Solution of Simultaneous Equations Models by High Performance Methods

José Juan López Espín
Universidad Miguel Hernández (Elche, Spain)
Domingo Giménez Cánovas
Universidad de Murcia (Murcia, Spain)

Contents

Introduction

- The solution of a S.E.M. in high performance parallel systems is studied.
- The methods analyzed here are ILS and 2SLS.
- Parallel algorithms for distributed memory have been developed.
- The methods have been analyzed in different parallel systems.

Simultaneous Equations Models

The scheme of a system with M equations, M endogenous variables and k predetermined variables is (structural form)

$$Y_{1t} = \beta_{12}Y_{2t} + \beta_{13}Y_{3t} + \ldots + \beta_{1M}Y_{Mt} + \gamma_{11}X_{1t} + \ldots + \gamma_{1k}X_{kt} + u_{1t}$$
$$Y_{2t} = \beta_{21}Y_{1t} + \beta_{23}Y_{3t} + \ldots + \beta_{2M}Y_{Mt} + \gamma_{21}X_{1t} + \ldots + \gamma_{2k}X_{kt} + u_{2t}$$
$$\ldots$$
$$Y_{Mt} = \beta_{M1}Y_{1t} + \beta_{M2}Y_{2t} + \beta_{M3}Y_{3t} + \ldots + \beta_{MM-1}Y_{M-1t} + \gamma_{M1}X_{1t} + \ldots + \gamma_{Mk}X_{kt} + u_{Mt}$$

These equations can be represented in matrix form

$$BY_t + \Gamma X_t + u_t = 0$$

Simultaneous Equations Models

The structural form can be expressed in reduced form

$$Y_t = \Pi X_t + v_t$$

with $\Pi = -B^{-1}\Gamma$ and $v_t = -B^{-1}u_t$

$$Y_{1t} = \pi_{11}X_{1t} + \ldots + \pi_{1k}X_{kt} + v_{1t}$$

$$\ldots$$

$$Y_{Mt} = \pi_{M1}X_{1t} + \ldots + \pi_{Mk}X_{kt} + v_{Mt}$$

When ILS and 2SLS?

Three kind of equations:

- Underidentified >> not solve
- Overidentified >> 2SLS
- Just-identified >> ILS (also 2SLS)

OLS (Ordinary Least Squares)

OLS can be used to solve a regression model

$$Y_t = a_1 X_{1t} + \ldots + a_n X_{nt} + u_t$$

In matrix form

$$Y = bX + u$$

The expression of the estimator is

$$\hat{b} = (XX)^{-1} XY$$

ILS (Indirect Least Squares)

- The technique ILS needs the equation to be **exactly identified**
- Structural coefficients can be univocally obtained from the reduced form to solve an equation

$$-B_i\Pi = \Gamma_i$$

2SLS (Two Step Least Squares)

- OLS can not be used in structural form because random variable and endogenous variables are correlated
- Endogenous variables are replaced for approximations (proxys variables)
- The proxy of Y is calculated using OLS with Y and the exogenous in the system.
- When the endogenous have been replaced, OLS is used again in the equation

Parallel Algorithm for distributed memory

- Try to parallelize at the uppest level
- 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.
- ScaLAPACK and PBLAS libraries are used to make a portable program

OLS$_p$ (Parallel OLS)

```
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
```

In the experiments ***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.

ILS for a system (Parallel ILS)

- ILS in different equations can shared the Pi matrix
- Pi is calculated at the beginning of the algorithm and is used for all the processors
- Each processor needs to access Pi, and the system's structure, but it does not need the sample data.

```
1: $\Pi^t = OLS_p(Y, X, estimation = \text{false})$
2: Distribute $\Pi$ 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:     if equation $i$ is exactly identified then
7:         Solve $-B_i\Pi = \Gamma_i$
8:     end if
9: end for
10: END PARALLEL
```

2SLS for a system (Parallel 2SLS)

- 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.

The first version of 2SLS

- All the proxys are calculated at the beginning of the algorithm
- All the proxys are distributed in all the processors
- Each processor solves an equation using OLS sequentially

```
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
```

The 2nd v. of 2SLS (inverse decomposition)

Solve an equation where the proxy variables have been substituted before (they are calculated at the beginning)

$$y_j = a_0 + a_1 \hat{y}_{j_1} + \ldots + a_m \hat{y}_{j_m} + g_1 x_{j_1} + \ldots + g_k x_{j_k} + e$$

The set of endogenous variables of the equation is $\hat{Y}_1$ and X_1 is the set of predetermined, and then the variables of the equation are the matrix [$\hat{Y}_1$ X_1]

And ([$\hat{Y}_1$ X_1]^t [$\hat{Y}_1$ X_1])$^{-1}$[$\hat{Y}_1 X_1$]^t y_j must be solved

The 2nd v. of 2SLS (inverse decomposition)

The inverse:
$$\begin{pmatrix} X_1' \\ \hat{Y}_1' \end{pmatrix} \begin{pmatrix} X_1 \hat{Y}_1 \end{pmatrix} = \begin{pmatrix} X_1'X_1 & X_1'\hat{Y}_1 \\ \hat{Y}_1'X_1 & \hat{Y}_1'\hat{Y}_1 \end{pmatrix}^{-1} =$$

$$\begin{pmatrix} (X_1'X_1)^{-1} & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} -(X_1'X_1)^{-1}X_1'\hat{Y}_1 \\ Id \end{pmatrix} (\hat{Y}_1'\hat{Y}_1 - \hat{Y}_1'X_1(X_1'X_1)^{-1}X_1'\hat{Y}_1)^{-1}(-\hat{Y}_1'X_1(X_1'X_1)^{-1}, Id)$$

Using

$$\begin{pmatrix} A & B \\ B' & D \end{pmatrix}^{-1} = \begin{pmatrix} A^{-1} & 0 \\ 0 & 0 \end{pmatrix} + \begin{pmatrix} -A^{-1}B \\ Id \end{pmatrix} (D - B'A^{-1}B)^{-1}(-A^{-1}B, Id)$$

The 2nd v. of 2SLS (inverse decomposition)

$(X_1'X_1)$ is taken from X'X

$(X_1'X_1)^{-1}$ is calculated (cost $2/3k^3$)

$X_1'\hat{Y}_1$ is taken from X' Y

$(X_1'X_1)^{-1}X_1'\hat{Y}_1$ is calculated (cost $2k^2m$)

$\hat{Y}_1'X_1(X_1'X_1)^{-1}X_1'\hat{Y}_1$ is calculated (cost $2m^2k$)

$\hat{Y}_1'\hat{Y}_1$ is taken from $\hat{Y}'Y$

$(\hat{Y}_1'\hat{Y}_1 - \hat{Y}_1'X_1(X_1'X_1)^{-1}X_1'\hat{Y}_1)^{-1}$ is calculated (cost $2/3m^3$)

The 2nd v. of 2SLS (inverse decomposition)

To calculate $[X_1\ \hat{Y}_1]'y_j$

- $X'_1 y_j$ can be taken from $X_t Y$ which was calculated to obtain *Pi*
- $(\hat{Y}_1{}' y_j)$ can be taken from $\hat{Y}'Y$

The 2nd v. of 2SLS (inverse decomposition)

Finally, the algorithm is

```
1: $\hat{Y} = OLS_p(Y, X, estimation =$ true$)$ {saving $\Pi$, $X^tX$ 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:     $i = q + (j-1)p$
7:     $OLS_{2ver}(y_i, \hat{Y}, X, X^tX, X^tY, \hat{Y}\hat{Y})$
8: end for
9: END PARALLEL
```

The 3rd v. of 2SLS (QR decomposition)

X is decomposed as QR using Householder method, where Q is orthogonal and R upper triangular.

$$X = QR = (Q_1|Q_2)\begin{pmatrix} R_1 \\ 0 \end{pmatrix} = Q_1R_1$$

$$\Pi = (X^tX)^{-1}X^tY = (R_1^tQ_1^tQ_1R_1)^{-1}R_1^tQ_1^tY = (R_1^tR_1)^{-1}R_1^tQ_1^tY_1 = R_1^{-1}Q_1^tY$$

$$\hat{Y} = X\Pi = QRR_1^{-1}Q_1^tY = Q\begin{pmatrix} Id \\ 0 \end{pmatrix}Q_1^tY = Q_1Q_1^tY$$

The 3rd v. of 2SLS (QR decomposition)

```
$OLS_{3ver}(y_i, X_e, \text{estimation}=\textbf{false})$

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 $\rightarrow 2K(K+d)$}
```

The algorithm is

```
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:     $OLS_{3ver}(y_i, X_e, \text{estimation}=\textbf{false})$
10: end for
11: END PARALLEL
```

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

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

Computer System

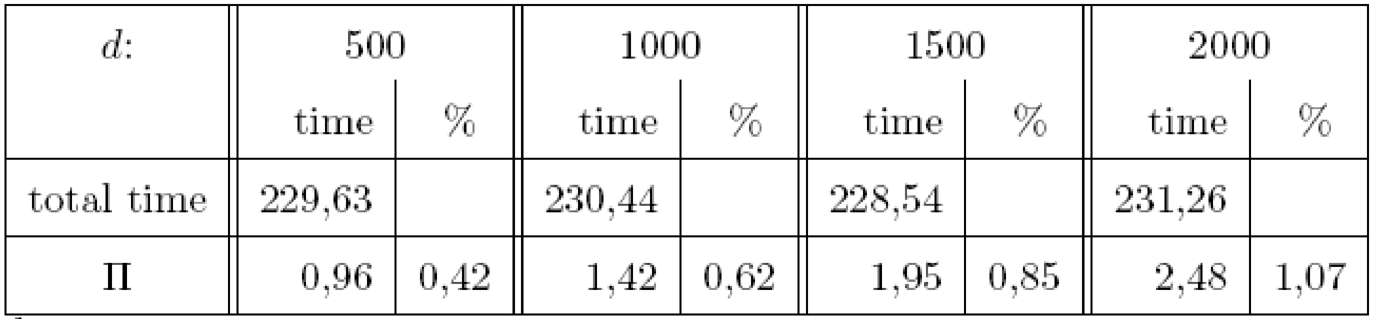
- Kefren: A cluster of 20 biprocessors Pentium Xeon 2 Ghz interconnected by a SCI net with a Bull 2D topology in a mesh of 4 £ 5. Each node has 1 Gigabyte RAM.
- Marenostrum: A supercomputer based on PowerPC processors, BladeCenter architecture, a Linux system and a Myrinet interconnection. The main 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.

ILS

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

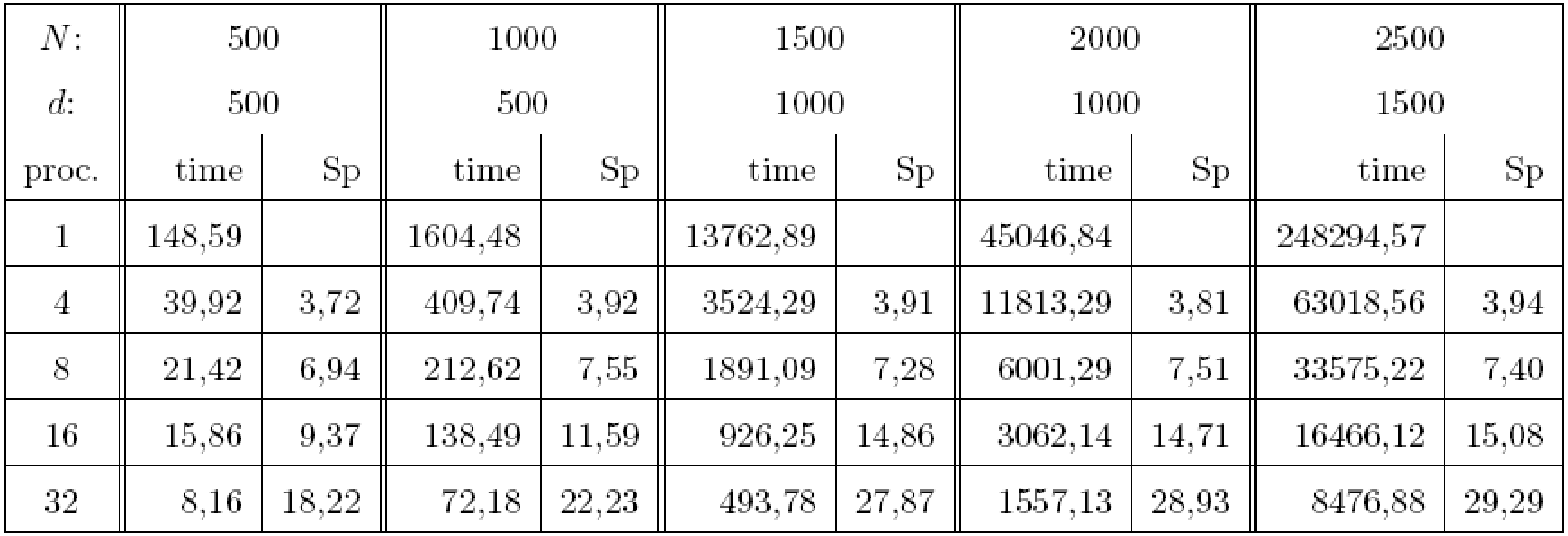
ILS



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

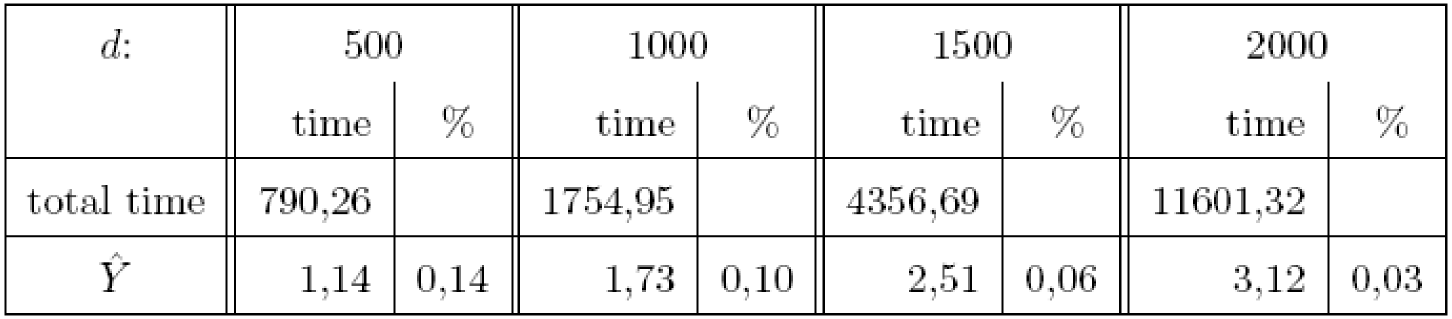
The first version of 2SLS



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

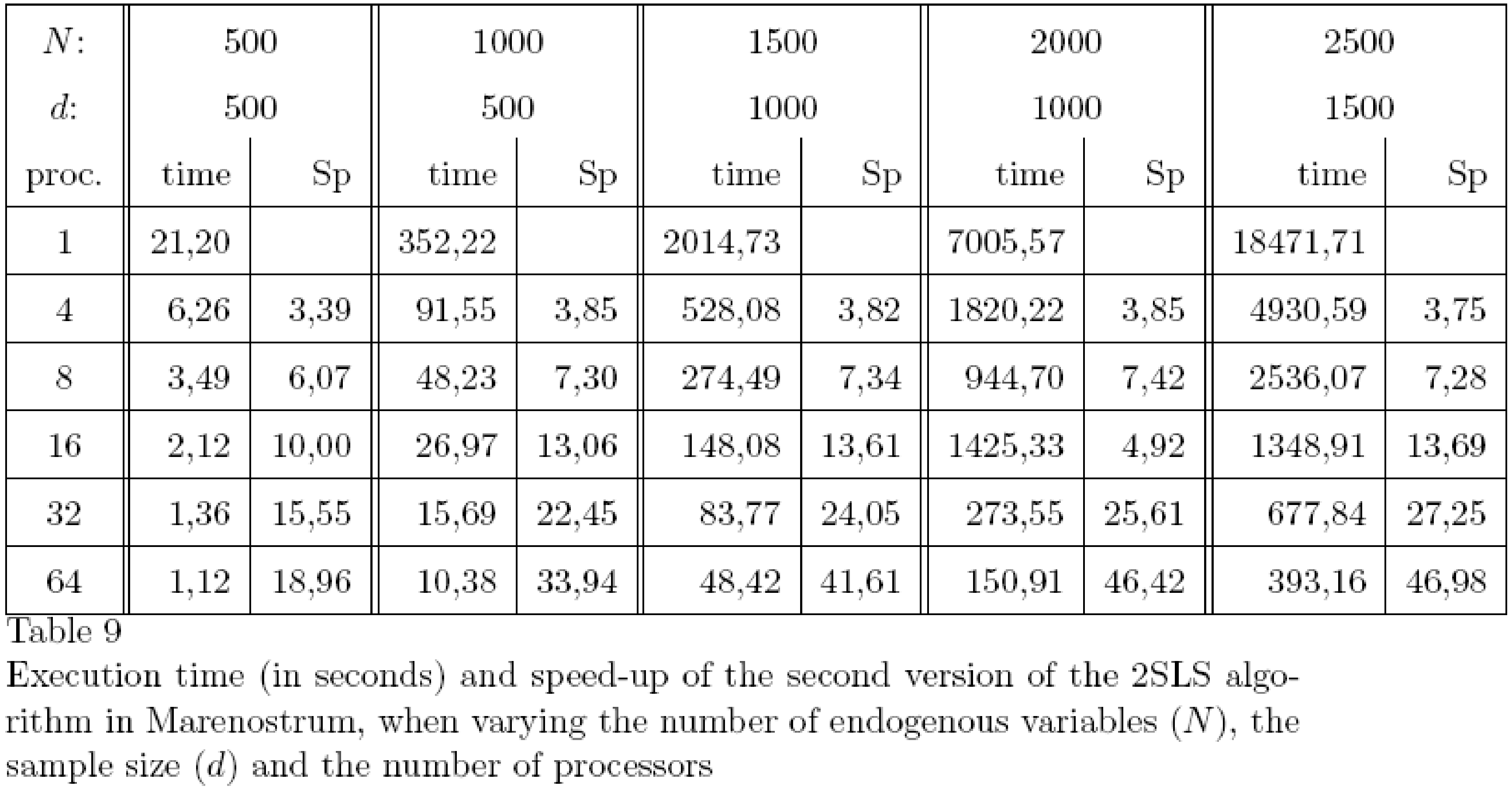
The first version of 2SLS



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

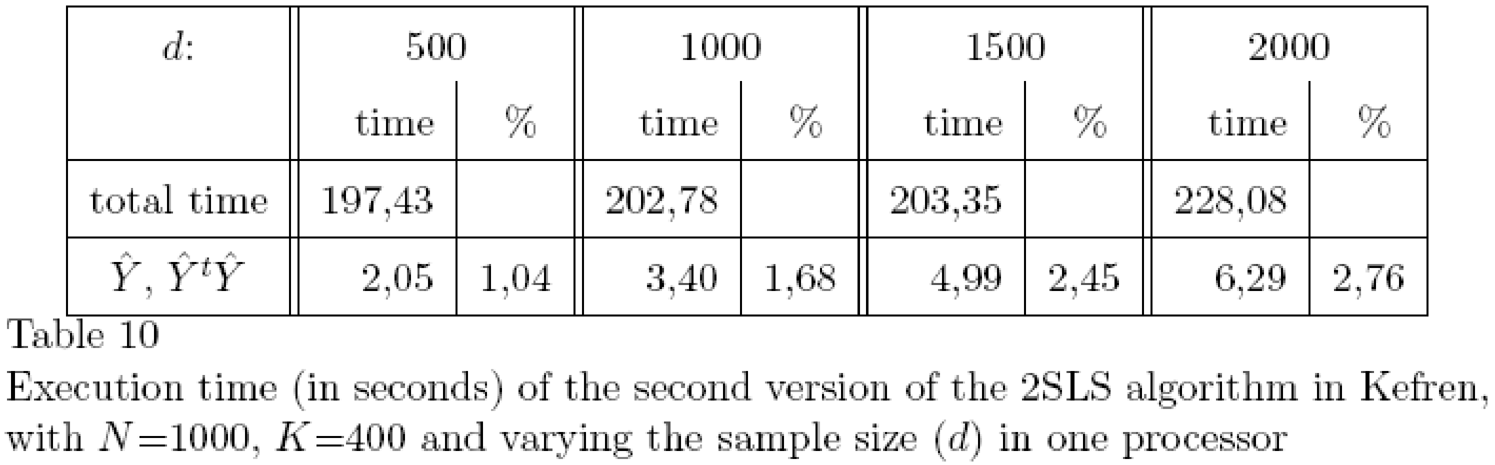
The 2nd v. of 2SLS (inverse decomposition)



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

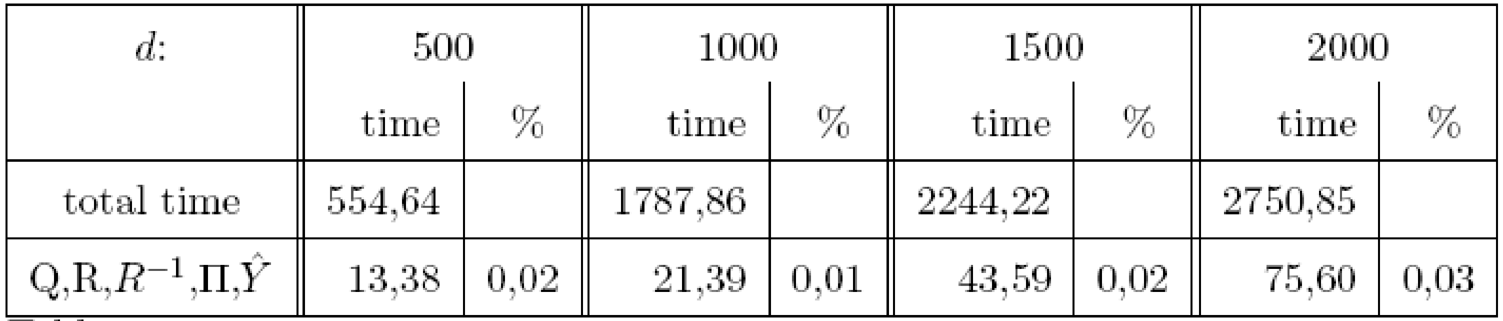
The 2nd v. of 2SLS (inverse decomposition)



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

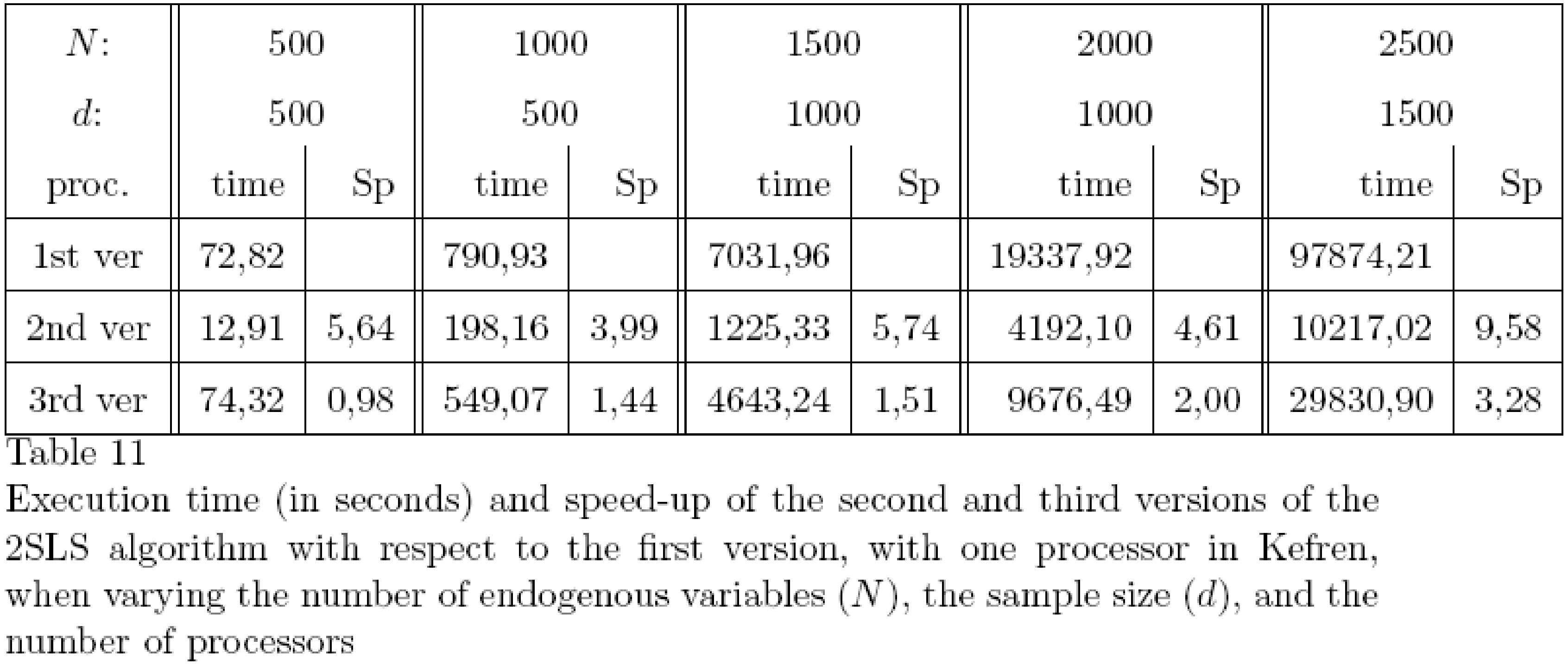
The 3rd v. of 2SLS (QR decomposition)



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

Comparison between the three techniques



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

Conclusions and Future works

- Sometimes a Simultaneous Equations Model needs special software and be solved in High Performance Systems
- Tools will be made freely available to the scientific community
- Application to real problems
- Develop an algorithm to find the best model