

Multivariate time series estimation via projections and matrix equations

Federico Poloni¹
Giacomo Sbrana²

¹U Pisa, Italy, Dept of Computer Science
²NEOMA Business School, Rouen, France

Structured Numerical Linear and Multilinear Algebra
Analysis, Algorithms and Applications — September 2014

Moving average models

Moving average models [Lütkepohl '06 book, Tsay '14 book]

$\{u_t \in \mathbb{R}^n\}$ Gaussian independent $N(0, I)$

$$y_t = A_0 u_t + A_1 u_{t-1} \quad MA(1)$$

$$y_t = A_0 u_t + A_1 u_{t-1} + A_2 u_{t-2} + \cdots + A_q u_{t-q} \quad MA(q)$$

(A_i real square matrices, y_t, u_t vectors)

Example: Yearly inflation data looks a lot like a y_t

Our problem: **estimation** Given output y_t for $t = 1, 2, \dots, T$, find the A_i 's

Scalar models popular for macroeconomics; multivariate ones less used because of estimation difficulties

Transfer function formalism

To $A_0 u_t + A_1 u_{t-1} + A_2 u_{t-2} + \dots + A_q u_{t-q}$ we associate

$$G(\lambda) = A_0 + A_1 \lambda + A_2 \lambda^2 + \dots + A_q \lambda^q$$

$G(\lambda)$ assumed nonsingular ($\det(A_0) \neq 0$), and all eigvals outside unit circle

Definition

$$G^*(\lambda) := A_0^\top + A_1^\top \lambda^{-1} + A_2^\top \lambda^{-2} + \dots + A_q^\top \lambda^{-q}$$

λ “behaves like” a complex number on the unit circle

$$G(\lambda)G^*(\lambda) = M_{-q} \lambda^{-q} + \dots + M_{-1} \lambda^{-1} + M_0 + M_1 \lambda + \dots + M_q \lambda^q$$

is **palindromic** ($M_0 = M_0^\top$ and $M_{-i} = M_i^\top$ for each i)

Autocovariance generating function

$$G(\lambda)G^*(\lambda) = M_{-q}\lambda^{-q} + \dots + M_{-1}\lambda^{-1} + M_0 + M_1\lambda + \dots + M_q\lambda^q$$

Theorem [Box, Jenkins, '76 book]

$$M_i = \mathbb{E} \left[y_t y_{t-i}^\top \right] \text{ for each } i = -q, \dots, q$$

- 1 Estimate $\tilde{M}_i := \frac{1}{T-1} \sum_{i=1}^T y_t y_{t-i}^\top$
- 2 Factorize $\tilde{M}(\lambda) = \tilde{G}(\lambda)\tilde{G}^*(\lambda)$

Point 2. is **spectral factorization** [Wiener, Kailath, Kucera, Ran et al, Ephremidze et al; many more for the scalar case]

Spectral factorization and matrix equations

$$M_{-1}\lambda^{-1} + M_0 + M_1\lambda = (A_1\lambda + A_0)(A_1\lambda + A_0)^*$$

Several approaches to solve it:

- $Y = A_0 A_0^\top$ solves $M_1 Y^{-1} M_1^\top + Y = M_0$ [Ran et al., Meini, CH Guo...]
- $Z = -A_0^{-\top} A_1^\top$ solves $M_{-1} + M_0 Z + M_1 Z^2$ [Bini et al, Higham and Kim, Meerbergen and Tisseur...]
- $G(\lambda)$ left spectral divisor of $M(\lambda)$ [Gohberg–Lancaster–Rodman]:
linearization (**palindromic?**) + Schur form

Extreme accuracy not necessary here — larger error from approximations \tilde{M}_i already

What can go wrong

- Factorize $\tilde{M}(\lambda) = \tilde{G}(\lambda)\tilde{G}^*(\lambda)$

Factorization $\tilde{M}(\lambda) = \tilde{G}(\lambda)\tilde{G}^*(\lambda)$ exists iff $\tilde{M}(z) \geq 0$ for each $z \in \mathbb{C}, |z| = 1$

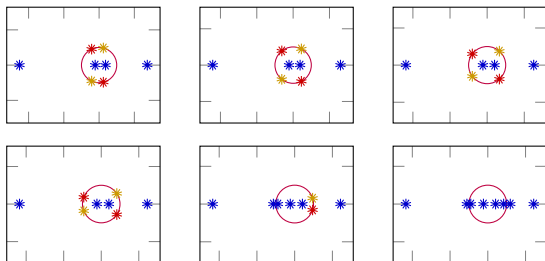
\tilde{M}_i are only approximate, so this property may fail

Possible solutions . . .

- Apply a correction to $\tilde{M}(z)$
- Robust factorization algorithm \rightarrow do something in case of error (something like a “robust chol”)
- Get better \tilde{M}_i 's

Possible solutions

Apply a correction to $\tilde{M}(z)$ [Brüll, P, Sbrana, Schröder, preprint] regularization approach: perturb polynomial to move eigenvalues



Other approaches for similar structures around

[Alam et al, Benner and Voigt, Grivet-Talocia, Guglielmi-Overton. . .]

Robust factorization algorithm Not much usable around! [Kucera, 91]

Trying to modify some (Riccati-type eigensolvers, symbolic Cholesky factorization [Janashia, Lagvilava, Ephremidze '11])

Get better \tilde{M}_i 's

Other (slower) estimation methods typically get better \tilde{M}_i 's, such as **Maximum Likelihood** (ML)

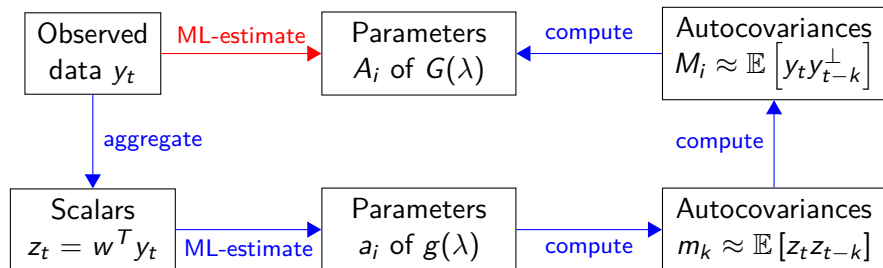
Our idea: get the \tilde{M}_i 's from **projection + scalar ML**

Algorithm

- 1 Choose weight row vector w , build scalar process $z_t = \{wy_t\}_{t=1,\dots,T}$
- 2 Use scalar ML to get $g(\lambda)$
- 3 $g(\lambda)g^*(\lambda) = m(\lambda) = wM(\lambda)w^T$
- 4 Repeat with many different w 's to reconstruct $M(\lambda)$

Convergence properties to correct solution when $T \rightarrow \infty$ (technical) [P, Sbrana, submitted]

The algorithm in a picture



We take the **blue** road rather than the **red**

Many (**easier**) scalar problems rather than a (**difficult**) multivariate one

A 2×2 example

- $w = \begin{bmatrix} 1 & 0 \end{bmatrix}$:

$$\begin{aligned} wM(\lambda)w^T &= \begin{bmatrix} 1 & 0 \end{bmatrix} \left(\begin{bmatrix} a & b \\ c & d \end{bmatrix} \lambda^{-1} + \begin{bmatrix} e & f \\ f & g \end{bmatrix} + \begin{bmatrix} a & c \\ b & d \end{bmatrix} \lambda \right) \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ &= a\lambda^{-1} + e + a\lambda \end{aligned}$$

- $w = \begin{bmatrix} 0 & 1 \end{bmatrix}$:

$$\begin{aligned} wM(\lambda)w^T &= \begin{bmatrix} 0 & 1 \end{bmatrix} \left(\begin{bmatrix} a & b \\ c & d \end{bmatrix} \lambda^{-1} + \begin{bmatrix} e & f \\ f & g \end{bmatrix} + \begin{bmatrix} a & c \\ b & d \end{bmatrix} \lambda \right) \begin{bmatrix} 0 \\ 1 \end{bmatrix} \\ &= d\lambda^{-1} + g + d\lambda \end{aligned}$$

- $w = \begin{bmatrix} 1 & 1 \end{bmatrix}$:

$$m(\lambda) = (a + b + c + d)\lambda^{-1} + (e + 2f + g) + (a + b + c + d)\lambda$$

If $b = c$, enough to get the remaining entries

Symmetric case

For a **symmetric** MA(1),

- $w = e_i$ for all i and $w = e_j + e_k$ for all $j \neq k$ enough to get $M(\lambda)$
- $A\lambda^{-1} + B + A\lambda$, with $A = A^\top, B = B^\top$ can be **simultaneously diagonalized** (congruence) \rightarrow easier to find spectral factorization

This is enough to solve some easier models:

- Exponentially weighted moving average (random-walk-plus-noise) – MA(1) with symmetric matrices
- “Multiple trends” — MA(2) with symmetric matrices and $M_1 = -4M_2$

Nonsymmetric case

Idea Use $w(\lambda)$, not constant w !

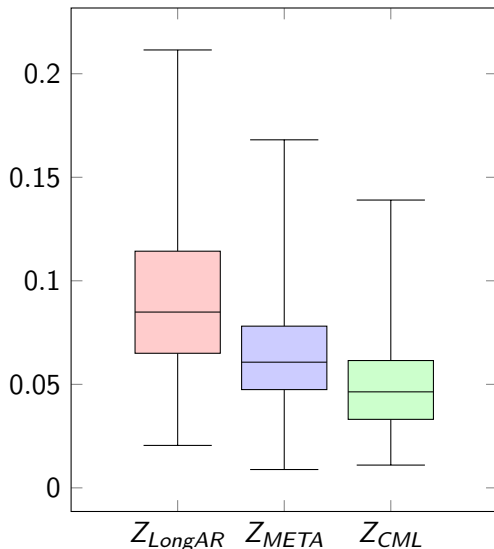
$$\begin{bmatrix} 1 & \lambda \end{bmatrix} \begin{bmatrix} y_{t,1} \\ y_{t,2} \end{bmatrix} = y_{t,1} + \lambda y_{t,2}$$

- $w = \begin{bmatrix} 1 & \lambda \end{bmatrix}$

$$\begin{aligned} w(\lambda)M(\lambda)w^* &= \begin{bmatrix} 1 & \lambda \end{bmatrix} \left(\begin{bmatrix} a & b \\ c & d \end{bmatrix} \lambda^{-1} + \begin{bmatrix} e & f \\ f & g \end{bmatrix} + \begin{bmatrix} a & c \\ b & d \end{bmatrix} \lambda \right) \begin{bmatrix} 1 \\ \lambda^{-1} \end{bmatrix} \\ &= c\lambda^{-2} + (a + d + f)\lambda^{-1} + e + 2b + g \\ &\quad + (a + d + f)\lambda + c\lambda^2 \end{aligned}$$

$MA(q + 1)$: more difficult to estimate, but still scalar

Some results



- LongAR** Quicker but less accurate method
- META** Our method (Moment Estimation Through Aggregation)
- CML** Conditional Maximum Likelihood. Scales badly, basically impossible to beat

Frobenius norm relative errors, 300 trials, 2×2 problem, excluding 48 non-passive samples.

`eigvals(theta)=[-0.927492, -0.172508]`

Some more results

Symmetric model (EWMA), against out-of-the-box ML on the MA(1) representation (**larger** parameter space, difficult to enforce symmetry)

	T	Θ META	Θ ML	Σ_u META	Σ_u ML	Time META	Time ML
$d = 2$, easy	200	202.52	236.77	108.28	109.11	1.55	59.99
	400	121.41	138.13	83.31	82.93	3.13	107.28
	1000	80.83	101.53	48.65	48.96	8.34	226.64
$d = 2$, harder	200	69.51	78.26	97.50	98.31	1.36	46.53
	400	48.26	56.48	80.91	81.52	3.74	113.36
	1000	28.01	34.07	47.60	48.02	7.63	210.06
$d = 3$, easy	200	205.07	254.49	135.26	136.41	2.65	201.41
	400	162.95	187.40	93.48	93.21	5.77	316.74
	1000	93.85	108.92	60.08	61.13	12.62	523.67
$d = 3$, harder	200	86.66	107.22	123.86	124.19	2.81	202.95
	400	57.03	67.25	95.13	96.75	5.66	295.93
	1000	29.91	37.04	61.78	62.13	13.12	541.36

Frobenius norm parameter errors $\times 10^3$, average on 500 trials

+ real-life inflation test by Bankitalia

Parallelizability

```
@parallel for i = 1:nWeights
  # Estimate  $g_i(\lambda)$  with weight  $w_i$ 
end
# Put together estimates to form  $M(\lambda)$  (fast)
# Spectral factorization (fast)
```

vs

```
while (convergence)
  #compute likelihood( $G(\lambda)$ )
  #compute  $G_{i+1}(\lambda)$  from  $G_i(\lambda)$  using BFGS-like search
end
function likelihood(G)
  for t = 1:T
    # update state[t] from state[t-1], compute likelihood
  end
end
```

Why isn't this...

Why isn't this model reduction/system identification? Similar to tangential interpolation [Antoulas, Beattie, Gugercin '10 inbook] but...

- No access to input u_t
- No possibility to get $G(\lambda)$ exactly: $G(\lambda)Q$ equally good
- We work on a “squared” version: not clear how to relate $g(\lambda)$ and $G(\lambda)$ in our setting

Why isn't this compressed sensing? Similar to phase retrieval [Candès, Strohmer, Voroninski '11], but

- it is a “matrix version”, uncertainty by an orthogonal matrix
- no sparsity/rank constraint, we want full reconstruction with a full set of measurements

Why isn't this Kalman filtering?

- KF assumes model known, our goal is estimating it here

Conclusions and issues

- Interesting idea to explore, new application area
- Works well in some cases (including symmetric problems)
- Still missing ideas: fit $G(\lambda)$ directly **without squaring**, find optimal weights and equations in LS
- Tricky to develop with the current software ecosystem
- **Open problem:** better “regularizing” spectral factorization algorithm:
We know well how to solve Riccati-like equations, now we want to **solve the ones that have no solution!**

Conclusions and issues

- Interesting idea to explore, new application area
- Works well in some cases (including symmetric problems)
- Still missing ideas: fit $G(\lambda)$ directly **without squaring**, find optimal weights and equations in LS
- Tricky to develop with the current software ecosystem
- **Open problem:** better “regularizing” spectral factorization algorithm:
We know well how to solve Riccati-like equations, now we want to **solve the ones that have no solution!**

Thanks for your attention