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 for the Ordinary Least Squares, the Indirect Least Squares and the Two-stage Least Squares methods are developed. Algorithms for distributed memory are studied theoretically and experimentally. The algorithms make an extensive use of basic libraries like BLAS, LAPACK, and ScaLAPACK to obtain efficient and portable versions.

Key words: Simultaneous Equations 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 Equations Models have been used in econometrics [4,5], but recently they have begun to be used in other fields. Examples can be found in networks simulation [6], biological microsystems, psychology [22], medicine [3,7], and even in the study of the air traffic in New York [11] and of

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

**Departamento de Informática y Sistemas, Universidad de Murcia, 30100 Murcia, Spain

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

[1] Corresponding author

7 November 2007

the effect of cabin altitude and arterial oxygen saturation level on passenger comfort during a prolonged flight [10].

In some cases the models to solve are very large (for instance 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 [8]. 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 theoretically. These techniques are also developed for distributed memory (using MPI [17]). Additionally, they use basic libraries like BLAS [18], LAPACK [19] and ScaLAPACK [20] intensively to obtain efficient, portable versions of the algorithms.

In 1994, a company of econometric software, QMS (Quantitative Micro Software), began the development of software for this type of systems, with Eviews 1.0, and the last version is 6.0. Eviews includes linear regression techniques, solution of Simultaneous Equations Models, time series, and other econometric problems [16].

One free tool for econometric is Ox [15]. This software includes fewer problems than Eviews, although it does include 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 Equations Models. For example, the SPSS 12.0 [21] includes 2SLS.

All the above software is conceived to solve small systems. When a large system of Simultaneous Equations Models has to be solved, it is not possible to use these routines. Our software works in parallel and can solve systems with thousands of equations. The cost is reduced as much as possible and a lot of information is shared to reduce the memory.

The rest of the paper is organized as follows. In the second section the theoretical bases of Simultaneous Equations Models are stated. In sections 3 and 4 sequential and parallel algorithms are developed and studied. In the fifth section experimental results are presented. Conclusions and future work are in section 6.

2 Simultaneous Equations Models

Three types of variables appear in a Simultaneous Equations System:

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

The scheme of a system with N equations, N endogenous variables and K predetermined variables is:

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

where $Y_1, Y_2, \ldots, Y_N$ are endogenous variables, $X_1, X_2, \ldots, X_K$ are predetermined variables, and $u_1, u_2, \ldots, u_N$ are random variables.

Equation (1) can be represented in matrix form as:

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

with:

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

The problem lies in obtaining $\beta_{1,2}, \beta_{1,3}, \ldots, \beta_{N,N-1}, \gamma_{1,1}, \ldots, \gamma_{N,K}$ from a representative sample of the model.

The structural model (equation 2) can be expressed in reduced form:

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

with:

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

A number of variables will be used to analyze the algorithms. The system will be in structural form, and each equation will have 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, 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 one equation.
- The number of predetermined variables in the system, K.
- The number of endogenous variables in equation i, n_i.
- The number of predetermined variables in equation i, k_i.
- X is the predetermined matrix. It has $d \times K$ dimension and contains the data of predetermined variables.
- Y is the endogenous matrix. It has $d \times N$ dimension and contains the data of endogenous variables.

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. To study the kind of equation two conditions are evaluated: Order Condition and Range Condition [5].

3 Sequential Algorithms

In this section sequential algorithms for three techniques are studied. First, Ordinary Least Squares is studied (OLS). This algorithm is used as the basis for Simultaneous Equations Models in the other methods. Next, Indirect Least Squares (ILS) and Two Step Least Squares (2SLS) are studied.

An algorithm for the technique of Ordinary Least Squares (OLS) in a multiple regression model is developed. 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

$$Y = \begin{pmatrix} Y_1 \\ \ldots \\ Y_T \end{pmatrix} \ , \ X = \begin{pmatrix} X_{1,1} \ \ldots \ X_{n,1} \\ \ldots \\ X_{1,T} \ \ldots \ X_{n,T} \end{pmatrix} \ , \ \beta = \begin{pmatrix} \beta_1 \\ \ldots \\ \beta_n \end{pmatrix} \ , \ u = \begin{pmatrix} u_1 \\ \ldots \\ u_T \end{pmatrix} \tag{8}$$

where T is the sample size.

The computation in OLS consists of $(X^tX)^{-1}X^tY$. The resulting matrix is the estimator of β ($\hat{\beta}$). $\hat{Y} = X\hat{\beta}$ is calculated to obtain an estimation of Y.

Algorithm 1 shows a scheme of OLS. It is possible to use LAPACK and BLAS libraries here. In the experiments (section 5), *dgemm* has been used for multiplication, and *dgesv* for the inverse, and a portable program is obtained using the LAPACK library [19].

Algorithm 1 Scheme of OLS algorithm

1: Compute X^tX $\{\text{cost} \rightarrow 2K^2d\}$
2: Compute X^tY $\{\text{cost} \rightarrow 2NKd\}$
3: Compute $(X^tX)^{-1}(X^tY)$ $\{\text{cost} \rightarrow \frac{2}{3}K^3 + 2K^2N\}$
4: **if** estimation=**true then**
5: Compute $X(X^tX)^{-1}(X^tY)$ $\{\text{cost} \rightarrow 2NKd\}$
6: **end if**

The cost is shown in each line of the algorithm. To study the cost, only flops (floating point operations) are taken into account. The total cost of the OLS algorithm is obtained by adding together the costs of each line of the algorithm:

$$T_{OLS}(N, d, K, \delta_{est}) = \frac{2}{3}K^3 + 2K^2(d+N) + 2NKd + \delta_{est}2NKd \qquad (9)$$

where δ_{est} is 1 if *estimation* is requested in the algorithm, and 0 in the other case. The parameter *estimation* is true when the estimation of the endogenous variables is required.

3.2 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 calculated in equation 4.

The technique estimates the values of Π by OLS (using all the endogenous variables as dependent and all predetermined variables as independent), and from Π obtains the values of the structural form of the equation to be solved. Thus, ILS solves the equations systems $-B_i\Pi = \Gamma_i$, with B_i being the row of B, and Γ_i the row of Γ, corresponding to the equation which is being solved, and Π is common to all the equations.

Algorithm 2 shows a scheme of the computation when the only method used in the solution of the Simultaneous Equations Model is ILS. 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.

Algorithm 2 Scheme of ILS

```
1: Π^t = OLS(Y, X, estimation = false)
2: for i=1...N do
3:    if equation i is exactly identified then
4:       Solve −B_i Π = Γ_i
5:    end if
6: end for
```

Since a lot of equations are solved by ILS, and the equations share Π, this is calculated in line 1. In each equation, B_i and Γ_i are obtained by solving $-B_i\Pi = \Gamma_i$ (line 4). The number of unknown variables in Γ_i is the number of predetermined ones in the equation (k_i), and in B_i it is the number of endogenous ones in the equation minus one ($n_i - 1$), because the main endogenous has coefficient 1. The number of equations is the number of columns of Π (K) and, since the equation is exactly identified ($K = n_i + k_i - 1$), it is possible to create and solve an equations system.

To obtain Π, OLS is called using the endogenous matrix as dependent (with

$d \times N$ dimension) and the predetermined matrix as independent (with $d \times K$ dimension), and the cost is $T_{OLS}(N, d, K, \delta_{est} = 0)$. The cost of solving $-B_i\Pi = \Gamma_i$ (line 4) is $\frac{2}{3}(K - k_i)^3 + 2(K - k_i)^2$ to calculate B_i, and $2Nk_i$ to calculate Γ_i. Therefore, the total cost of solving a system which has all the equations exactly identified by ILS is:

$$T_{ILS}(N, d, K) = T_{OLS}(N, d, K, \delta_{est} = 0) + \sum_{i=1}^{N} \left(\frac{2}{3}(K - k_i)^3 + 2(K - k_i)^2 + 2Nk_i \right) \tag{10}$$

Again, the computations are based on multiplication and inverse of matrices, and BLAS and LAPACK can be used extensively (*dgemm* and *dgesv* can be used on OLS and when solving $-B_i\Pi = \Gamma_i$).

3.3 Estimation by Two-stage Least Squares

In an equation, the problem of the correlation between random and endogenous variables is avoided by substituting the original variable for a new variable called *proxy*. After that, a call to OLS is made. The computation here is again made basically through matrix multiplications and inverses. In this case the cost is higher than in ILS, where an exactly identified equation is solved.

A *proxy* variable is calculated by making an estimation by OLS with all the predetermined variables of the system as independent variables, and the endogenous variable to be substituted as the dependent variable. When several equations are solved by 2SLS, it is possible that the same *proxy* is often used. For this reason, the *proxy* variables must be calculated before starting to solve equations.

Algorithm 3 Scheme of 2SLS

1: $\hat{Y} = OLS(Y, X, estimation =$**true**$)$
2: **for** i=1...N **do**
3: OLS(y_i,X_e,estimation=**false**)
4: **end for**

Algorithm 3 shows the scheme of 2SLS when solving a complete system. In 2SLS, the *proxys* are initially calculated (N *proxys* are calculated, and the cost is $T_{OLS}(N, d, K, \delta_{est} = 1)$). It could be that some of the N *proxys* are not used, but in parallel algorithms it is better to calculate all of them, since this can be done efficiently because the matrices X and Y are distributed in all the processors. For this reason, and given that we want to obtain a good parallel version, all *proxys* are calculated at the beginning (line 1).

To compute $(X_e^t X_e)^{-1} X_e^t y_i$, OLS uses matrix X_e, which is formed by the endogenous variables (they now have the values of the *proxys*) without the main endogenous and the predetermined variables of the i-th equation. Thus, matrix X_e has size $d \times (n_i - 1 + k_i)$. y_i is the main endogenous variable of the equation.

To solve the equation by 2SLS, it is necessary that $n_i - 1 + k_i \leq K$ (over identified or exactly identified equation). When the equation is exactly identified, the solutions of the ILS and 2SLS algorithms coincide.

The total cost is:

$$\begin{aligned} T_{2SLS}(N, d, K, n_i) = T_{OLS}(N, d, K, \delta_{est} = 1) + \\ + \sum_{i=1}^{N} (T_{OLS}(1, d, k_i + n_i - 1, \delta_{est} = 0)) \end{aligned} \tag{11}$$

Equation 11 shows that the cost of 2SLS depends heavily on d. The parameter affects all parts and d has higher effect than in equation 10. Experimental results are in tables 4 and 7 and show the relation between the total cost and parameter d.

3.4 A general algorithm

A general algorithm to solve a system of Simultaneous Equations is shown. In a normal system there will be non identified, over identified and exactly identified equations.

Because the solution of a system involves using ILS and 2SLS (2SLS can be used in an exactly identified equation, but ILS is faster than 2SLS and therefore is used), Π is calculated at the beginning of algorithm 4 (line 1) and is used in ILS. Then all the *proxy* variables are calculated by $\hat{Y} = X\Pi$ (line 2). Hence all the equations are solved.

Algorithm 4 Scheme of the general algorithm

```
1: Π^t = OLS(Y, X,estimation=false)
2: Ŷ = XΠ^t
3: for i=1...N do
4:    if equation i is exactly identified then
5:       Solve −B_i Π = Γ_i
6:    else if equation i is over identified then
7:       OLS(y_i,X_e,estimation=false)
8:    else
9:       nothing {equation i is not identified}
10:   end if
11: end for
```

4 Parallel algorithms

Parallel versions for distributed memory have been developed. Parallelization can be made at different levels: in the basic matrix operations using PBLAS and ScaLAPACK [20], dividing the work in the loops among the processors in the system, etc. The parallelization has been made each time at the highest possible level.

In distributed memory, it is assumed that X and Y are distributed between all processors in ScaLAPACK style. When the algorithm finishes the solution is also distributed.

In the theoretical study, the typical model of the communications is used [9], it is assumed that t_s is the start-up time in point-to-point communications and t_{s_b} in broadcast, and t_w and t_{w_b} are the word-sending time in a point-to-point and a broadcast communication.

In some parts of the algorithms, a matrix is distributed in all the processors in ScaLAPACK style, and it is necessary to send the complete matrix to all processors. MPI library can be used here (MPI_ Alltoall routine). The cost of this routine is assumed as $T_{A2A}(T,p) = (p-1)t_s + Tt_w$, where T is the total data to send and p is the number of processors. t_s and t_w depend on the hardware system. The cost of $T_{A2A}(T,p)$ depends lineally on T and p.

4.1 Parallel Ordinary Least Squares

Algorithm 5 shows a scheme of the parallel OLS algorithm. The multiplications and the inverse are carried out in parallel using all the processors. It is possible

to perform these functions using ScaLAPACK and PBLAS libraries. In the experiments (section 5) *pdgemm* has been used to perform the multiplications, and *pdgesv* to compute the inverse. The use of ScaLAPACK allows us to obtain a portable routine.

Algorithm 5 Scheme of the parallel OLS algorithm

1: Compute X^tX {Parallel Multiplications}
2: Compute X^tY {Parallel Multiplications}
3: Compute $(X^tX)^{-1}(X^tY)$ {Parallel Inverse}
4: **if** estimation=**true then**
5: Compute $X(X^tX)^{-1}(X^tY)$ {Parallel Multiplications}
6: **end if**

At the end of the algorithm, the solution is distributed in the system. Thus, the cost of the parallel OLS algorithm when using p processors is the cost in equation 9 divided by p, plus a communication cost of lower order.

4.2 Parallel Indirect Least Squares

Algorithm 6 shows a scheme of the parallel version of ILS. In the first step, matrix Π is obtained by calling parallel OLS (algorithm 5), 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. Because each processor solves several equations, each processor needs to access Π, B_i and Γ_i, but it does not need the sample data (only the system's structure).

If B and Γ are distributed at the beginning of the algorithm, all the processors have the complete structure of the system and it is not necessary to send data from other processors.

If B and Γ are in processor P_0, because each processor only needs a part of B and Γ ($\frac{N}{p}$ rows of each matrix), P_0 sends only this part, with a cost of $(p-1)\left(t_s + t_w \frac{N^2+NK}{p}\right)$.

Since many neighboring equations have a similar structure (number of endogenous and predetermined variables), equations are not assigned to processors by adjacent blocks. The equations are assigned to the processors cyclically.

Algorithm 6 Scheme of the parallel ILS algorithm

```
1: Π^t = OLS_p(Y, X, estimation = false)
2: Distribute Π to all the processors
3: IN PARALLEL Each processor q DO
4: for j=1...N/p do
5:    i = q + (j − 1)p
6:    if equation i is exactly identified then
7:       Solve −B_i Π = Γ_i
8:    end if
9: end for
10: END PARALLEL
```

To obtain Π, OLS is called using the endogenous matrix as dependent (with size $d \times N$) and the predetermined matrix as independent (with size $d \times K$), and the cost is $T_{OLS}(N, d, K, \delta_{est} = 0)$. When OLS ends, Π is stored in ScaLAPACK style and it is necessary to send the complete matrix to all the processors. The cost of sending Π is $T_{A2A}(KN, p)$.

The cost of solving $-B_i\Pi = \Gamma_i$ is $\frac{2}{3}(K - k_i)^3 + 2(K - k_i)^2$ to calculate B_i, and $2Nk_i$ to calculate Γ_i. Therefore, the cost when all equations in a system are solved by ILS is:

$$
\begin{aligned}
T_{ILS_p}(N, d, K) = T_{OLS_p}(N, d, K, \delta_{est} = 0) + T_{A2A}(KN, p) + \\
+ \max_{q=1\ldots p} \left\{ \sum_{j=1}^{\frac{N}{p}} \left(\tfrac{2}{3}(K - k_{q+(j-1)p})^3 + 2(K - k_{q+(j-1)p})^2 + 2Nk_{q+(j-1)p} \right) \right\} \approx \\
\approx T_{OLS_p}(N, d, K, \delta_{est} = 0) + T_{A2A}(KN, p) + \\
+ \sum_{i=1}^{\frac{N}{p}} \left(\tfrac{2}{3}(K - k_i)^3 + 2(K - k_i)^2 + 2Nk_i \right)
\end{aligned}
\tag{12}
$$

To simplify the problem it is assumed that the structure of all the equations is similar, and the cost of solving the first $\frac{N}{p}$ equations is similar to that of solving any other $\frac{N}{p}$ equations.

The cost will depend to a greater extent on K and N and to a lesser on d. Table 2 shows a large increase in the cost of the algorithm when N and K are increased (in experiments K is 40 per cent of N) and d is fixed. Table 4 shows that the total cost of ILS algorithm does not depend on d.

When ILS algorithm has finished, each processor has the solution of $\frac{N}{p}$ equations. If results are sent to processor P_0, the cost is approximately $(p-1)t_s +$

$t_w \sum_{i=1}^{N-\frac{N}{P}} (n_i + k_i - 1) \approx (p-1)t_s + t_w NK$ (due to $n_i + k_i - 1 = K$, since the equations are exactly identified).

4.3 Parallel Two-stage Least Squares (First Version)

Three different versions of the 2SLS algorithm are presented. The first is a basic algorithm which will be improved in the second and the third versions. In the first version, the structure of the parallel 2SLS algorithm is stated. In the others versions, the same structure is followed but matrix decompositions are used to obtain lower costs.

Since the solution of a system by 2SLS requires the use of OLS in each equation with *proxys*, predetermined and endogenous variables, it is necessary to send all the *proxys* and part of the system structure to each processor. Algorithm 7 shows the scheme of 2SLS when solving a complete system. Matrix X_e is formed by the endogenous variables (which now take the values of the *proxys*) of the equation, excluding the main endogenous and the predetermined variables. Thus, matrix X_e has size $d \times (n_i - 1 + k_i)$. In 2SLS, initially all the *proxys* are calculated. It could be that several of the N *proxys* will not be used, but since they can not be identified and the matrices are distributed at the beginning of the algorithm, it is better to calculate all of them. The cost is $T_{OLS_p}(N, d, K, \delta_{est} = 1)$.

When OLS ends, $\hat{Y}$ is distributed in all the processors in ScaLAPACK style and it is necessary to send the complete matrix to all the processors. The cost of sending $\hat{Y}$ is $T_{A2A}(dN, p)$.

Because each processor solves several equations, each processor needs to access $\hat{Y}$, X, B and Γ. If B and Γ are distributed at the beginning of the algorithm, all the processors have the complete structure of the system and it is not necessary to receive data from other processors. If B and Γ are in processor P_0, and given that each processor only needs a part of B and Γ ($\frac{N}{p}$ rows of each matrix), P_0 sends only this part, with cost $(p-1)\left(t_s + t_w \frac{N^2+NK}{p}\right)$.

Algorithm 7 Scheme of the parallel 2SLS algorithm

1: $\hat{Y} = OLS(Y, X, estimation =$**true**$)$
2: Distribute $\hat{Y}$ to all the processors
3: IN PARALLEL Each processor q DO
4: **for** j=1...$\frac{N}{p}$ **do**
5: $i = q + (j-1)p$
6: OLS(y_i,X_e,estimation=**false**)
7: **end for**
8: END PARALLEL

When the 2SLS algorithm has finished each processor has the solution of $\frac{N}{p}$ equations. If results are sent to processor P_0, the cost is approximately $(p-1)t_s + t_w \sum_{i=1}^{N-\frac{N}{p}} (n_i + k_i - 1)$.

Thus, the total cost without considering the cost of sending data to P_0, and considering the same approximation as in equation 12, is:

$$\begin{aligned} T_{2SLS_{p,1ver}}(N,d,K) &= T_{OLS_p}(N,d,K,\delta_{est}=1) + T_{A2A}(dN,p) + \\ + \max_{q=1...p} &\left\{ \sum_{j=1}^{\frac{N}{p}} \left(T_{OLS}(1,d,k_{q+(j-1)p} + n_{q+(j-1)p} - 1, \delta_{est}=0) \right) \right\} \approx \\ &\approx T_{OLS_p}(N,d,K,\delta_{est}=1) + T_{A2A}(dN,p) + \\ &\quad + \sum_{i=1}^{\frac{N}{p}} \left(T_{OLS}(1,d,k_i + n_i - 1, \delta_{est}=0) \right) \end{aligned} \tag{13}$$

As experimental results show (tables 5, 6 and 7), the total cost of the first version of 2SLS (equation 13) depends heavily on parameters N, K and d.

4.4 Second Version of 2SLS (Inverse Decomposition)

Since most of the equations in a system are over identified, it is necessary to improve the 2SLS algorithm. To obtain an optimized algorithm several objectives can be defined:

- ILS and 2SLS must share information.
- Each call to 2SLS must share more information to reduce the number of operations.

- Perform the maximum number of operations between all the processors at the beginning of the algorithm to be used for any processor in the other parts of the algorithm.

The first objective was satisfactorily achieved in the first version. But it is possible to obtain better results in the other objectives by using a special decomposition of the matrix in 2SLS. Thus, the following operations are made at the beginning of the algorithm and using all the processors:

- Π is calculated as $\Pi^t = (X^tX)^{-1}X^tY$ (using OLS(Y,X,estimation=**false**)), and additionally matrices X^tX and X^tY are saved for later use.
- Matrix $\hat{Y} = X\Pi^t$ (all the *proxys* variables) is calculated.
- Matrix $\hat{Y}^t\hat{Y}$ is calculated.

By using OLS(Y,X,estimation=**true**) and storing Π, the first two points are fulfilled.

Now, when ILS is used to solve an equation, the Π matrix is available, and when 2SLS is used, all the *proxys* had been calculated. Π has been used to calculate the *proxys*.

When 2SLS is used to solve an equation, it is not necessary to calculate the *proxys* but the algorithm calls OLS using the new variables and the predetermined variables of the equation. It has a high cost and does not use information calculated previously. But a decomposition of the matrix is possible according to the theorem bellow.

Theorem 1 *For symmetrical matrices A and D, if all necessary inverses exist, then*

$$\begin{pmatrix} A & B \\ B^t & D \end{pmatrix}^{-1} = \begin{pmatrix} A^{-1} & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} -E \\ Id \end{pmatrix} (D - B^tA^{-1}B)^{-1}(-E^t, Id) \qquad (14)$$

where $E = A^{-1}B$

Proof see Appendix A of [12]

Let us suppose that an equation is going to be solved using 2LSL. Since the *proxys* have already been calculated, the first step is to substitute the original endogenous variables in the equation by the *proxys*:

$$y_j = \alpha_0 + \gamma_1 x_{j_1} + \ldots + \gamma_k x_{j_k} + \alpha_1 \hat{y}_{j_1} + \ldots + \alpha_m \hat{y}_{j_m} + \epsilon \qquad (15)$$

The set of *proxys* will be denoted by $\hat{Y}_1$ and the set of predetermined variables by X_1. The matrix used in subsequent calls to OLS is $[X_1\hat{Y}_1]$ (with dimension $d\times(m+k)$). Then a call to OLS is made to calculate $([X_1\hat{Y}_1]^t[X_1\hat{Y}_1])^{-1}[X_1\hat{Y}_1]^t y_j$.

Using the previous theorem, the inverse of the matrix can be calculated as follows:

$$\left(\begin{bmatrix} X_1^t \\ \hat{Y}_1^t \end{bmatrix} \begin{bmatrix} X_1^t\hat{Y}_1^t \end{bmatrix}\right)^{-1} = \begin{pmatrix} X_1^tX_1 & X_1^t\hat{Y}_1 \\ \hat{Y}_1^tX_1 & \hat{Y}_1^t\hat{Y}_1 \end{pmatrix}^{-1} = \begin{pmatrix} (X_1^tX_1)^{-1} & 0 \\ 0 & 0 \end{pmatrix} +$$

$$\begin{pmatrix} -(X_1^tX_1)^{-1}X_1^t\hat{Y}_1 \\ Id \end{pmatrix} (\hat{Y}_1^t\hat{Y}_1 - Y_1^tX_1(X_1^tX_1)^{-1}X_1^t\hat{Y}_1)^{-1}(-Y_1^tX_1(X_1^tX_1)^{-1}, Id) \tag{16}$$

where a lot of data can be taken from the matrix calculated at the beginning of the algorithm. The operations necessary in equation 16 are shown bellow (some of then were obtained in previous computations and others are performed at this point):

- $X_1^tX_1$ is taken from X^tX, which was calculated at the same time as Π.
- $(X_1X_1)^{-1}$ is calculated (cost $\frac{2}{3}k^3$).
- $X_1^t\hat{Y}_1$ is taken from X^tY, which was calculated at the same time as Π. It is because $X^t\hat{Y} = X^tX\Pi = X^tX(X^tX)^{-1}X^tY = X^tY$.
- $(X_1^tX_1)^{-1}X_1^t\hat{Y}_1$ is calculated (cost $2k^2m$).
- $\hat{Y}_1^tX_1(X_1^tX_1)^{-1}X_1^t\hat{Y}_1$ is calculated (cost $2m^2k$).
- $\hat{Y}_1^t\hat{Y}_1$ is taken from $\hat{Y}^t\hat{Y}$, which was calculated earlier.
- $(\hat{Y}_1^t\hat{Y}_1 - Y_1^tX_1(X_1^tX_1)^{-1}X_1^t\hat{Y}_1)^{-1}$ is calculated (cost $\frac{2}{3}m^3$).
- Multiplication of the first and the second matrices in equation 16 has cost $2m^2k$, because the identity matrix is not computed.
- Multiplication of the previous matrix and the third matrix in equation 16 has cost $2m^2k$ because it also includes the identity matrix.

The cost of additions is not considered. The total cost is $\frac{2}{3}m^3 + \frac{2}{3}k^3 + 6m^2k + 2k^2m$.

Matrix $[X_1\hat{Y}_1]^t y_j$ must be also calculated. $X_1^t y_j$ can be taken from X^tY, which was calculated to obtain Π. $\hat{Y}_1^t y_j$ can be taken from $\hat{Y}^t\hat{Y}$ ($\hat{Y}^t\hat{Y} = \hat{Y}^tY$, see equation 22).

Finally, the multiplication of the two matrices is made and the total cost is $T_{OLS_{2ver}} = \frac{2}{3}(m^3+k^3) + 2(m+k)^2 + 6m^2k + 2k^2m$ when only one equation is solved.

It can be compared with the cost of the OLS which was used in the first version of 2SLS for an equation with m endogenous variables, k predetermined variables and sample size d, $T_{OLS_{1ver}} = T_{OLS}(1, d, m+k, \delta_{est} = 0) = \frac{2}{3}(m+k)^3 + 2(m+k)^2 d + 2(m+k)^2 + 2(m+k)d$.

Since $T_{OLS_{2ver}}$ does not depend on d, the cost of solving the equations (without considering the matrix operations in lines 1, 2 and 3 of algorithm 9) is independent of the sample size d.

On comparing the two expressions, it is possible to deduce that $T_{OLS_{1ver}} > T_{OLS_{2ver}}$. Because $\frac{2}{3}m^3 + \frac{2}{3}k^3 \leq \frac{2}{3}(m+k)^3 = \frac{2}{3}(m^3 + k^3 + 3m^2k + 3mk^2)$, it is enough to see that $6m^2k + 2k^2m < 2(m+k)^2 d + 2(m+k)d$. To calculate Π, $(X^tX)^{-1}$ must be calculated, and it is necessary that $K \leq d$ (X has $d \times K$ dimension), and $0 < m + k \leq K \leq d$ because the equation is identified.

Using the previous inequality, it is deduced that $(m+k)^3 < d(m+k)(m+k+1)$ and $3m^2k+k^2m \leq (m+k)^3$. Then $3m^2k+k^2m < (m+k)^3 < d(m+k)(m+k+1)$ and $T_{OLS_{2ver}} < T_{OLS_{1ver}}$.

Of course, there are several extra operations ($\hat{Y}\hat{Y}$ in line 3 of algorithm 9 must be computed) at the beginning of the algorithm in the second version. These extra operations have a cost of $2N^2d$. Experimental results show that the second version is faster than the first version.

When $k_i > 0$ and $n_i > 1$, the equation can be solved by OLS_{2ver}. When $k_i = 0$ or $n_i = 1$, which is very improbable, OLS can be used with a reduced cost because matrix $\hat{Y}^t\hat{Y}$ has been calculated before.

Algorithm 8 shows the scheme of OLS using the inverse decomposition equation 16.

Algorithm 8 Scheme of the OLS algorithm using inverse decomposition (OLS_{2ver})

1: Compute
$(X_1X_1)^{-1}$
$(X_1^tX_1)^{-1}X_1^t\hat{Y}_1$
$Y_1^tX_1(X_1^tX_1)^{-1}X_1^t\hat{Y}_1$
$(\hat{Y}_1^t\hat{Y}_1 - Y_1^tX_1(X_1^tX_1)^{-1}X_1^t\hat{Y}_1)^{-1}$

2: Compute $\begin{pmatrix} X_1^tX_1 & X_1^t\hat{Y}_1 \\ \hat{Y}_1^tX_1 & \hat{Y}_1^t\hat{Y}_1 \end{pmatrix}^{-1}$ {using equation 16}

3: Compute $([X_1\hat{Y}_1]^t[X_1\hat{Y}_1])^{-1}[X_1\hat{Y}_1]^t y_j$

The cost of OLS using the inverse decomposition is:

$$T_{OLS_{2ver}}(k_i, n_i) = \frac{2}{3}(k_i^3 + n_i^3) + 2(n_i + k_i)^2 + 6n_i^2 k_i + 2k_i^2 n_i \tag{17}$$

A scheme of the second version of 2SLS is shown in algorithm 9. A call to OLS_{2ver} is made sequentially in each equation and the scheme studied above is used.

Algorithm 9 Scheme of the parallel 2SLS algorithm 2nd version

1: $\hat{Y} = OLS_p(Y, X, estimation =$**true**) {saving Π, X^tX and X^tY}
2: $\hat{Y}^t\hat{Y}$ {parallel multiplications}
3: Distribute $\Pi, X^tX, X^tY, \hat{Y}, \hat{Y}\hat{Y}$ to all the processors
4: IN PARALLEL Each processor q DO
5: **for** j=1...$\frac{N}{p}$ **do**
6: $\quad i = q + (j-1)p$
7: $\quad OLS_{2ver}(y_i, \hat{Y}, X, X^tX, X^tY, \hat{Y}\hat{Y})$
8: **end for**
9: END PARALLEL

The total cost of the second version of 2SLS is:

$$\begin{gathered}
T_{2SLS_{p,2ver}}(N, d, K) = \\
T_{OLS_p}(N, d, K, \delta_{est} = 1) + \frac{2N^2 d}{p} + T_{A2A}(K^2 + N^2 + 2KN + dN, p) + \\
+ \max_{q=1...p} \left\{ \sum_{\substack{j=1 \\ t=q+(j-1)p}}^{\frac{N}{p}} \left(\frac{2}{3}(k_t^3 + n_t^3) + 2(n_t + k_t)^2 + 6n_t^2 k_t + 2k_t^2 n_t \right) \right\} \approx \\
\approx T_{OLS_p}(N, d, K, \delta_{est} = 1) + \frac{2N^2 d}{p} + T_{A2A}(K^2 + N^2 + 2KN + dN, p) + \\
\sum_{i=1}^{\frac{N}{p}} \left(\frac{2}{3}(k_i^3 + n_i^3) + 2(n_i + k_i)^2 + 6n_i^2 k_i + 2k_i^2 n_i \right)
\end{gathered} \tag{18}$$

Here, the same approximation as in 12 is considered. Equation 18 shows that the sum does not depend on parameter d.

4.5 Third Version of 2SLS (QR Decomposition)

A new algorithm using the Householder decomposition is studied. The exogenous matrix X is decomposed as QR and the new matrices are used. Similar

ideas have been used in [1], [2] and [14], where QR decomposition is used in different applications of Least Square problem. Now the same idea is adapted to Simultaneous Equations Models.

In [2] Givens Rotation is used instead of Householder decomposition when a column in X, and therefore in R, is lost. It is useful when looking for the best model of Least Squares.

In the solution of Simultaneous Equations Models, Householder decomposition is used at the beginning of the algorithm. In each equation, X_1 (the set of predetermined variables) has only a few columns of X and Givens Rotation loses its power.

Given the exogenous matrix X (with $d \times K$ dimension), there is an orthogonal matrix Q (with $d \times d$ dimension) and a triangular matrix R (with $d \times K$ dimension), with $X = QR$. R has the form $\begin{pmatrix} R_1 \\ 0 \end{pmatrix} \begin{matrix} K \times K \\ (d-K) \times K \end{matrix}$, where R_1 is an upper triangular matrix.

Partitioning Q in $(Q_1|Q_2)$, where Q_1 is $d \times K$, it is obtained:

$$X = QR = (Q_1|Q_2) \begin{pmatrix} R_1 \\ 0 \end{pmatrix} = Q_1 R_1 \tag{19}$$

Therefore:

$$\Pi = (X^t X)^{-1} X^t Y = (R_1^t Q_1^t Q_1 R_1)^{-1} R_1^t Q_1^t Y = (R_1^t R_1)^{-1} R_1^t Q_1^t Y_1 = R_1^{-1} Q_1^t Y \tag{20}$$

An expression for $\hat{Y}$ is:

$$\hat{Y} = X\Pi = QRR_1^{-1}Q_1^t Y = Q \begin{pmatrix} Id \\ 0 \end{pmatrix} Q_1^t Y = Q_1 Q_1^t Y \tag{21}$$

Using this expression it is easy to prove:

$$\hat{Y}^t \hat{Y} = Y^t Q_1 Q_1^t Q_1 Q_1^t Y = Y^t Q_1 Q_1^t Y = Y^t \hat{Y} \tag{22}$$

Because R_1 is an upper triangular matrix, it is less costly to calculate its inverse. LAPACK can be used here and also in the QR decomposition. Routines used in the experiments in section 5 are: *dtrtri* for the inverse; *dgeqrf* for QR

decomposition; and *dorgqr* to calculate Q. The cost of $dgeqrf$ is $\frac{2}{3}K^2(3d-K)$ and the cost of *dorgqr* is $\frac{2}{3}K^2(3d-K)$. The three versions of the 2SLS algorithm have been compared using only one processor (table 11).

A parallel version can be developed by substituting the LAPACK functions ($dgeqrf$, *dorgqr* and *dtrtri*) for their equivalent ones in ScaLAPACK ($pdgeqrf$, *pdorgqr* and *pdtrtri*).

The scheme of a new OLS using QR decomposition is shown in algorithm 10.

Algorithm 10 Scheme of OLS using QR decomposition (OLS_{3ver})

1: Obtain Q_1 and R_1 {cost $\rightarrow \frac{4}{3}K^2(3d-K)$}
2: Compute R_1^{-1} {cost $\rightarrow \frac{1}{3}K^3$}
3: $coef = R_1^{-1}Q_1^t y_i$ {cost $\rightarrow 2K(K+d)$}
4: **if** estimation=**true then**
5: $\quad est = Q_1 Q_1^t y_i$ {cost $\rightarrow 2dK$}
6: **end if**

Algorithm 10 is used by each processor in each equation solved by 2SLS. When the OLS is used in this case, the X matrix is formed by the *proxy* variables and the predetermined variables of the equation, and y_i is the main endogenous variable of the i-th equation. The cost of OLS using the QR decomposition is:

$$T_{OLS_{3ver}}(d, K, \delta_{est}) = 4K^2d - K^3 + 2K(K+d) + \delta_{est}2dK \tag{23}$$

On comparing expressions 23 and 17, it is possible to deduce that $T_{OLS_{3ver}} > T_{OLS_{2ver}}$. Because 2SLS is used in over identified equations, $K+1 > n_i + k_i > 0$, and then $3K^3 \geq 3(n_i+k_i)^3 > \frac{2}{3}(n_i^3+k_i^3) + 6n_i^3k_i + 2n_ik_i^3$. Furthermore $4K^2d - K^3 \geq 3K^3$ and $2K(K+d) \geq 4K^2 > 2(n_i+k_i)^2$, because $K \leq d$ (X has $d \times K$ dimension). Using the previous inequalities, it is deduced that $T_{OLS_{2ver}} < T_{OLS_{3ver}}$.

The scheme of 2SLS using the QR decomposition is shown in algorithm 11.

Algorithm 11 Scheme of parallel 2SLS algorithm using QR

1: Obtain Q_1 and R_1 {QR decomposition of X in parallel}
2: $\hat{Y} = Q_1 Q_1^t Y$ {parallel multiplications}
3: Distribute $\hat{Y}$ to all the processors
4: IN PARALLEL Each processor q DO
5: **for** j=1...$\frac{N}{p}$ **do**
6: $\quad i = q + (j-1)p$
7: $\quad OLS_{3ver}$(y_i,X_e,estimation=**false**)
8: **end for**
9: END PARALLEL

The problem here is that 2SLS and ILS algorithms do not share any information. In the general case both algorithms are used and Π has to be calculated to be used by ILS. Because 2SLS can be used in an exactly identified equation, one solution could be not to calculate Π and to solve these equations by 2SLS, but the cost would be higher than using ILS. Section 5 compares the cost of ILS and 2SLS (QR version) when all the equations in a system are exactly identified (table 12). A scheme of 2SLS using QR decomposition where Π matrix is calculated, is shown in algorithm 12.

Algorithm 12 Scheme of parallel 2SLS using QR decomposition in a system where ILS is used in exactly identified equations

1: Obtain Q_1 and R_1 {QR decomposition of X in parallel}
2: Compute R_1^{-1} {parallel inverse}
3: $\Pi^t = R_1^{-1}Q_1Y$ {parallel multiplications}
4: $\hat{Y} = Q_1Q_1^tY$ {parallel multiplications}
5: Distribute $\hat{Y}$ to all the processors
6: IN PARALLEL Each processor q DO
7: **for** j=1...$\frac{N}{p}$ **do**
8: $i = q + (j-1)p$
9: OLS_{3ver}(y_i,X_e,estimation=**false**)
10: **end for**
11: END PARALLEL

The cost of the inverse of R_1 is that of the sequential inverse ($\frac{1}{3}K^3$) divided by the number of processors, plus a communication cost of lower order, which is negligible and it is not considered. The cost of the sequential QR decomposition of X is considered in equation 23. Here the cost of the parallel QR decomposition (which is denoted $T_{QR_p}(d,K)$) is considered, but this cost is not that of the sequential version. A parallel algorithm can be seen in [13].

The total cost of the third version of 2SLS (using the same approximation as in equation 12) is:

$$\begin{gathered}
T_{2SLS_{p,3ver}}(N,d,K) = \frac{K^3}{3p} + T_{QR_p}(d,K) + T_{A2A}(dN,p) + \\
+ \max_{q=1...p}\left(\sum_{j=1}^{\frac{N}{p}}\left(T_{OLS_{3ver}}(k_{q+(j-1)p} + n_{q+(j-1)p} - 1, d, \delta_{est}=0)\right)\right) \approx \\
\approx \frac{K^3}{3p} + T_{QR_p}(d,K) + T_{A2A}(dN,p) + \sum_{i=1}^{\frac{N}{p}}\left(T_{OLS_{3ver}}(d, k_i + n_i - 1, \delta_{est}=0)\right)
\end{gathered} \tag{24}$$

Section 5, table 11, shows that the second version of the 2SLS algorithm

(inverse decomposition) is better than the QR version.

4.6 A general algorithm

A general algorithm to solve a system of Simultaneous Equations Models is shown. In a general system, there will be non identified, over identified and exactly identified equations. ILS and 2SLS are used. The second version of 2SLS is used in the general algorithm because it has the lowest cost.

Algorithm 13 Scheme of parallel general algorithm

1: $\Pi^t = OLS_p(Y, X$,estimation=**false**$)$
2: $\hat{Y} = X\Pi^t$ {parallel multiplications}
3: $\hat{Y}\hat{Y}$ {parallel multiplications}
4: Distribute $\Pi, X^tX, X^tY, \hat{Y}, \hat{Y}\hat{Y}$ to all the processors
5: IN PARALLEL Each processor q DO
6: **for** j=1...$\frac{N}{p}$ **do**
7: $i = q + (j-1)p$
8: **if** equation i is exactly identified **then**
9: Solve $-B_i\Pi = \Gamma_i$
10: **else if** equation i is over identified **then**
11: **if** $k_i > 0$ and $n_i > 1$ **then**
12: $OLS_{2ver}(y_i$,$\hat{Y}$,X,X^tX,X^tY,$\hat{Y}\hat{Y})$
13: **else**
14: $OLS(y_i,\ X_e$, estimation= **false**$)$
15: **end if**
16: **else**
17: nothing {equation i is not identified}
18: **end if**
19: **end for**
20: END PARALLEL

4.7 Summary of costs

Table 1 shows a summary of the theoretical costs of the algorithms for solving Simultaneous Equations Models. The techniques are ILS and 2SLS.

The table shows that the costs in communications and operations before the sum of the costs of solving the equations in the second version of 2SLS are lower than in the other two versions.

Because $d \geq K \geq k_i + n_i$, in equations where $n_i + k_i$ is small, the cost of the third version of 2SLS is greater than the cost of the first version.

Technique	version	cost
ILS		$T_{ILS_p}(N,d,K) = T_{OLS_p}(N,d,K,\delta_{est}=0) + T_{A2A}(KN,p) + \sum_{i=1}^{\frac{N}{p}} \left(\frac{2}{3}(K-k_i)^3 + 2(K-k_i)^2 + 2Nk_i\right) \approx \frac{N}{p}(K-k_i)^3$
2SLS	First	$T_{2SLS_{p,1ver}}(N,d,K) = T_{OLS_p}(N,d,K,\delta_{est}=1) + T_{A2A}(dN,p) + \sum_{i=1}^{\frac{N}{p}} (T_{OLS}(1,d,k_i+n_i-1,\delta_{est}=0)) \approx \frac{N}{p}(\frac{2}{3}(k_i+n_i-1)^3 + 2(k_i+n_i-1)^2 d)$
2SLS	Second (inverse decomp.)	$T_{2SLS_{p,2ver}}(N,d,K) = T_{OLS_p}(N,d,K,\delta_{est}=1) + \frac{2N^2 d}{p} + T_{A2A}(K^2+N^2+2KN+dN,p) + \sum_{i=1}^{\frac{N}{p}} \left(\frac{2}{3}(k_i^3+n_i^3) + 2(n_i+k_i)^2 + 6n_i^2 k_i + 2k_i^2 n_i\right) \approx \frac{N}{p}(\frac{2}{3}(k_i^3+n_i^3) + 6n_i^2 k_i + 2k_i^2 n_i)$
2SLS	Third (QR decomp.)	$T_{2SLS_{p,3ver}}(N,d,K) = \frac{K^3}{3p} + T_{QR_p}(d,K) + T_{A2A}(dN,p) + \sum_{i=1}^{\frac{N}{p}} (T_{OLS_{3ver}}(d,k_i+n_i-1,\delta_{est}=0)) \approx \frac{N}{p}(4(k_i+n_i-1)^2 d - (k_i+n_i-1)^3)$

Table 1
Summary of theoretical costs of algorithms ILS and different versions of 2SLS

It has been demonstrated that the cost of the sum of solving the equations in the second version is lower than in the others. Tables 7, 10 and 13 show that the cost of the operations before solving the equations in the three algorithms are negligible in comparison with the cost of solving the equations. Hence, the total cost of the second version is the lowest of the three.

In an exactly identified equation ($K-k_i = n_i-1$), the cost of the ILS algorithm is lower than the cost of the second version of 2SLS.

5 Experimental results

Experimental results have been obtained in:

- Kefren: A cluster of 20 biprocessors Pentium Xeon 2 Ghz interconnected by a SCI net with a Bull 2D topology in a mesh of 4×5. Each node has 1 Gigabyte RAM.
- Marenostrum: A supercomputer based on PowerPC processors, BladeCenter architecture, a Linux system and a Myrinet interconnection. The main characteristics are: 10240 IBM Power PC 970MP processors at 2.3 GHz

(2560 JS21 blades), 20 TB of main memory, 280 + 90 TB of disk storage and a peak Performance of 94,21 Teraflops. Marenostrum is the most powerful supercomputer in Europe and the fifth in the world, according to the last TOP500 list.

As has been shown, the theoretical cost depends on a large number of parameters. To simplify the experiments, these have been made with different numbers 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.

In all the experiments, the system to solve is in P_0, and the solutions are in P_0 when the algorithm finishes.

5.1 ILS

Algorithm 6 is experimentally analyzed. The system has all its equations exactly identified.

Tables 2 and 3 show the execution time and the speed-up obtained for the parallel ILS algorithm.

N:	500		1000		1500		2000		2500	
d:	500		500		1000		1000		1500	
proc.	time	Sp	time	Sp	time	Sp	time	Sp	time	Sp
1	18,17		230,78		1073,21		3118,85		7181,54	
2	9,28	1,96	137,53	1,68	530,95	2,02	1613,91	1,93	3663,72	1,96
4	5,04	3,61	58,59	3,94	267,45	4,01	809,66	3,85	1839,56	3,90
8	2,64	6,88	29,77	7,75	136,09	7,89	395,17	7,89	914,82	7,85
16	2,02	9,00	17,78	12,98	67,52	15,89	198,47	15,71	453,58	15,83

Table 2
Execution time (in seconds) and speed-up of ILS algorithm in Kefren, when varying the number of endogenous variables (N), the sample size (d) and the number of processors

Table 2 shows a good speed-up in all cases. Table 3 shows good speed-up when systems with large N and d are solved. This is because when d is small, the cost of the communications between the processors is not negligible compared with the cost of arithmetic operations. Superlinear speed-up is obtained in several experiments.

Table 4 shows the cost of the ILS algorithm when d varies and the rest of

N:	500		1000		1500		2000		2500	
d:	500		500		1000		1000		1500	
proc.	time	Sp	time	Sp	time	Sp	time	Sp	time	Sp
1	6,43		68,37		346,06		1263,33		2455,03	
4	1,73	3,72	18,52	3,69	87,18	3,97	310,81	4,06	550,55	4,46
8	1,01	6,37	9,56	7,15	44,52	7,77	156,62	8,07	287,36	8,54
16	0,63	10,22	5,25	13,02	24,53	14,11	74,63	16,93	147,30	16,67
32	0,45	14,44	4,78	14,29	12,62	27,42	38,99	32,40	77,82	31,55
64	0,33	19,27	1,90	35,96	7,70	44,92	21,40	59,03	40,39	60,79

Table 3
Execution time (in seconds) and speed-up of ILS algorithm in Marenostrum, when varying the number of endogenous variables (N), the sample size (d) and the number of processors

d:	500		1000		1500		2000	
	time	%	time	%	time	%	time	%
total time	229,63		230,44		228,54		231,26	
Π	0,96	0,42	1,42	0,62	1,95	0,85	2,48	1,07

Table 4
Execution time (in seconds) of ILS algorithm in Kefren, with N=1000, K=400, and varying the sample size (d), in one processor

the parameters are fixed. The total cost does not depend on d. Only the cost of calculating Π (line 1 of algorithm 6) increases with d, but this represents about 1 per cent of the total.

The most costly part in the algorithm is the solution of the equations (lines 4 to 9 of algorithm 6), and this must be parallelized.

5.2 2SLS

5.2.1 First Version

The first version of the 2SLS algorithm is experimentally analyzed (tables 5 and 6). The equations generated randomly are over identified.

It can be seen that the speed-up is not bad and it is better when the sample size (d) and the number of variables (N and K) are increased.

Table 5 shows a good speed-up when few processors are used or when the

system is large. When a lot of processors are used in a small system the speed-up is worse because of the communication cost.

N:	500		1000		1500		2000		2500	
d:	500		500		1000		1000		1500	
proc.	time	Sp	time	Sp	time	Sp	time	Sp	time	Sp
1	72,82		790,93		7031,96		19337,92		97874,21	
2	38,77	1,88	412,97	1,92	3538,48	1,99	9767,20	1,98	48124,48	2,03
4	19,83	3,67	206,92	3,82	1916,58	3,67	5154,38	3,75	22541,92	4,34
8	10,70	6,81	108,68	7,28	1082,13	6,50	2639,33	7,33	10686,11	9,15
16	7,45	9,77	65,34	12,10	561,43	12,53	1493,48	12,95	5531,40	17,68

Table 5
Execution time (in seconds) and speed-up of the first version of the 2SLS algorithm in Kefren, when varying the number of endogenous variables (N), the sample size (d) and the number of processors

N:	500		1000		1500		2000		2500	
d:	500		500		1000		1000		1500	
proc.	time	Sp	time	Sp	time	Sp	time	Sp	time	Sp
1	148,59		1604,48		13762,89		45046,84		248294,57	
4	39,92	3,72	409,74	3,92	3524,29	3,91	11813,29	3,81	63018,56	3,94
8	21,42	6,94	212,62	7,55	1891,09	7,28	6001,29	7,51	33575,22	7,40
16	15,86	9,37	138,49	11,59	926,25	14,86	3062,14	14,71	16466,12	15,08
32	8,16	18,22	72,18	22,23	493,78	27,87	1557,13	28,93	8476,88	29,29

Table 6
Execution time (in seconds) and speed-up of the first version of the 2SLS algorithm in Marenostrum, when varying the number of endogenous variables (N), the sample size (d) and the number of processors

It can be seen that the times in tables 5 and 6 are higher than those in tables 2 and 3.

d:	500		1000		1500		2000	
	time	%	time	%	time	%	time	%
total time	790,26		1754,95		4356,69		11601,32	
$\hat{Y}$	1,14	0,14	1,73	0,10	2,51	0,06	3,12	0,03

Table 7
Execution time (in seconds) of the first version of the 2SLS algorithm in Kefren, with N=1000, K=400 and varying the sample size (d), in one processor

Table 7 shows the cost of the first version of the 2SLS algorithm when d varies and the rest of the parameters are fixed. The total cost depends heavily on d. The cost of calculating $\hat{Y}$ (line 1 of algorithm 7) increases with d to a lower degree than the total cost, and it is about 0.1 per cent of the total.

Because the cost of calculating $\hat{Y}$ is negligible compared with the total cost, it is important to parallelize the solution of the equations (lines 4 to 7 of algorithm 7).

5.2.2 Second Version (Inverse Decomposition)

The second version of the 2SLS algorithm is experimentally analyzed (tables 8 and 9). It can be seen that the speed-up is good and it is better when N, K and d are increased.

N:	500		1000		1500		2000		2500	
d:	500		500		1000		1000		1500	
proc.	time	Sp	time	Sp	time	Sp	time	Sp	time	Sp
1	12,91		198,16		1225,33		4192,10		10217,02	
2	7,59	1,70	105,05	1,89	639,04	1,92	2154,62	1,95	5363,62	1,90
4	4,17	3,10	53,82	3,68	333,30	3,68	1086,43	3,86	2700,53	3,78
8	2,85	4,53	29,45	6,73	185,37	6,61	562,03	7,46	1388,88	7,36
16	1,68	7,68	16,72	11,85	99,30	12,34	299,77	13,98	721,13	14,17

Table 8
Execution time (in seconds) and speed-up of the second version of the 2SLS algorithm in Kefren, when varying the number of endogenous variables (N), the sample size (d) and the number of processors

Table 10 shows the cost of the second version of the 2SLS algorithm when d is varied and the other parameters are fixed. It can be seen that the total cost depends lightly on d. The cost of calculating $\hat{Y}$ and $\hat{Y}^t\hat{Y}$ (lines 1 and 2 of algorithm 9) increases with d to a high degree, and this cost and the communications influence the total cost by about 2 per cent. Furthermore, the cost of calculating $\hat{Y}$ and $\hat{Y}^t\hat{Y}$ is negligible compared with the total cost. Thus, it is important to parallelize the solution of the equations (lines 5 to 8 of algorithm 9). A big reduction of the execution time with respect to the first version can be observed (tables 7 and 10).

N:	500		1000		1500		2000		2500	
d:	500		500		1000		1000		1500	
proc.	time	Sp	time	Sp	time	Sp	time	Sp	time	Sp
1	21,20		352,22		2014,73		7005,57		18471,71	
4	6,26	3,39	91,55	3,85	528,08	3,82	1820,22	3,85	4930,59	3,75
8	3,49	6,07	48,23	7,30	274,49	7,34	944,70	7,42	2536,07	7,28
16	2,12	10,00	26,97	13,06	148,08	13,61	1425,33	4,92	1348,91	13,69
32	1,36	15,55	15,69	22,45	83,77	24,05	273,55	25,61	677,84	27,25
64	1,12	18,96	10,38	33,94	48,42	41,61	150,91	46,42	393,16	46,98

Table 9
Execution time (in seconds) and speed-up of the second version of the 2SLS algorithm in Marenostrum, when varying the number of endogenous variables (N), the sample size (d) and the number of processors

d:	500		1000		1500		2000	
	time	%	time	%	time	%	time	%
total time	197,43		202,78		203,35		228,08	
$\hat{Y}$, $\hat{Y}^t\hat{Y}$	2,05	1,04	3,40	1,68	4,99	2,45	6,29	2,76

Table 10
Execution time (in seconds) of the second version of the 2SLS algorithm in Kefren, with N=1000, K=400 and varying the sample size (d) in one processor

5.2.3 Third Version (QR Decomposition)

The last version of the 2SLS algorithm (QR decomposition) is experimentally analyzed in one processor. The results have been compared with those of the other versions. The third version is better than the first but worse than the second.

Table 11 shows a comparison between the first, second and third versions of the algorithm with one processor. It can be seen that the second algorithm is the best.

When an exactly identified equation is solved by 2SLS using the third version, the cost is higher than the cost when ILS is used. Table 12 shows a comparison between both algorithms. The cost of 2SLS is tested without calculating Π.

Table 13 shows the cost of the third version of the 2SLS algorithm when d is varied and other parameters are fixed. It can be seen that the total cost depends heavily on d. The cost of calculating $Q, R, R^{-1}, \Pi, \hat{Y}$ (lines 1 to 4 of algorithm 12) increases with d to the same degree as the total cost.

N:	500		1000		1500		2000		2500	
d:	500		500		1000		1000		1500	
proc.	time	Sp	time	Sp	time	Sp	time	Sp	time	Sp
1st ver	72,82		790,93		7031,96		19337,92		97874,21	
2nd ver	12,91	5,64	198,16	3,99	1225,33	5,74	4192,10	4,61	10217,02	9,58
3rd ver	74,32	0,98	549,07	1,44	4643,24	1,51	9676,49	2,00	29830,90	3,28

Table 11
Execution time (in seconds) and speed-up of the second and third versions of the 2SLS algorithm with respect to the first version, with one processor in Kefren, when varying the number of endogenous variables (N), the sample size (d), and the number of processors

N:	500		1000		1500		2000		2500	
d:	500		500		1000		1000		1500	
proc.	time	Sp	time	Sp	time	Sp	time	Sp	time	Sp
3rd ver	130,97		801,11		6332,98		14212,58		41815,91	
ILS	18,17	7,21	230,78	3,47	1073,21	5,90	3118,85	4,56	7181,54	5,82

Table 12
Execution time (in seconds) and speed-up of the ILS algorithm with respect to the third version of 2SLS (algorithm 11), with one processor in Kefren, when varying the number of endogenous variables (N), the sample size (d) and the number of processors

d:	500		1000		1500		2000	
	time	%	time	%	time	%	time	%
total time	554,64		1787,86		2244,22		2750,85	
Q,R,R^{-1},Π,$\hat{Y}$	13,38	0,02	21,39	0,01	43,59	0,02	75,60	0,03

Table 13
Execution time (in seconds) of the third version of the 2SLS algorithm in Kefren, with N=1000, K=400, and varying the sample size (d), in one processor

It is important to parallelize the solution of the equations (lines 7 to 10 of algorithm 12) because the cost of calculating $Q, R, R^{-1}, \Pi, \hat{Y}$ is negligible compared with the total cost (about 0.02 per cent).

6 Conclusions

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 and message-passing systems. Simultaneous Equations Models appear in different scientific fields, and because the size of the applications varies greatly, it is interesting to have tools for the solution of small and large systems. Routines in LAPACK and ScaLAPACK style have been developed for distributed memory environments. The parallel versions of the algorithms have proved to be scalable, and thus appropriate to afford solutions for huge systems, like those in the simulation of national or world economy.

Acknowledgment

This work has been funded in part by the Spanish MCYT and FEDER under Grant TIC2003-08238-C02-02 and Fundación Séneca 02973/PI/05.

The author gratefully acknowledges the computer resources, technical expertise and assistance provided by the Barcelona Supercomputing Center - Centro Nacional de Supercomputación (Marenostrum) and the Networking Group of the Polytechnic University of Valencia (Kefren).

References

[1] Foschi, P., Kontoghiorghes, E.J., Estimation of VAR Models: Computational Aspects, Computational Economics, Springer, 21(1), pages 3-22, 2003.

[2] Gatu, C., Kontoghiorghes, E.J., Parallel algorithms for computing all possible subset regression models using the QR decomposition, Parallel Computing, 29(4), pages 505-521, 2003.

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

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

[5] Gujarati, D.N., Basic Econometrics, McGraw Hill, 1995.

[6] Harchol-Balter, M., Black, P.E., Queueing Analysis of Oblivious Packet-Routing Networks, SODA: ACM-SIAM Symposium on Discrete Algorithms, pages 583-592, 1994.

[7] Henry, R., Lu, I., Beightol, L., Eckberg, D., Interactions between CO2 Chemoreflexes and Arterial Baroreflexes, Am. Journal of Physiology, 274 (Heart Circ. Physiol. 43), pages 2177-2187, 1998.

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

[9] Kumar, V., Grama, A., Gupta, A., Karypis, G., Introduction to Parallel Computing. Design and Analysis of Algorithms, The Benjamin Cumming Publishing Company, 1994.

[10] Lu, I., Jones, S.P., 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, 2004.

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

[12] Mardia, K.V., Kent J.T., Bibby J.M., Multivariate Analysis, Academic Press, 1979.

[13] Stewart, G.W., A Parallel Implementation of the QR Algorithm, Parallel Computing, 5(1-2), pages 187-196 1987.

[14] Yanev, P., Foschi, P., Kontoghiorghes, E.J., Algorithms for Computing the QR Decomposition of a Set of Matrices with Common Columns, Algorithmica, 39(1), pages 83-93, 2004.

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

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

[17] http://www.mpi-forum.org

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

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

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

[21] http://www.spss.com/

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