Solution of Simultaneous Equations Models in high performance systems

José-Juan López-Espín *,1 Domingo Giménez **

Abstract

The solution of Simultaneous Equations Models in high performance systems is analyzed. Parallel algorithms of the Ordinary Least Squares, the Indirect Least Squares and the Two-stage Least Squares methods are developed. Algorithms for shared memory (using OpenMP) and distributed memory (using MPI) are studied theoretically and experimentally. The algorithms make an extensive use of basic libraries like BLAS and LAPACK to obtain efficient and portable versions.

Key words: Simultaneous Equation Models, High Performace Computing, Econometrics

1 Introduction

The solution of Simultaneous Equations Models in high performance parallel systems is studied. These models can be solved through a variety of methods. The methods analyzed here are Indirect Least Squares (ILS) and Two-stage Least Squares (2SLS). Both techniques use Ordinary Least Squares (OLS) as their basis, and this has also been analyzed.

Traditionally the Simultaneous Equation Models have been used in econometrics [5,6], but recently they have begun to be used in networks simulation [1],

* Facultad de Ciencias Experimentales, Universidad Miguel Hernández, 03202 Elche, Spain

**Departamento de Informática y Sistemas, Universidad de Murcia, 30071 Murcia, Spain

Email addresses: jlopez@umh.es (José-Juan López-Espín), domingo@dif.um.es (Domingo Giménez).

[1] Corresponding author

biological microsystems, psychology [14], medicine [2,4], and even to study the air traffic in New York [8] or the efect of cabin altitude and arterial oxygen saturation level on passenger comfort during a prolonged flight [7].

In 1994 QMS began the development of software for this type of systems, with Eviews 1.0. The last version is 6.0. Eviews includes linear regression techniques, solution of Simultaneous Equations Models, times series, and other econometric problems [10].

One free tool for econometric is Ox [9]. This software includes fewer problems than Eviews, althougt it includes those studied in this work.

Other statistics packages include 2SLS, but this is used in the solution of linear regression equations, and not for solving Simultaneous Equation Models. For example, the SPSS 12.0 includes 2SLS.

All this software is for sequential processors, but in some cases the models to solve are very large (modelization of the world economy) and it may be preferable to solve the systems using high performance computers. The only previous works we know on parallel algorithms to solve this type of problems are those by Kontoghiorghes [3]. In that work parallel algorithms for OLS and full information methods (3SLS) are studied. In our work, parallel algorithms for the solution of Simultaneous Equations Models using limited information techniques (ILS and 2SLS) are developed**,** both for shared memory systems (using OpenMP) and for distributed memory systems (using MPI). Additionally, they use basic libraries like BLAS and LAPACK intensively to obtain efficient and portable versions of the algorithms.

The remainder of the paper is structured as follows. In Section 2, Simultaneous Equations Models and sequential algorithms to solve them are explained. In Section 3, parallel algorithms for OLS, ILS and 2SLS are developed both for shared and distributed memory systems. In Section 4, some experimental results are shown. Finally, in Section 5 the work will be summarized, and future research lines will be outlined.

2 Solution of Simultaneous Equations Models in sequential systems

A Simultaneous Equations System is a regression equation system where three types of variables appear:

- Endogenous variables. They are internal variables of the system, which influence, and are influenced by, the other variables.

- Predetermined variables. They influence the system, but are not influenced by the system. They can be: exogenous (external to the system) and endogenous in the time (they are lagged endogenous variables, which influence the system, but can not be influenced because their data are previous).
- White noise (random variables). They form the non controllable part of the regression equation.

The scheme of a system with M equations, M endogenous variables and k predetermined variables is:

$$\begin{aligned}
&Y_{1,t} = \beta_{1,2}Y_{2,t} + \beta_{1,3}Y_{3,t} + \ldots + \beta_{1,M}Y_{M,t} + \gamma_{1,1}X_{1,t} + \ldots + \gamma_{1,k}X_{k,t} + u_{1,t} \\
&Y_{2,t} = \beta_{2,1}Y_{1,t} + \beta_{2,3}Y_{3,t} + \ldots + \beta_{2,M}Y_{M,t} + \gamma_{2,1}X_{1,t} + \ldots + \gamma_{2,k}X_{k,t} + u_{2,t} \\
&\ldots \\
&Y_{M,t} = \beta_{M,1}Y_{1,t} + \ldots + \beta_{M,M-1}Y_{M-1,t} + \gamma_{M,1}X_{1,t} + \ldots + \gamma_{M,k}X_{k,t} + u_{M,t}
\end{aligned} \tag{1}$$

where $Y_1, Y_2, \ldots, Y_M$ are endogenous variables, $X_1, X_2, \ldots, X_k$ are predetermined variables, and $u_1, u_2, \ldots, u_M$ are random variables.

Equation 1 can be represented in matrix form as:

$$BY_t + \Gamma X_t + u_t = 0 \tag{2}$$

with:

$$\begin{aligned}
Y_t = \begin{pmatrix} Y_{1,t} \\ \ldots \\ Y_{M,t} \end{pmatrix} \ , \ X_t = \begin{pmatrix} X_{1,t} \\ \ldots \\ X_{k,t} \end{pmatrix} \ , \ u_t = \begin{pmatrix} u_{1,t} \\ \ldots \\ u_{M,t} \end{pmatrix} \\
\Gamma = \begin{pmatrix} \gamma_{1,1} & \ldots & \gamma_{1,k} \\ & \ldots & \\ \gamma_{M,1} & \ldots & \gamma_{M,k} \end{pmatrix} \ , \ B = \begin{pmatrix} \beta_{1,1} & \ldots & \beta_{1,k} \\ & \ldots & \\ \beta_{M,1} & \ldots & \beta_{M,k} \end{pmatrix}
\end{aligned} \tag{3}$$

The problem consists in obtaining $\beta_{1,1}, \beta_{1,2}, \ldots, \beta_{M,M-1}, \gamma_{1,1}, \ldots, \gamma_{M,k}$ from a representative sample of the model.

The structural model 2 can be expressed in reduced form:

$$Y_t = \Pi X_t + v_t \tag{4}$$

with:

$$\Pi = -B^{-1}\Gamma \ , \ v_t = -B^{-1}u_t \tag{5}$$

To analyze the algorithms a number of variables will be used. Predetermined variables are called exogenous (they can also be lagged endogenous). The system will be in structural form, and each equation has one endogenous variable (called main endogenous) as a function of the other variables. The variables which appear in the algorithms are:

- The number of data per equation is d. We suppose d has the same value for each equation.
- The number of endogenous variables in the system, N. We suppose it coincides with the number of equations in the systems. Each endogenous variable is the main one in an equation.
- K is the number of exogenous variables in the system.
- The number of endogenous variables in equation i is n_i ($N < \sum_{i=1}^{N} n_i \leq N^2$).
- The number of exogenous variables in equation i is k_i ($\sum_{i=1}^{N} k_i \geq K$).

There are three types of equations: non identified ($n_i - 1 > K - k_i$), with no solution; exactly identified ($n_i - 1 = K - k_i$), for which it is possible to use ILS or 2SLS; and over identified ($n_i - 1 < K - k_i$), where it is necessary to use 2SLS.

2.1 Estimation by Ordinary Least Squares

The technique of Ordinary Least Squares (OLS) applies to the multiple regression model. This algorithm is used as the basis in other methods for Simultaneous Equation Models. There is only one equation, with one dependent variable, Y, n independent variables $(X_1, \ldots, X_n)$, and one random variable u.

$$Y_t = \beta_1 X_{1,t} + \ldots + \beta_n X_{n,t} + u_t \tag{6}$$

or in matrix form:

$$Y = X\beta + u \tag{7}$$

with:

$$X = \begin{pmatrix} X_{1,1} & \dots & X_{n,1} \\ & \dots & \\ X_{1,N} & \dots & X_{n,N} \end{pmatrix} \ , \ Y = \begin{pmatrix} Y_1 \\ \dots \\ Y_N \end{pmatrix} \ , \ \beta = \begin{pmatrix} \beta_1 \\ \dots \\ \beta_n \end{pmatrix} \ , \ u = \begin{pmatrix} u_1 \\ \dots \\ u_N \end{pmatrix} \tag{8}$$

The computation in OLS consists of $(X^tX)^{-1}X^tY$. The resulting matrix is the estimator of β ($\hat{\beta}$), and it is common to successive calls to OLS in different steps of ILS and 2SLS. A scheme of the algorithm is shown in figure 1. The scheme calls to multiplication and inverse routines, which are routines in the basic libraries BLAS [11] and LAPACK [12]. Thus, efficient parallel versions of the algorithm can be easily obtained by calling to BLAS or PBLAS.

The algorithm uses the funtions $CreateMatrix(n, m)$, which creates a new matrix with $n \times m$ dimension and $Copy\_Column$(colum in matrix,variable in equation) to copy in a matrix the variable Main_Endogen_Equation$_i$, which is the main endogenous variable in the i-th equation.

LAPACK and BLAS libraries are used to multiply of two matrices and for the inverse. Multiplication is performed using *dgemm*, and the inverse using *dgesv*. A portable and easily parallelizable program is obtained simply by changing *dgemm* for *pdgemm* and *dgesv* for *pdgesv* from ScaLAPACK library [13].

Since the value of $(X^tX)^{-1}X^t$ is common to each call to OLS, it is calculated only once, and it is used in successive calls to OLS. The algorithm uses the parameter *optX* to know whether it is the first call to OLS (then $(X^tX)^{-1}X^t$ is calculated and is returned in $Xaux$). Otherwise, the value is got from the matrix $Xaux$.

In some cases, the value of $\hat{\beta}$ is used and in others the value of $X\hat{\beta}$ (an estimated value of Y). For this reason OLS uses the parameter *option* to return the requested value.

Considering the cost of a matrix multiplication is $2n^3$ and the cost of the inverse is $\frac{5}{3}n^3$ (with n the matrix size), the cost of a call to OLS where the matrix is computed is:

$$T_{OLS1}(d, K) = \frac{5}{3}K^3 + 4K^2d + 4Kd \tag{9}$$

```
Mendo = CreateMatrix(d, 1)
Copy_Column(Mendo, 1, Main_Endogen_Equation_i)
IF optX == 0
        Mexog = CreateMatrix(d, k)
        FOR i = 1 TO N //for each equation
                FOR j = 1 TO k_i //for each exogenous variable in the equation
                        Copy_Column(Mexog, j, Exogen_j_Equation_i)
                NEXT j
        NEXT i
        T = Trasp (Mexog)
        m1= Mult (T, Mexog)
        in = Inverse (m1)
        m2 = Mult (in, T)
        Xaux = m2
ELSE
        m2 = Xaux
END_IF
sol = Mult (m2,Mendo)
m3 = Mult (Mexog, sol)
r2 = R (Mendo, m3)
IF option = TRUE
        solution = m3
ELSE
        solution = sol
END_IF
RETURN(solution, r2)
```

Fig. 1. OLS scheme

and

$$T_{OLS2}(d, K) = 4Kd \tag{10}$$

when the matrix is not computed.

2.2 Estimation by Indirect Least Squares

The ILS technique needs the equation to be exactly identified, which means the values of B and Γ can be univocally obtained from those of Π in equation 4.

The technique estimates the values of Π by OLS, and from Π obtains the values of the structural form of the equation it is solving. Thus, ILS solves the equations systems $-B_i\Pi = \Gamma_i$, with B_i being the row of B corresponding

to the equation which is being solved, and Π is common to all the equations. The function returns three vectors: two containing the estimated coefficients of endogenous and exogenous variables, and the other the correlation values obtained by OLS. Figure 2 shows a scheme of the computation in each step of the algorithm. In the first step, matrix Π is obtained and it is used in successive steps. In each step the equation system to be solved is formed, and a call is made to solve equation i. Figure 3 shows the scheme of the solution of one equations system. Again, the computations are based on multiplication and inverse of matrices, and BLAS and PBLAS can be used extensively.

```
IF optPi == 0
        j = 0
        Mpi = CreateMatrix(N,K)
        Mr2 = CreateMatrix(1,N)
        FOR i=1 TO N
                (m, r)= OLS(sis,Equation_i,Xaux,j)
                Include(Mr2,r)
                j++
                Include_VectorRow(m,Mpi)
        NEXT i
        Mpi = −Mpi
        Pi = Mpi
ELSE
        Mpi = Pi
END_IF
Mcoefex = CreateMatrix(1,K)
Mcoefend = CreateMatrix(1,N)
Mark_Endogen(ecu,Mcoefend)
Mark_Exogen(ecu,Mcoefex)
Solution_ILS(Mcoefex, Mcoefend, Mpi)
RETURN(Mcoefend, Mcoefex, Mr2)
```

Fig. 2. Scheme of the first part of ILS: computation of Π

Since all the equations solved by ILS share Π, it is calculated only in the first call to ILS. To know if it is necessary to calculate Π the parameter $optPi$ is used (the value of Π is stored in the matrix Pi and used in successive iterations).

The algorithm uses the functions $Include$ and $Include\_VectorRow$ to store a number or a vector (row vector) in a matrix. The function $Mark\_Endogen(equation, vectorRow)$ is used to put a signal in a row vector in the place where the equation has an endogenous variable, and the rest of the values in the vector are zero. For example, if the first equation in the system has the third and the fourth endogenous variables of the system, the vector will have a signal in the third and the fourth places and zeros in the rest.

```
IF (More incognites than equations)
        RETURN (error)
Mdep = CreateMatrix(Num_Zeros_vf1,1)
Mindep = CreateMatrix(Num_Zeros_vf1, Num_Incognites_vf2)
FOR i = 1 TO Colums(vf1)
        IF vf1_i = 0
                FOR j = 1 TO Colums(vf2)
                        IF vf2_j = -1
                                Include( Mdep, M_ji )
                        END_IF
                        IF vf2_j = unknow
                                Include( Mindep, M_ji )
                        END_IF
                NEXT j
        END_IF
NEXT i
IF (Invertible((Mindep))=FALSE)
        RETURN (error)
END_IF
in = Inverse ( Mindep)
Msol = Mult (in, Mdep)
Complete_Vector (vf2, Msol)
vf1=Mult(vf2, M)
RETURN(vf1, vf2)
```

Fig. 3. Scheme of the second part of ILS: formation and solution of the equations system

The same idea is used in $Mark\_Exogen(equation, vectorRow)$. The above vectors will be used to construct a system and to solve it. The variables in the system will be the coefficients of the equation and they will be calculated in $Solutions\_ILS$. After $Solutions\_ILS$, the value of the coefficients of the endogenous and exogenous variables of the equations will be in vectors $Mcoefend$ and $Mcoefex$.

The $Solutions\_ILS$ algorithm receives a matrix (with real numbers) and two row vectors (with numbers and unknows parameters). The conditions are: the first vector ($vf1$) has unknown quantities and zeros, and the second vector ($vf2$) has unknown quantities, zeros and one -1 (in an equation the main endogenous variable has coefficient -1). The number of unknown quantities coincides with the number of columns in $vf1$ (this property is used to check if the equation is identified). The number of zeros in $vf1$ is equal to the number of unknown quantities of the $vf2$ vector (the equation is exactly identified, and the number of zeros in $vf1$ is $K - k_i$). The multiplication of $vf2$ by the matrix is equal to $vf1$, and it is possible to calculate the unknown quantities to satisfy these conditions.

The $Solutions\_ILS$ algorithm constructs a system with the unknown quantities of $vf2$, and solves it. This is possible because the number of zeros in $vf1$ (and the number of equations) is the same as the number of unknown quantities in $vf2$. Subsequently, $vf1$ is calculated by the multiplication of $vf2$ by the matrix**.** The algorithm uses the function $Includes(matrix, number)$ to put a number into a matrix in the next position (starting in (1,1) from left to right), and $CompleteVector(vector1, vector2)$ puts the values of $vector2$ in the unknown quantities of $vector1$.

As in OLS, we have different costs when matrix Π is calculated from when it has been previously calculated. To obtain Π, OLS is called N times, and the cost is $T_{OLS1}(d, K) + (N-1)T_{OLS2}(d, K)$. After that, the coefficients of the equation are obtained. The number of unknowns is $n_i - 1$, and to apply ILS it is necessary for the equation to be exactly identified, and $n_i - 1 + k_i = K$. The cost of solving the system (using Solutions_ILS) is $2(K-k_i)^2 + T_{inverse}(K-k_i)$, and the cost to calculated vf1 is $2NK$ and the costs are:

$$T_{ILS1}(N, d, K, k_i) =$$
$$= T_{OLS1}(d, K) + (N-1)T_{OLS2}(d, K) + 2(K-k_i)^2 + \frac{5}{3}(K-k_i)^3 + 2NK =$$
$$= \frac{5}{3}K^3 + 4K^2d + 4Kd + (N-1)4Kd + 2(K-k_i)^2 + \frac{5}{3}(K-k_i)^3 + 2NK =$$
$$= \frac{5}{3}K^3 + 4K^2d + 4NKd + 2(K-k_i)^2 + \frac{5}{3}(K-k_i)^3 + 2NK \qquad (11)$$

$$T_{ILS2}(N, d, K, k_i) = 2(K-k_i)^2 + \frac{5}{3}(K-k_i)^3 + 2NK \qquad (12)$$

2.3 Estimation by Two-stage Least Squares

With 2SLS it is not necessary for the equation to be exactly identified.

In an equation the problem of the correlation between random and endogenous variables is avoided by substituting the original variable by a new variable called proxy. The variable is obtained by applying OLS to the predetermined variables in the system. Once all the endogenous variables have been substituted, OLS is applied to the equation. The main problem of 2SLS is that the calculated proxy may not be a good approximation of the original variable. Figure 4 shows the scheme of 2SLS. Each variable is substituted by one proxy by calls to OLS. After that, a call to an especial OLS is made. The computation in the especial OLS is again made basically through matrix multiplications and inverses. In this case the cost is higher than in ILS.

$MRproxy$ = CreateMatrix(1,$n_i - 1$)
FOR j=2 TO n_i
    ($proxy, r$) = OLS ($sis, ecu, Xaux, j - 2$)
    Include ($Mrproxy, r$)
    Change ($proxy, Datos\_n_j$)
NEXT j
($sol, r2$) = SpecialOLS (sis, ecu)
$solution$ = Trasp(sol)
FOR j=2 TO n_i
    Recuperate ($Datos\_n_j$)
NEXT j
RETURN ($solution, Mrproxy, r2$)

Fig. 4. 2SLS scheme

$Mexog$ = CreateMatrix(d, $n_i + k_i - 1$)
FOR j = 2 TO $n_i + k_i$
    Copiar_Columna($Mexog$, j, Variable$_j$)
NEXT j
$Mendo$ = CreateMatrix(d, 1)
Copy_Column($Mendo$, 1, Main_Endogenous_$ecua$)
T = Trasp ($Mexog$)
$m1$= Mult (T, $Mexog$)
$m2$ = Mult (T, $Mendo$)
in = Inversa ($m1$)
sol = Mult (in, $m2$)
$m3$ = Mult ($Mexog$, sol)
$r2$ = R ($Mendo$, $m3$)
RETURN(sol, $r2$)

Fig. 5. Scheme of the special version of OLS used in 2SLS

In 2SLS, first the proxys are calculated ($n_i - 1$ `proxys` are calculated, with n_i the number of endogenous variables in the equation), and the cost is $T_{OLS1}(d, K) + (n_i - 2)T_{OLS2}(d, K)$.Then OLS is applied using these proxys (the routine is called special OLS).

The algorithm of 2SLS uses the function *Change* to change the values of the jth endogenous variable for the proxy's values. And at the end the algorithm uses *Recuperate* to put the original values of the variables in the same place.

The special OLS uses matrix X used in $(X^tX)^{-1}X^tY$, which is formed by the endogenous variables (they now have the value of the proxys), without the main endogenous, and the predetermined variables. Thus, matrix X has size $(n_i - 1 + k_i) \times (n_i - 1 + k_i)$. To solve the equation, $n_i - 1 + k_i \leq K$. When the equation is exactly identified the two values coincide, and OLS and special OLS have the same cost, although they are applied to different data. The cost is:

$$T_{2SLS}(N,d,K,n_i) = 2T_{OLS1}(d,K) + (n_i-2)T_{OLS2}(d,K) = \\ 8K^2d + 8Kd + \frac{10}{3}K^3 + (n_i-2)4Kd \quad (13)$$

It has been supposed the worst time possible (the cost of OLS1 and special OLS is the same).

2.4 Determination coefficient

The determination coefficient, R^2, gives information on how good the adjustment to the theoretical model has been. When R^2 approaches 0 the model is well adjusted. R^2 is computed each time OLS is applied. Its value is:

$$R^2 = 1 - \frac{\sum_{i=1}^{N}\left(Y_i - \hat{Y}_i\right)^2}{\sum_{i=1}^{N}\left(Y_i - \bar{Y}_i\right)^2} \quad (14)$$

In ILS a value of R^2 will be obtained for each call to OLS. In 2SLS values of R^2 are provided for each proxy and in special OLS.

2.5 Sequential tool

A tool has been developed for sequential systems. The tool has been made in Excel and is conceived to be used by researchers in different fields (econometics, psychology, biology, computer sciences, etc) who may not be expert in computation but who could need to solve not very large Simultaneous Equations Models, with reduced necessities of computation. For more expert users, routines to be used on top of BLAS and LAPACK have been developed in such a way that they can be included in programs to solve Simultaneous Equations Models in different ways, depending on the experience of the user. For users with high computational necessities it could be convenient to have parallel versions of the routines. These versions are explained in the next section.

3 Solution of Simultaneous Equations Models on parallel systems

Parallel versions for Shared Memory (OpenMP) and Distributed Memory (MPI) have been developed. The loops in the programs have different costs

in different iterations, and some results obtained in one iteration can be used in successive iterations. This makes it difficult to obtain parallel versions perfectly balanced.

Parallelization can be made at different levels: in the basic matrix operations using PBLAS, dividing the work in the loops between the threads in the system, etc. The parallelization has been made each time at the highest possible level.

3.1 Shared Memory Algorithms

3.1.1 Paralelization of ILS

The solution of only one equation by ILS has been parallelized, but in the solution of one system all the identified equations must be solved.

ILS has two parts: a loop where matrix Π is obtained, and the solution of ILS, which also consists of a loop where matrix Π is used. The analysis of the cost of the two parts shows the time of the second part is very low in comparison with that of the computation of Π. Furthermore, the first iteration of the loop has a cost similar to that of the rest of the iterations. In each iteration a call to OLS is made, but in the first iteration OLS returns a matrix $((X^tX)^{-1}X^t)$ which will be used in successive steps.

The parallelization of the first iteration has been entrusted to the basic functions (matrix multiplication). LAPACK is used to obtain the inverse. The works in successive iterations are distributed between the different threads, using a `parallel for`. The solution of ILS is parallelized in the basic operations. A scheme of the algorithm is shown in figure 6.

The same idea is applied to solve a complete system. The computation of matrix Π is made only once when ILS is applied to solve all the equations in the system.

If p processors are available, when only one equation is solved the cost is:

```
IF optPi == 0
        Mpi = CreateMatrix(N,K)
        Mr2 = CreateMatrix(1,N)
        //first iteration
        (m, r)= OLS(sis,Equation_1,Xaux,0)
        Include(Mr2,r)
        Include_RowVector(m,Mpi)
        //successive iterations
        PARALLEL FOR i
        FOR i=2 TO N //for each equation
                (m, r)= OLS(sis,Equation_i,Xaux,1)
                Include(Mr2,r)
                Include_RowVector(m,Mpi)
        NEXT i
        END_PARALLEL_FOR
        Mpi = -Mpi
        Pi = Mpi
ELSE
        Mpi = Pi
END_IF
Mcoefex = CreateMatrix(1,K)
Mcoefend = CreateMatrix(1,N)
Mark_Endogenous(ecu,Mcoefend)
Mark_Exogenous(ecu,Mcoefex)
Solution_ILS(Mcoefex, Mcoefend, Mpi)
RETURN(Mcoefend, Mcoefex, Mr2)
```

Fig. 6. Scheme of the parallel ILS

$$T_{ILS1,p}(N,d,K,n_i,p) =$$
$$T_{OLS1,p}(d,K,p) + \frac{(N-1)}{p}T_{OLS2,p}(d,K,p) + \frac{2(K-k_i)^2}{p} + \frac{5}{3}(K-k_i)^3 + \frac{2NK}{p} =$$
$$= \frac{5}{3}K^3 + \frac{4K^2d}{p} + \frac{4Kd}{p} + \frac{N-1}{p}4Kd + \frac{2(K-k_i)^2}{p} + \frac{5}{3}(K-k_i)^3 + \frac{2NK}{p} \quad (15)$$

and when C are solved with ILS, the cost is:

$$T = T_{ILS1,p}(N,d,K,n_i,p) + \frac{(C-1)}{p}T_{ILS2}(N,d,K,n_i) =$$
$$\frac{4K^2d}{p} + \frac{2KdN}{p} + \frac{2K^2}{p} + \frac{10}{3}K^3 + \frac{(C-1)}{p}\left(2K^2 + \frac{5}{3}K^3\right) \quad (16)$$

The solution of only one equation by 2SLS has been parallelized. 2SLS is divided in two parts: computation of the `proxys`, and the application of the special OLS routine. In the loop to calculate the `proxys` not all the iterations have the same cost, and this must be borne in mind to obtain an efficient parallel algorithm. The difference in the costs of the computations is caused by the fact that some results of OLS are used in successive iterations. Using the same idea as in ILS the first iteration is parallelized at a low level (matrix multiplications), obtaining a matrix $Xaux$ which will be used in the successive iterations. Once the first iteration has been performed, the other iterations are divided between the available threads with a `parallel for`. When the loop has been performed, the parallelization of the special OLS is made at a low level. A scheme of the algorithm is shown in figure 7.

$MRproxy$ = CreateMatrix(1,$n_i - 1$)
//first iteration
($proxy, r$) = OLS ($sis, ecu, Xaux, 0$)
Include ($Mrproxy, r$)
Change ($proxy, Data\_n_2$)
//other iterations
PARALLEL FOR j
FOR j=3 TO n_i
    ($proxy, r$) = OLS ($sis, ecu, Xaux, 1$)
    Include ($Mrproxy, r$)
    Change ($proxy, Data\_n_j$)
NEXT j
END_PARALLEL_FOR
($sol, r2$) = EspOLS (sis, ecu)
$solution$ = Trasp(sol)
Recuperate ($Datos\_n_j$)
RETURN ($solution, Mrproxy, r2$)

Fig. 7. Scheme of the parallel 2SLS

No information is shared between the different equations in the system, and the cost is obtained by multiplying the number of equations to solve by the cost of the solution of only one equation, which is:

$$T_{2SLS,p}(N, d, K, n_i, p) = 2T_{OLS1,p}(d, K, p) + \frac{(n_i - 2)}{p} T_{OLS2}(d, K) = \frac{8K^2d}{p} + \frac{8Kd}{p} + \frac{10}{3}K^3 + \frac{(n_i - 2)}{p} 2Kd \quad (17)$$

The previous explanation is valid for the solution of an individual equation, but to solve a complete Simultaneous Equations System it is better to parallelize at a higher level. In a system there can appear non identified, overidentified and exactly identified equations. For non identified equations no estimation can be done, for overidentified equations it is necessary to apply 2SLS, and for exactly identified equations ILS or 2SLS can be applied.

In a large system, equations of all the types will appear, and ILS and 2SLS are applied. In 2SLS, each thread works in the solution of the equations without sharing information with the other threads. In ILS, matrix Π is common to all the equations. The first iteration is solved as described, then the loop to solve the other equations is solved in parallel, and all the threads use the matrix computed.

3.2 Distributed Memory Algorithms

In distributed memory the data are initially in processor P_0, which distributes the data between all the processors in the system. This distribution is carried out with a `broadcast`. After receiving the data, each processor solves the equations which it has assigned. Finally, the solutions of the individual equations are sent back to processor P_0, using point-to-point communications.

The data to transfer in the initial broadcast are: data of the system and endogenous and predetermined variables $(4 + N + K)$, data of the equation and endogenous and predetermined variables in it $(4 + n_i + k_i)$, data of the sample corresponding to the variables, which in each equation are the data of the main endogenous and the predetermined $((k_i + 1)d)$. The sending of the data is done via a broadcast, and the cost is:

$$t_{s_b} + t_{w_b}\left(4 + N + K + \sum_{i=1}^{N}(4 + n_i + k_i + (k_i + 1)d)\right) \tag{18}$$

Where t_{s_b} is the start-up time and t_{w_b} is the word-sending time in a broadcast communication.

We consider C_1 equations have been solved by ILS and C_2 by 2SLS (the rest of the equations are non identified). Processor P_0 receives information from each one of the other $p - 1$ processors, and the cost of the reception of the

resuls will be: $(p-1)t_s + t_w \sum_{j=1}^{p} s_j$, where not all the processors send the same information. The number of data to send is $s = \sum_{j=1}^{p} s_j$.Where t_s is the start-up time and t_w is the word-sending time in a point to point communication.

When a processor applies ILS, Π is calculated only once, and the data to send are the values of the determination coefficients of each row of Π, and the coefficients of the endogenous and exogenous variables of the equation (except the main endogenous), the number of equation and the key of the method that it has been used. In total, $N + n_i + k_i + 1$ data.

When 2SLS is applied, the coefficients of the endogenous (except the main endogenous) and the predetermined variables of the equation and the determination coefficients computed for each proxy are sent $(2n_i + k_i + 1)$. When a equation is not solve, two values are sending. Thus, the total size is:

$$s = N + \sum_{i=1}^{C_1}(N + n_i + k_i + 1) + \sum_{i=1}^{C_2}(2n_i + k_i + 1) + (N - C_1 - C_2)2 \quad (19)$$

4 Experimental results

Experimental results have been obtained in three systems:

- An HP160 with four nodes, each node with four processors. The four processors in a node share a local memory, and OpenMP can be used inside each node. Algorithms for distributed memory can be used, and processors from the same or different nodes can run. Because the system is used by a large number of researchers, only four processors have been used in the experiments.
- A cluster of ten biprocessors Intel Xeon 2 with SCI connection. The MPI algorithms have been tested in this cluster.
- The MPI algorithms have also been tested in Marenostrum. Marenostrum compromises 2282 JS20 compute nodes and 42 p615 servers. Each node has two processors at 2.2 Ghz running Linux operating system with 4 GB of memory RAM and 40 GB local disk storage.

As has been shown, the theoretical cost depends on a large number of parameters. To simplify the experiments, they have been made with different number of endogenous variables (N), and the number of exogenous variables has been taken as 40% of N. The simultaneous equations systems used in the experiments have been created randomly.

Figures 8 and 9 show the speed-up achieved in HP160 with the shared memory versions of ILS and 2SLS when only one equation is solved and when a system of equations is solved. In all the cases, the speed-up obtained is satisfactory, and it is higher when more computation is made.

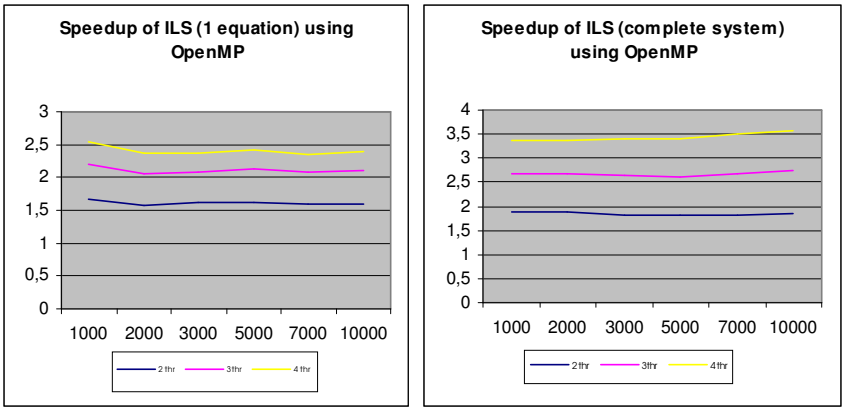

Fig. 8. Speed-up obtained with the shared memory version of ILS in HP160, for different problem sizes and numbers of threads, when solving one equation (left) and a system of equations (right)

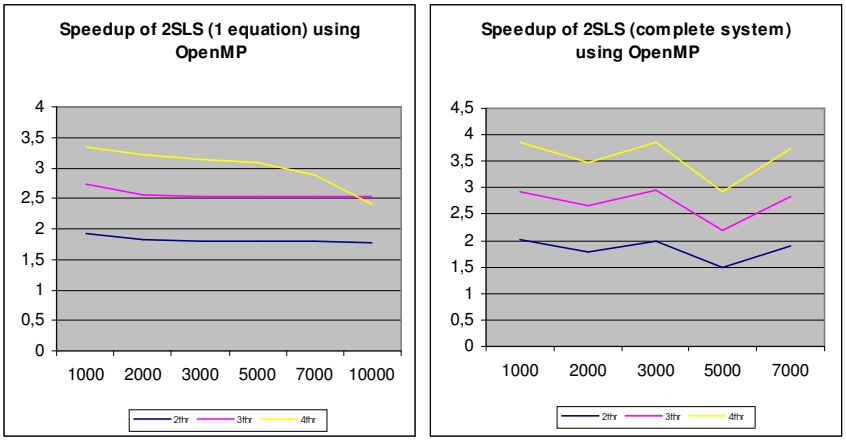

Fig. 9. Speed-up obtained with the shared memory version of ILS in HP160, for different problem sizes and numbers of threads, when solving one equation (left) and a system of equations (right)

The distributed memory version has been run in the cluster of processors and in Marenostrum to analyze the scalability of the algorithms. The results in the clusters are compared in tables 1 and 2. The tables show the execution time and the speed-up obtained when varying the number of processors and the

size:	500		1000		2000	
proc.	time	Sp	time	Sp	time	Sp
1	18.16		346.77		5320.91	
3	6.64	2.73	117.20	2.96	1823.38	2.92
5	4.28	4.24	71.32	4.86	1065.92	4.99
10	3.74	4.86	36.84	9.41	559.36	9.51
18	2.23	8.14	24.55	14.13	310.05	17.16

Table 1
Execution time (in seconds) and speed-up of the message-passing version of ILS in a cluster of processors, when varying the number of endogenous variables and the number of processors

size:	500		1000	
proc.	time	Sp	time	Sp
1	67.38		1238.07	
5	13.92	4.84	251.87	4.92
10	8.52	7.91	125.47	9.87

Table 2
Execution time (in seconds) and speed-up of the message-passing version of 2SLS in a cluster of processors, when varying the number of endogenous variables and the number of processors

number of endogenous, with ILS and 2SLS and when a system of equations is solved.

The same experiments have beeen carried out in Marenostrum. Tables 3 and 4 show the execution time and the speed-up obtained when varying the number of processors and the number of endogenous variables (the sample size is always 100), with ILS and 2SLS and when a system of equations is solved. Because the computations performed by the processors in the systems are independent, the scalability is satisfactory, and when the number of processors increases a speed-up close to the optimum can be obtained when the size of the problem increases.

5 Conclusions and Future Works

Algorithms for the solution of Simultaneous Equations Models with Ordinary Least Squares, Indirect Least Squares and Two-stage Least Squares methods have been developed for sequential, shared memory and message-passing sys-

size:	1000		2000		3000		4000		5000	
proc.	time	Sp	time	Sp	time	Sp	time	Sp	time	Sp
1	54.36		1473.26		6680.22		18117.75		43289.03	
2	27.74	1.96	763.66	1.93	3387.13	1.97	9358.91	1.94	21363.3	2.02
4	16.24	3.35	432.09	3.41	1970.5	3.39	5373.4	3.37	12332.45	3.51
8	8.18	6.65	217.49	6.77	996.72	6.70	2622.39	6.9	6221.59	6.95
16	4.48	12.14	110.58	13.32	492.71	13.56	1338.99	13.53	5845.15	7.4
24	3.22	16.89	74.59	19.75	332.17	20.11	900.35	20.12	2079.94	20.81
32	2.62	20.73	57.57	25.59	248.71	26.86	675.57	26.82	1569.7	27.58
64	1.19	45.7	30.66	48.05	130.69	51.12	349.07	51.9	827.73	52.3

Table 3
Execution time (in seconds) and speed-up of the message-passing version of ILS in Marenostrum, when varying the number of endogenous variables and the number of processors

size:	1000		2000		3000		4000	
proc.	time	Sp	time	Sp	time	Sp	time	Sp
1	377,68		4995,44		20037,01		58256,46	
2	188,72	2	2505,67	1,99	9987,18	2	29324,64	1,99
4	106,06	3,56	1415,61	3,53	5778,24	3,47	16844,37	3,46
8	53,23	7,1	726,41	6,88	2983,46	6,72	8289,87	7,03
16	26,48	14,26	379,12	13,18	1494,39	13,41	4109,45	14,17
24	17,83	21,19	245,18	20,37	1013,14	19,78	2811,31	20,72
32	13,76	27,45	180,26	27,71	720,24	27,82	2091,05	27,86
64	7,05	53,61	94,89	52,65	377,44	53,09	1098,18	53,05

Table 4
Execution time (in seconds) and speed-up of the message-passing version of 2SLS in Marenostrum, when varying the number of endogenous variables and the number of processors

tems. Simultaneous Equations Models appear in different scientific fields, and because the size in the applications varies greatly, it is interesting to have tools for the solution of small and large systems. A tool for sequential processors with Excel has been developed. For the solution of bigger systems, routines in LAPACK style have been developed for shared and distributed memory environments. The parallel versions of the algorithms have proved to be scalable, and so appropriate to afford the solution of huge systems like those appearing in the simulation of national or world economy.

After additional tests with real data and in more computational systems, the tools will be provided to the scientific community on a web site. More methods are planned to be developed and included in the tools.

Acknowledgment

This work has been funded in part by the Spanish MCYT and FEDER under Grant TIC2003-08238-C02-02.

Systems of the Servicio de Apoyo a la Investigación Tecnológica of the Polytechnic University of Cartagena, Barcelona Supercomputing Center and of the High Performance Computing and Networking Group of the Polytechnic University of Valencia have been used for the experiments.

References

[1] Harchol-Balter, M., Black, P. E. 1993. Queueing Analysis of Oblivious Packet-Routing Networks, SODA: ACM-SIAM Symposium on Discrete Algorithms.

[2] Henry, R., Lu, I., Beightol, L. and Eckberg, D. 1998. Interactions between CO2 Chemoreflexes and Arterial Baroreflexes, Am. Journal of Physiology, 274(Heart Circ. Physiol. 43), H2177-H2187.

[3] Kontoghiorghes, E. J. 2000. Parallel Algorithms for Linear Models: Numerical Methods and Estimation Problems, Kluwer Academic Publishers.

[4] Gonschorek, A., Lu, I., Halliwill, J., Beightol, J.A., Taylor, J.A., Painter, H. and Eckberg, D. 2001. Influence of Respiratory Motor Neurone Activity on Human Autonomic and Haemodynamic Rhythms, Clinical Physiology, 21(3), 323-334.

[5] Greene, W. 1998. Econometric Analysis, third edition, Prentice Hall.

[6] Gujarati, D. 1995. Basic Econometrics, McGraw Hill.

[7] Lu, I. and Jones, S.P. 2004. An Investigation of the Effect of Cabin Pressure Altitude and Level of SaO2 on Passenger Comfort - A Simultaneous Equation Model, Boeing Technical Report Series, M&CT-TECH-04-019

[8] Lu, I., Peixoto, J. and Taam, W. 2003. A Simultaneous Equation Model for Air Traffic in the New York Area, The Seventh Air Transport Research Society World Conference Journal.

[9] http://www.doornik.com/

[10] http://www.eviews.com/

[11] http://www.netlib.org/blas/

[12] http://www.netlib.org/lapack/

[13] http://www.netlib.org/scalapack/

[14] http://www.sswr.org/papers2002/616.htm