12]

[10] Faust and Tryon (1994) give a more complete discussion of this point.

[11] This assumes that T/b is an integer, a convenient assumption in both notation and algorithm implementation, which is maintained throughout.

[12] Intex Solutions, Inc., Needham, Massachusetts. Hollinger and Leonid Spivakovsky wrote Portable TROLL, the software used in our simulations, drawing on the mainframe TROLL program developed at the Massachusetts Institute of Technology and the National Bureau of Economic Research.

In general, this algorithm will be denoted Gauss(T)/X, where T signifies that the equations are blocked by time, and X is the algorithm used to solve the time blocks.[13]

2.4 Combining Country and Time Blocking

We can now state four basic algorithms we consider:

1 Gauss(T)/Newton. This algorithm ignores the country block structure. The baseline Fair-Taylor algorithm is a special case with the number of time blocks set equal to the number of time periods in the solution.

2 Gauss(C)/Gauss(T)/Newton. Use Gauss-Seidel iterations over country blocks and solve the country blocks using method 1.

3 Jacobi(C)/Gauss(T)/Newton. Use Jacobi's method in place of Gauss-Seidel in the iterations over country blocks.

4 Distributed Jacobi(C)/Gauss(T)/Newton. Solve the country blocks in method 3 simultaneously on different processors.

Each of the four types involves Gauss(T) iterations, which may involve different numbers of time blocks. In our simulations, we use 5 different time blockings for each of the four basic algorithms, giving 20 different methods in all.[14]

3 Implementation of the Distributed Algorithm

This section provides a stylized description of our implementation of the distributed Jacobi algorithm (for details, see Faust and Tryon, 1994). We choose a low-tech approach that can be implemented easily without special hardware or software. The only hardware requirement for the algorithm is to have multiple processors (perhaps separate computers) that share access to some file storage device.

The algorithm is programmed in a simple master-worker setup. There is one worker process per country block, solving the country block using the specified variant of Fair-Taylor. The master coordinates actions of the worker programs,

[13] For simplicity, we do not consider Jacobi iterations over time blocks.

[14] There are, of course, many further alternatives we could have considered; see Gilli and Pauletto (1994)

and both the master and worker programs are implemented in Portable TROLL's programming language. All communication between master and worker is done through disk files on the shared file server. This method of communication is simple, but extremely portable, allowing us to run the algorithm essentially unaltered on a network of DOS-based PCs as well as a multi-processor UNIX workstation.

The master controls the operation of the workers with a simple program:

Master program

```
1. Initialize all workers
2. Loop:
3.         Start each worker
4.         Wait until all workers stop
5.         Read status of each worker
6. Continue loop until all workers report convergence
7. Stop all workers
```

The worker program implements the core of the algorithm. The variable W_j^i in the worker program refers to those elements of Y_j^i that enter some equation in any block other than the j^{th} block. These are the only variables that must be communicated by worker j to some other worker.[15] The convergence metric we use is

$$d(v, w) = \max_{k \in 1,\ldots,n} \frac{|v_k - w_k|}{|w_k| + \gamma} \tag{5}$$

for any $(n \times 1)$ vectors v and w, and in the simulations $\gamma = 10$.

[15] Worker k only needs to read in those elements of W_j, $j \neq k$, that are used in block k. The workers in our implementation exploit this efficiency.

Program for worker j

```
1.  Outer loop:
2.          Wait until master says start
3.          Read $W_k^0, k \neq j$, from common space
4.          Set $Y_j^0 = Y_j^i$ from previous loop or initialize
5.          Store $W_j^0$
6.          $i = 0$
7.          Middle loop:
8.                  Solve for $Y_j^{i+1}$ using inner loop algorithm
9.                  Increment $i$
10          Continue loop while $d(Y_j^i, Y_j^{i-1}) \geq \varepsilon$ and $i < \bar{\imath}$
11.         Write $W_j^i$ to common space
12.         If $d(Y_j^i, Y_j^{i-1}) < \varepsilon$ and $d(W_j^i, W_j^0) < \varepsilon$, report convergence;
                    otherwise report nonconvergence
13. Continue loop until master says stop
```

As implemented in step 11, the inner loop iterations stop in two cases: if the convergence criterion for the problem is met, and if a maximum iteration count, $\bar{\imath}$, is reached. The workers stop after $\bar{\imath}$ iterations even if they have not converged in order to allow the workers to share whatever progress they have made. If $\bar{\imath}$ is too high, time may be wasted on early outer loop iterations needlessly refining solutions for the blocks. With $\bar{\imath}$ too low, too much time may be spent sharing results that have not changed much from the previous outer loop iteration.

4 Application to Multi-Country Models

4.1 The Experiments

Simulation results are reported for two models, MX-4 and RE-7, as described above. The experiments each take as an initial condition a baseline solution and involve solving the model for the change from baseline caused by a *shock* to an exogenous variable. We report results for one shock, a permanent 2 percent change in the basic money stock of one country.[16] In these models, money is essentially neutral in the long-run, but not in the short-run, and the shock immediately affects all countries through its effects on nominal exchange rates. The solution horizon in all cases is 120 periods (40 years in quarterly data).

4.2 Hardware and Software Implementation

The models and algorithms are implemented in Portable TROLL running under the SunOS (UNIX) operating system. Because we rely on TROLL's Newton and time blocking algorithms, it is important that this software provide an efficient environment for benchmarking the algorithms. Brillet (1994) presents some evidence that TROLL's Fair-Taylor algorithm is very efficient relative to other available packages.[17] The efficiency of solving multiple time periods simultaneously using Newton's method depends on the sparse matrix routines for handling the associated Jacobian. The results below suggest that TROLL's Harwell MA28 sparse matrix routines become inefficient when the matrix size is very large.

The hardware platform is a Solbourne model 704/6E UNIX server with four processors. When we have more than four country blocks, they share the four processors. We have also tested the algorithm on 80486-based PCs running under DOS and communicating across a token ring. The results, some of which are reported in Faust and Tryon (1994), are similar to those reported below.

4.3 The Results

The upper panel of Table 1 presents results for MX-4 with the convergence criterion set to $\varepsilon = 0.0001$. The first row gives the results for the algorithms that do not exploit the country block structure. The baseline Fair-Taylor algorithm solved the

[16] The permanent money shock proportionally changes the long-run steady-state values for certain nominal variables. In our simulations, the terminal conditions for the forward looking nominal variables are appropriately adjusted. The 120 period horizon used is long enough for the model to return to the steady state.

[17] On limited range of experiments with a 501 equation model, TROLL's Fair-Taylor algorithm was five times faster in simulating than the nearest of the three other packages tested.

Table 1: Solution times and outer loop iterations for MX-4.

Outer loop	Time blocks in middle loop				
Type-CPUs	120	60	12	4	1
			$\varepsilon = 0.0001$		
[none]-1	1295 (152)	1525 (151)	1745 (63)	.	.
Gauss(C)-1	1069 (94)	741 (62)	463 (26)	460 (21)	180 (8)
Jacobi(C)-1	789 (61)	709 (53)	701 (32)	.	299 (14)
Jacobi(C)-4	250 (61)	228 (53)	206 (32)	.	99 (14)
			$\varepsilon = 0.00001$		
[none]-1	11057 (1246)	11027 (926)	3681 (114)	.	.
Gauss(C)-1	6930 (553)	5715 (398)	1038 (53)	710 (28)	289 (12)
Jacobi(C)-1	6790 (512)	5273 (354)	1054 (52)	.	564 (22)
Jacobi(C)-4	2001 (512)	1787 (354)	337 (52)	.	187 (22)

Time in seconds; outer loop iterations in parentheses; "." means did not converge; *CPUs* means number of processors used. The middle loop is Gauss-Seidel over the specified number of time blocks with $\bar{\imath} = 2$. The inner loop is Newton.

model in 1295 seconds with 152 outer loop iterations. Reducing the number of time blocks from 120 to 12 reduces the outer loop iterations, but raises the solution time by a factor of 1.3. For less than 12 time blocks the algorithm did not converge.[18] This result appears to reflect numerical problems with handling the large Jacobian rather than instabilities in the model.

The second and third rows of the top panel show impressive efficiency gains for the single-processor algorithms that exploit the country block structure.[19] With 120 time blocks, the speedup factors relative to Fair-Taylor for Gauss-Seidel and Jacobi are 1.2 and 1.6, respectively. The fact that Gauss-Seidel is less efficient than Jacobi in this case is an anomaly. Reducing the number of time blocks provides much more favorable results when the time blocking is applied to the country blocks, as opposed to the full model. For Gauss-Seidel, using one time block instead of 120

[18] The expression *did not converge* covers several events: a maximum iteration count may have been reached, the algorithm may have diverged, or the algorithm may simply have stopped due to problems handling the Jacobian.

[19] Steven Symansky at the International Monetary Fund has reported similar results to the authors using MULTIMOD.

blocks requires less than one-tenth as many iterations and gives a speedup factor of 5.9.

The final row of the first panel gives the results for distributing the Jacobi(C) iterations over four processors. On a single processor Gauss-Seidel will generally be more efficient than Jacobi for resolving interactions among the country blocks. Thus, the gain from moving to four processors can be seen as the net effect of any loss from going from Gauss-Seidel to Jacobi on a single processor and any gain from distributing the Jacobi algorithm. For 120 time blocks, single processor Jacobi takes 1.7 times as long as Gauss-Seidel; but distributing the Jacobi algorithm gives a speedup of 3.0, for an net speedup factor of 1.8.

For a fixed number of time blocks, the maximum theoretical speedup factor from distributing the Jacobi(C) algorithm over four processors is four. Three factors account for the actual speedup, typically between 3 and 3.5, shown in Tables 1 and 2. First, there are certain overhead costs associated with running the worker processes. Second, the distributed algorithm requires inter-processor communication overhead. Third, the distributed algorithm has a *synchronization barrier*: at the end of each outer loop iteration, each worker must wait until all others have finished (worker algorithm, step 2). Thus, if the load on the workers is not balanced, some processors may lay idle for a time. Rough calculations indicate that over 80 percent of the shortfall relative to a speedup factor of 4 is due to the synchronization barrier.

Finally, comparing the results for the two values of ε suggests that the efficiency gain from reducing the number of time blocks grows as the convergence criterion is tightened. With $\varepsilon = 0.00001$, the distributed Jacobi algorithm is 10.7 times faster with 1 time block than with 120, as compared with the speedup factor of 2.5 for $\varepsilon = 0.0001$. This fact is consistent with the single-time-block algorithm having a higher asymptotic rate of convergence than the algorithms with more time blocks. With $\varepsilon = 0.00001$ the distributed Jacobi algorithm with one time block is 59 times faster than the baseline Fair-Taylor algorithm.

The results for the larger model, RE-7, are shown in Table 2. Without country

Table 2: Solution times and outer loop iterations for RE-7.

Outer loop	Time blocks in middle loop				
Type-CPUs	120	60	12	4	1
			$\varepsilon = 0.0001$		
[none]-1	6080 (97)	3939 (52)	10085 (22)	.	.
Gauss(C)-1	2541 (43)	1591 (27)	1085 (11)	1811 (6)	.
Jacobi(C)-1	2635 (44)	1728 (28)	2068 (21)	3446 (10)	.
Jacobi(C)-4	795 (44)	516 (28)	618 (21)	1073 (10)	.
			$\varepsilon = 0.00001$		
[none]-1	11677 (189)	10504 (110)	21581 (44)	.	.
Gauss(C)-1	5859 (93)	3897 (55)	2593 (22)	3458 (10)	.
Jacobi(C)-1	5780 (90)	5366 (71)	3875 (32)	5015 (14)	.
Jacobi(C)-4	1653 (90)	1569 (71)	1162 (32)	1843 (14)	.

See the notes to Table 1. There are 7 country blocks.

blocking, reducing the number of time blocks from 120 to 12 monotonically reduces the number of outer loop iterations, but the effect on solution time is not monotonic in this case. Country blocking on a single processor with 120 time blocks results in more than a two times speedup over the baseline algorithm.

In the country block algorithms, there are substantial gains from reducing the number of time blocks, but these gains are not monotonic, and the algorithm would not solve at all with 1 time block. These results appear to be caused by difficulty handling the large Jacobians—the typical country block in RE-7 is almost as large as all of MX-4, and the U.S. model is larger than MX-4. While the benefits from reducing the number of time blocks are not as impressive with the bigger model, the speedup factor for the distributed Jacobi algorithm over the baseline Fair-Taylor algorithm are slightly more impressive: 7.6 versus 5.2 (120 time blocks, $\varepsilon = 0.0001$).

5 Conclusions

The most widely used approach to solving macroeconometric models is the Fair-Taylor algorithm, which treats each time period as a separate block and includes each equation for that time period in the block. This paper explores the potential of algorithms that exploit the block structure of macroeconometric models in different ways. Our first variation, blocking the models by country and then distributing the country blocks to multiple processors, reduces solution times fairly consistently. The second variation, solving multiple time periods simultaneously within a country block, gives mixed results. On MX-4, with country blocks of 39 endogenous variables, solving all time periods in a single block reduces solution times by a factor of 10 in some cases. On RE-7, with country blocks of between 130 and 190 endogenous variables, solving multiple times periods simultaneously yields smaller reductions in solution times and leads to convergence problems when the number of time periods solved simultaneously is large.

References

Armstrong, J., Black, R., Laxton, D., and Rose, D. (1994) A Fast and Robust Method for Solving Rational Expectations Models, Bank of Canada Technical Report, forthcoming.

Blanchard, O.J., and Kahn, C.M. (1980) The Solution of Linear Difference Models under Rational Expectations, *Econometrica*, Vol. 48, pp. 1305–1311.

Boucekkine, R. (1994) An Alternative Methodology for Solving Nonlinear Forward-Looking Models, *Journal of Economic Dynamics and Control*, forthcoming.

Brillet, J. (1994) Solving Large Models on Micro-Computers: a Review of Available Packages, manuscript, INSEE.

Don, F., and Gallo, G. (1987) Solving Large Sparse Systems of Equations in Econometric Models, *Journal of Forecasting*, Vol. 6, pp. 167–180.

Edison, H., Marquez, J., and Tryon, R. (1987) The Structure and Properties of the Federal Reserve Board Multicountry Model, *Economic Modelling*, Vol. 4, pp. 115–315.

Fair, R. and Taylor, J. (1983) Solution and Maximum Likelihood Estimation of Dynamic Nonlinear Rational Expectations Models, *Econometrica*, Vol. 51, pp. 1169–1186.

Faust, J., and Tryon, R. (1994) A Distributed Block Approach to Solving Near-Block-Diagonal Systems with an Application to a Large Macroeconometric Model, IFDP No. 488, Federal Reserve Board.

Fisher, P. (1992) *Rational Expectations in Macroeconometric Models,* Kluwer Academic Publishers, Boston.

Fisher, P. and Hughes Hallett, A. (1988) Efficient solution Techniques for Linear and Non-Linear Rational Expectations Models, *Journal of Economic Dynamics and Control*, Vol. 12, pp. 635–657.

Gagnon, J. (1989) A Forward-Looking Multicountry Model: MX-3, IFDP No. 359, Federal Reserve Board.

Gilli, M. (1992) Causal Ordering and Beyond, *International Economic Review,* Vol. 33, pp. 957–971.

Gilli, M., and Pauletto, G. (1994) Parallel Algorithms for Solving Rational Expectations Models, manuscript, University of Geneva.

Gilli, M., Pauletto, G., and Garbeley, M., (1992) Equation Reodering for Iterative Processses—A Comment, *Computer Science in Economics and Management*, Vol. 5, pp. 147–153.

Juillard, M. (1994) DYNARE, a Program for the Resolution of Non-linear Models with Forward-looking Variables. Release 1.1, manuscript, CEPREMAP.

Laffargue, J. (1990) Résolution d'un Modèl Macroéconomique à Anticipations Rationnelles, *Annales d'Economie et de Statistique*, Vol. 17.

Klein, L. (1983) *Lectures in Econometrics*, Elsevier, Amsterdam.

Mansur, A. and Whalley, J. (1982) A Decomposition Algorithm for General Equilibrium Computation with Application to International Trade Models, *Econometrica*, Vol. 50, pp. 1547–2557.

Masson, P., Symansky, S., and Meredith, G. (1990) MULTIMOD Mark II: A Revised and Extended Model, Occasional Paper No. 71, International Monetary Fund.

Ortega, J. and Rheinboldt, W. (1970) *Iterative Solution of Nonlinear Equations in Several Variables,* Academic Press, New York.

Stevens, G., Berner, R., Clark, J., Hernandez-Cata, E., Howe, H., and Kwack, S. (1984) *The U.S. Economy in an Interdependent World: A multicountry Model,* Federal Reserve Board.

Taylor, J. (1993) *Macroeconomic Policy in an International Context: From Econometric Design to Practical Operation,* W.W.Norton, New York.

van der Laan, G. (1985) The Computation of General Equilibrium in Economies with a Block Diagonal Pattern, *Econometrica*, Vol. 53, pp. 659–665.

Varga, R. (1962) *Matrix Iterative Analysis.* Prentice-Hall, Englewood Cliffs, New Jersey.

Wilson, G. (1993) A Glossary of Parallel Computing Terminology, *IEEE Parallel and Distributed Technology,* Vol. 1, pp. 52–67.

International Finance Discussion Papers

IFDP Number	Titles	Author(s)
	1995	
516	Block Distributed Methods for Solving Multi-country Econometric Models	Jon Faust Ralph Tryon
515	Supply-side sources of inflation: evidence from OECD countries	Prakash Loungani Phillip Swagel
514	Capital Flight from the Countries in Transition: Some Theory and Empirical Evidence	Nathan Sheets
513	Bank Lending and Economic Activity in Japan: Did "Financial Factors" Contribute to the Recent Downturn?	Allan D. Brunner Steven B. Kamin
512	Evidence on Nominal Wage Rigidity From a Panel of U.S. Manufacturing Industries	Vivek Ghosal Prakash Loungani
511	Do Taxes Matter for Long-Run Growth?: Harberger's Superneutrality Conjecture	Enrique G. Mendoza Gian Maria Milesi-Ferretti Patrick Asea
510	Options, Sunspots, and the Creation of Uncertainty	David Bowman Jon Faust
509	Hysteresis in a Simple Model of Currency Substitution	Martin Uribe
508	Import Prices and the Competing Goods Effect	Phillip Swagel
507	Supply-side Economics in a Global Economy	Enrique G. Mendoza Linda L. Tesar
506	The Lucas Critique In Practice: Theory Without Measurement	Neil R. Ericsson John S. Irons
505	Real Exchange Rate Targeting and Macroeconomic Instability	Martin Uribe
504	Inferences from Parametric and Non-Parametric Covariance Matrix Estimation Procedures	Wouter J. Den Haan Andrew T. Levin
503	Exchange-Rate Based Inflation Stabilization: The Initial Real Effects of Credible Plans	Martin Uribe

Please address requests for copies to International Finance Discussion Papers, Division of International Finance, Stop 24, Board of Governors of the Federal Reserve System, Washington, DC 20551.

International Finance Discussion Papers

IFDP Number	Titles	Author(s)
	1995	
502	Strategic Returns to International Diversification: An Application to the Equity Markets of Europe, Japan, and North America	John Ammer Jianping Mei
501	Real Exchange Rate Movements in High Inflation Countries	John H. Rogers Ping Wang
500	Political Competition, Casual Relations Between Taxes and Spending, and Their Influence on Government Size: Evidence From State-Level Data	Diane Lim Rogers John H. Rogers
499	International Stock Price Spillovers and Market Liberalization: Evidence From Korea, Japan, and the United States	Sang W. Kim John H. Rogers
498	How Wide is the Border?	Charles Engle John H. Rogers
497	Constrained Suboptimality in Economies with Limited Communication	David Bowman
496	Saving-Investment Associations and Capital Mobility On the Evidence from Japanese Regional Data	Robert Dekle
495	Convertibility Risk, Default Risk, and the Mexdollar Anomaly	John H. Rogers
494	Government Budget Deficits and Trade Deficits Are Present-Value Constraints Satisfied in Long-Term Data?	Shaghil Ahmed John H. Rogers
493	Real Shocks and Real Exchange Rates in Really Long-Term Data	John H. Rogers
	1994	
492	Loss Aversion in a Consumption/Savings Model	David Bowman Deborah Minehart Matthew Rabin
491	Terms-of-Trade Uncertainty and Economic Growth: Are Risk Indicators Significant in Growth Regressions	Enrique G. Mendoza