

Obtaining the coefficients of a Vector Autoregression Model through minimization of parameter criteria

Alfonso L. Castaño, Javier Cuenca, Domingo Giménez
Scientific Computing and Parallel Programming Group, University of Murcia



Jose J. López-Espín, Alberto Pérez-Bernabeu
Center of Operations Research, Miguel Hernández University



Outline

- 1 Context
- 2 Vector Autoregression Models
- 3 Computational aspects
- 4 Preliminary results
- 5 Conclusions and Future Work

Context

- A project to apply to medical data techniques traditionally used in econometrics:
Simultaneous Equation Models,
[Vector Autoregression Models](#) (VAR),
others...
- Big Data:
large amounts of data, difficult to model by experts,
design [software tools](#) to help experts.
- Work in [computational aspects](#) of VAR:
Matrix computation,
Metaheuristics,
Parallel computing...

General ideas

- The value of a variable at a time is linearly related not only to its own past but also to that of all other dependent variables in the model.
- It is also allowed for independent variables as explanatory variables. Therefore, not all variables follow this pattern, and there are both dependent and independent variables.
- Dependent variables are analyzed by the model.
- Independent variables are used to predict endogenous variables.

Model

$y^{(j)}$ and $z^{(j)}$ are vectors of dependent and independent variables at time j , $0 \leq j \leq t$.

$y^{(j)}$ depends on the value of y and z at i and k previous time points.

The linear dependence of $y^{(j)}$ with $y^{(j-r)}$ and $z^{(j-s)}$ is expressed by the matrices $A^{(r)}$ and $B^{(s)}$, $1 \leq r \leq i$, $1 \leq s \leq k$.

C denotes an independent vector.

$$y^{(j)} \approx y^{(j-1)}A^{(1)} + \dots + y^{(j-i)}A^{(i)} + z^{(j-1)}B^{(1)} + \dots + z^{(j-k)}B^{(k)} + C$$

Matrix representation

Reduced example:

- Only dependent variables.
- Dependency of two previous instants.
- Three data each time.

$$\begin{pmatrix} y_1^{(3)} & y_2^{(3)} & y_3^{(3)} \\ y_1^{(4)} & y_2^{(4)} & y_3^{(4)} \\ \dots & \dots & \dots \\ y_1^{(t)} & y_2^{(t)} & y_3^{(t)} \end{pmatrix} \approx \begin{pmatrix} y_1^{(2)} & y_2^{(2)} & y_3^{(2)} \\ y_1^{(3)} & y_2^{(3)} & y_3^{(3)} \\ \dots & \dots & \dots \\ y_1^{(t-1)} & y_2^{(t-1)} & y_3^{(t-1)} \end{pmatrix} * \begin{pmatrix} A_{11}^{(1)} & A_{12}^{(1)} & A_{13}^{(1)} \\ A_{21}^{(1)} & A_{22}^{(1)} & A_{23}^{(1)} \\ A_{31}^{(1)} & A_{32}^{(1)} & A_{33}^{(1)} \end{pmatrix} \\
 + \begin{pmatrix} y_1^{(1)} & y_2^{(1)} & y_3^{(1)} \\ y_1^{(2)} & y_2^{(2)} & y_3^{(2)} \\ \dots & \dots & \dots \\ y_1^{(t-2)} & y_2^{(t-2)} & y_3^{(t-2)} \end{pmatrix} * \begin{pmatrix} A_{11}^{(2)} & A_{12}^{(2)} & A_{13}^{(2)} \\ A_{21}^{(2)} & A_{22}^{(2)} & A_{23}^{(2)} \\ A_{31}^{(2)} & A_{32}^{(2)} & A_{33}^{(2)} \end{pmatrix}$$

Matrix representation

d data for dependent variables, i previous instants of dependent variables and k of external variables (we suppose $k \leq i$)

$$y^{(j)} \approx y^{(j-1)}A_1 + \dots + y^{(j-i)}A_i + z^{(j-1)}B_1 + \dots + z^{(j-k)}B_k + C$$

$$\begin{pmatrix} y^{(i)} \\ \vdots \\ y^{(t)} \end{pmatrix} \approx \begin{pmatrix} y^{(i-1)} & \dots & y^{(0)} & z^{(i-1)} & \dots & z^{(i-k)} & 1 \\ & & \vdots & & & & \vdots \\ y^{(t-1)} & \dots & y^{(t-i)} & z^{(t-1)} & \dots & z^{(t-k)} & 1 \end{pmatrix} \begin{pmatrix} A_1 \\ \vdots \\ A_i \\ B_1 \\ \vdots \\ B_k \\ C \end{pmatrix}$$

Optimization problems arising

- Solve $\min_D \|Y - XD\|_2$, with Y the matrix of dependent variables, X the matrix of dependent and independent variables at earlier times, and D the model to be obtained, formed by A , B and C .
- Other information criteria can be used, for example Akaike (AIC), Schwarz (BIC) or Hannan-Quinn (HQC) criterion.
- Determine which variables are dependent and which independent.
- Determine the number of previous instants of time which gives the best prediction (minimizing MSE).

Work lines

- Matrix computation:
libraries (LAPACK),
matrix decomposition (QR, LQ...),
exploitation of the structure of the matrices (Toeplitz-like).
- Optimization:
libraries (STATA),
metaheuristics.
- Parallelism:
in matrix computation,
at the metaheuristic level,
multilevel.

Matrix computation

- LAPACK has routines for Least Square:
using QR or LQ factorization (XGELS), using complete orthogonal factorization (XGELSY), using SVD (XGELSS), using divide-and-conquer SVD (XGELSD),
- MKL implementation exploits multithread parallelism.
- Extend the ideas of algorithms for Toeplitz matrices to these vector-Toeplitz-like matrices?
Linear Square library for structured matrices, UPV.

Optimization

- STATA has supported time series analysis.
- With matrix computation and parallelism, reduce the execution time.
- With metaheuristics, reduce the execution time at the expense of suboptimal solutions.
- Analyze combination of matrix computation and metaheuristics.

One possibility, generate initial elements with matrix methods, and iterate over the reference set with metaheuristics.

The function to optimize can vary. Initially Least Square, and Akaike for the iterations.

Parallelism

- Some libraries (MKL) can be used with implicit parallelism, this can be exploited only for large amounts of data.
- Metaheuristics can be parallelized in many ways:
 - in hybrid methods with local search from the solution provided by Least Square, the amount of computation may not be enough for exploitation of parallelism;
 - in population based methods the computation of the fitness can be costly, but the exploitation of the matrices' structure reduces the cost
- Multilevel parallelism is a better approach, with one or two levels of explicit parallelism and implicit parallelism only for large problems.

Simulator

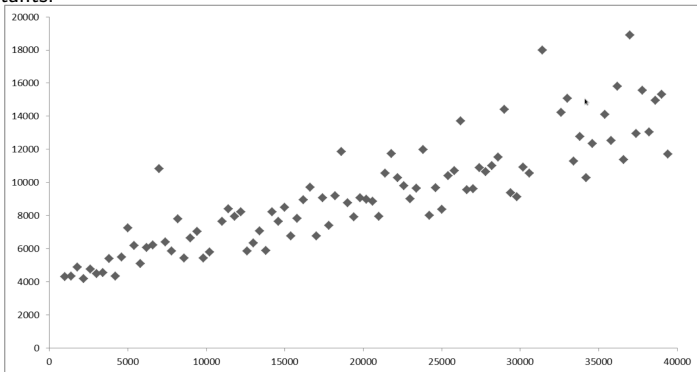
A simulator is being developed.

- At present in a preliminary version, available at https://github.com/fylux/tsf_var.
- Works with synthetically generated series.
- Uses the Least Square routines in LAPACK.
- Satisfactory results: approximation to the model used to generate the synthetic series.

Parallelism

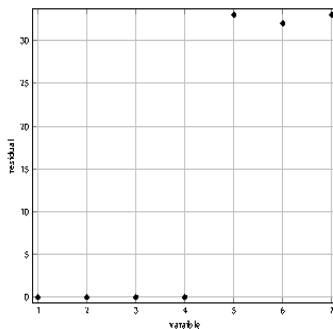
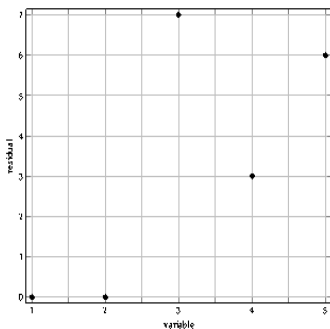
Few opportunities for parallelism, with 10 dependent variables and 10 independent variables, dependences on 20 previous instants for dependent and independent variables and 300000 times, speed-up of $1.305\times$ with 4 threads.

Low execution time: evolution of the execution time (msec) with the size of the series, for 3 dependent and 2 independent variables, with dependence on 4 and 2 previous instants.



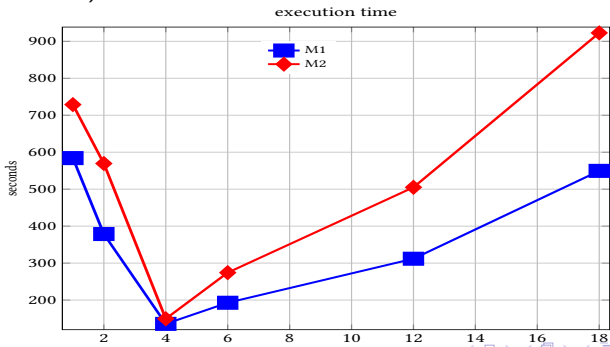
Discrimination of exogenous variables

Exogenous variables can be determined by analyzing the residual when the model is solved taking them as internal.



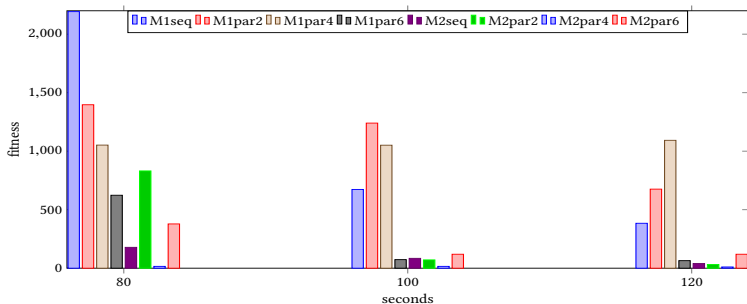
Metaheuristics

- Experiments with Genetic Algorithm, Scatter Search, GRASP, Taboo Search and hybridations.
- Sequential and shared-memory versions.
- With synthetic data and minimizing mean square distance (modify to optimize Akaike and to generate initial set with LAPACK).



Metaheuristics - Experiments

- Distributed metaheuristics:
 - M1: reference set of 100 elements, 150 combinations, 20% mutation
 - M2: reference set of 20 elements, 290 combinations, no mutations
- Multicore with 12 cores + hyperthreading



More Metaheuristics

- A simplified problem proposed as a project in a course on Methodology of Parallel Programming (<http://dis.um.es/~domingo/apuntes/AlgProPar/1718/practicafinal.pdf>): three internal variables and two time dependencies.
- Restrictions in the model: interval of integers, interval of floats, set of floats.
- Direct application of metaheuristics: hill climbing, simulated annealing, GRASP, iterative local search, genetic algorithm, ant colony, artificial bee colony.
- Sequential, shared-memory and message-passing implementations.

Metaheuristics in the simulator

- Not included yet.
- At present:
 - Experimenting with generation of initial elements with LAPACK, followed by local search to optimize the Akaike criterion.
 - Comparison of the solutions with those with the metaheuristics applied directly.
 - Application of parallelism to reduce the execution time of the metaheuristics, with combination with parallelism of matrix operations in the computation of fitness.
 - Simultaneous computation of fitness in population metaheuristics, with reuse of information and batched operations for reduction of the execution time.

Conclusions

- Matrix representation of VAR.
- Several computational problems arise:
matrix computation aspects,
metaheuristics,
exploitation of parallelism.
- Working on a simulator for solution of VARs:
now working with synthetic data,
shows capacity to solve synthetic models and to be used for
data analysis.

Future Work

- The simulator will be enriched with the results from the analysis of the different computational problems,
- and applied to real problems in different fields (medical and economic data).
- The work is part of a project applying econometric techniques to model medical data, so the models obtained with VARs will be compared with those with other techniques.