ML Estimation of State Space Models

May 25, 2012

Maximizing the likelihood

How can we estimate parameters when we cannot maximize likelihood analytically?

We need to

- Be be able to evaluate the likelihood function for a given set of parameters
- Find a way to evaluate a sequence of likelihoods conditional on difference parameter vectors so that we can feel confident that we have found the parameter vector that maximizes the likelihood

Numerical maximization of likelihood functions

Estimating richer state space models

- Likelihood surface may not be well behaved

We will need more sophisticated maximization routines

Numerical maximization of likelihood functions

Today:

- (Very brief) review of grid search, steepest ascent and Newton-Raphson algorithms
- Simulated annealing

Based on selected parts of Ch 5 of Hamilton and articles by Goffe, Ferrier and Rogers (1994) and Duffee (2011).

Grid search

Divide range of parameters into grid and evaluate all possible combinations

Pros:

- With a fine enough grid, grid search always finds the global maximum (if parameter space is bounded)

Cons:

- Computationally infeasible for models with large number of parameters

Steepest Ascent method

A blind man climbing a mountain.
Pros:

- Feasible for models with a large number of parameters

Cons:

- Can be hard to calibrate even for simple models to achieve the right rate of convergence
  - Too small steps and "convergence" is achieved to soon
  - Too large step and parameters may be sent off into orbit.
- Can converge on local maximum. (How could a blind man on $K2$ find his way to Mt Everest?)

Newton-Raphson

Newton-Raphson is similar to steepest ascent, but also computes the step size

- Step size depends on second derivative
- May converge faster than steepest ascent
- Requires concavity, so is less robust when shape of likelihood function is unknown

Simulated Annealing Goffe et al (1994)

- Language is from thermodynamics
- Combines elements of grid search with (strategically chosen) random movements in the parameter space
- Has a good record in practice, but cannot be proven to reach global max quicker than grid search.

Simulated Annealing: The Algorithm

Main inputs: $\Theta^{(0)}$, temperature T, boundaries of Θ, temperature reduction parameter r_T (and the function to be max/minimized $f(\Theta)$).

1. $\theta_j' = \theta_j^{(0)} + r \cdot v_j$ where $r \sim U[-1, 1]$ and v_i is an element of the step size vector V.
2. Evaluate $f(\Theta')$ and compare with $f(\Theta^{(0)})$. If $f(\Theta') > f(\Theta^{(0)})$ set $\Theta^{(1)} = \Theta'$. If $f(\Theta') < f(\Theta^{(0)})$ set $\Theta^{(1)} = \Theta'$ with probability $e^{(f(\Theta') - f(\Theta^{(0)})/T}$ and $\Theta^{(1)} = \Theta^{(0)}$ with probability $1 - e^{(f(\Theta') - f(\Theta^{(0)})/T}$.
3. After N_s loops through 1 and 2 step length vector V is adjusted in direction so that approx 50% of all moves are accepted.
4. After N_T loops through 1 and 3 temperature is reduced so that $T' = r_T \cdot T$ so that fewer downhill steps are accepted.

Simulated Annealing in MatLab

```
theta=[0.7,.8;]';
LB=[1,2;]';
UB=[-1,0;]';
sa_t= 5; %starting temperature
sa_rt=.5;
sa_nt=5;
sa_ns=20;
[xopt]=simannb( 'loglikeAR1data', theta, LB, UB,
sa_t, sa_rt, sa_nt, sa_ns, 1);
```

Code has three components

1. The main program that defines starting values for simulated annealing algorithm etc
2. A function that translates Θ into a state space system
3. A function that evaluates $\mathcal{L}(Z \mid \Theta)$

Point 2 and 3 are both done by `loglikeAR1data.m`

Simulated annealing output:

```
No.  of evaluations

 41

Current temperature

 5

Current optimum function value

 1.3994e+003

No.  of downhill steps

 4

No.  of accepted uphill steps

 0

No.  of rejections

 36

Current optimum vector

 0.6125

 0.8805
```

Simulated annealing output:

```
No.  of evaluations

 401

Current temperature

 2.5000

Current optimum function value

 1.3906e+003

No.  of downhill steps

 8

No.  of accepted uphill steps

 3

No.  of rejections

 189

Current optimum vector

 0.5246

 0.9565
```

Simulated annealing: Final output

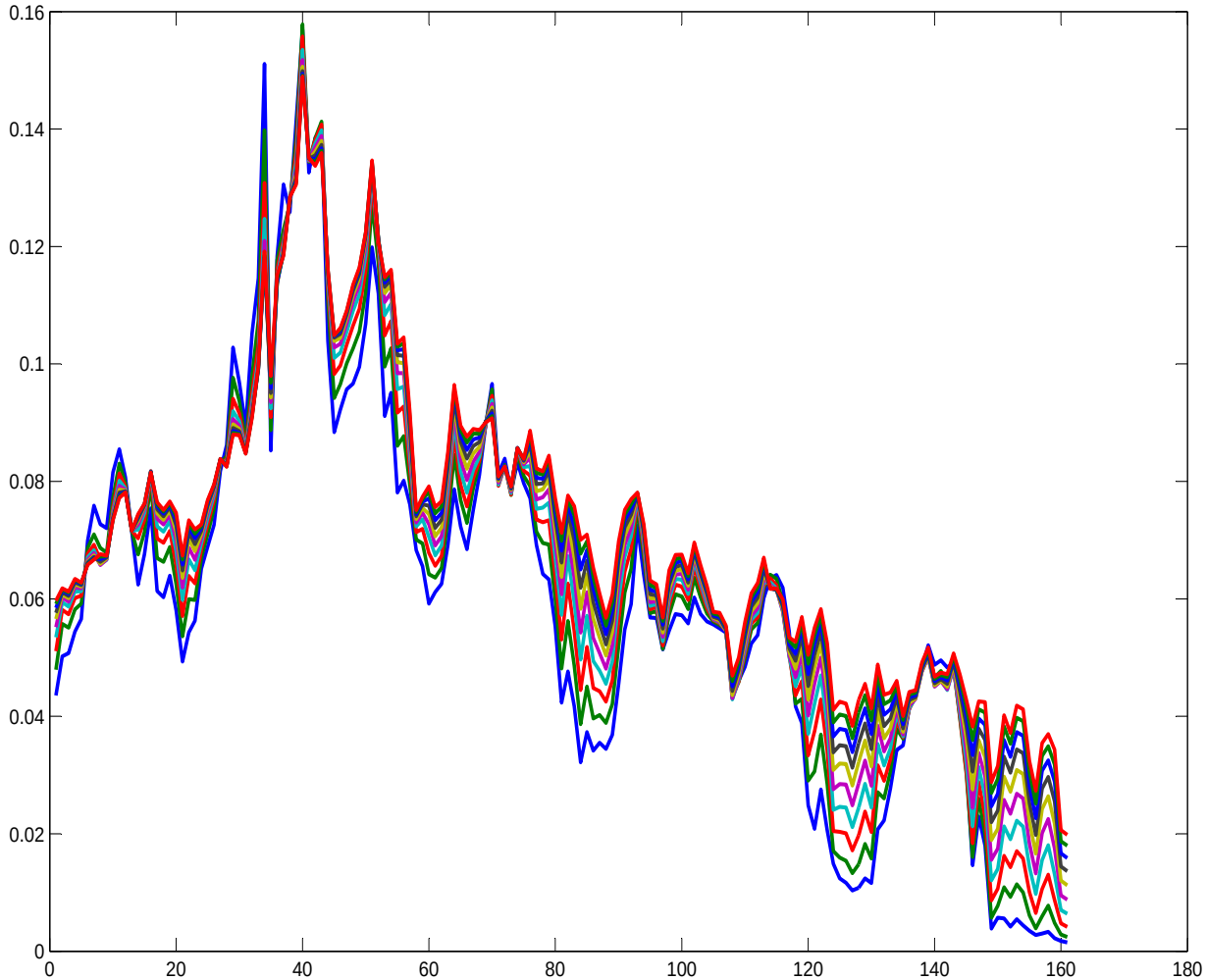
```
No.  of evaluations

 3201

Current temperature

 1.5259e-004

Current optimum function value

 1.3904e+003

No.  of downhill steps

 0

No.  of accepted uphill steps

 0

No.  of rejections

 200

Current optimum vector

 0.5095

 0.9467

Simulated annealing achieved termination after 3201 evals
```

Example: No-arbitrage term structure models

There is a large empirical finance literature using *affine no-arbitrage models* to model the term structure of interest rates.

- Uses latent factor models and no-arbitrage restrictions to estimate risk premia etc.

US Bond Yields



No-arbitrage term structure models

The departure point is the equilibrium condition for the price of an n period bond P_t^n

$$P_t^n = E_t \left[M_{t+1} P_{t+1}^{n-1} \right]$$

where M_{t+1} is the stochastic discount factor. In the absence of arbitrage, this relationship has to hold for all maturities n.

No-arbitrage term structure models

The next step is to make assumptions about the functional form of the SDF M_{t+1}. First, take logs of equilibrium condition

$$p_t^{n+1} = \log E\left[M_{t+1} P_{t+1}^n | \Omega_t\right]$$

and posit that the logarithm of the SDF follows

$$m_{t+1} = -r_t - \frac{1}{2}\Lambda_t' CC\prime\Lambda_t - \Lambda_t' C\varepsilon_{t+1}$$

where Λ_t is the price of risk vector given by

$$\Lambda_t = \Lambda_0 + \Lambda_X X_t$$

No-arbitrage term structure models

The one period risk-free rate r_t is an affine function of a vector of state variables X_t

$$r_t = \delta_0 + \delta_x' X_t$$

and the vector X_t follows a first order vector autoregression

$$X_{t+1} = AX_t + C\varepsilon_{t+1}$$

The approach uses that $r_t = -p_t^1$ and that in the absence of arbitrage any predictable return above r_t must be compensation for risk to estimate time variation in risk premia.

No-arbitrage term structure models

We can then write the log of bond prices as

$$p_t^n = A_n + B_n' X_t + v_t^n$$

$$\begin{aligned} A_{n+1} &= -\delta_0 + A_n + \frac{1}{2} B_n' C C' B_n - B_n' C C' \Lambda_0 \\ B_{n+1}' &= -\delta_X' + B_n' \mathbf{A} - (B_n' C C') \Lambda_X \end{aligned}$$

The recursions can be started from

$$A_1 = -\delta_0, B_1 = -\delta_X'$$

and the yield on an n-periods to maturity bond is given by

$$y_t^n = -\frac{1}{n} p_t^n$$

A State Space System

We have a state system of the form

$$X_{t+1} = AX_t + C\varepsilon_{t+1}$$

$$\begin{bmatrix} y_t^4 \\ \vdots \\ y_t^{40} \end{bmatrix} = \begin{bmatrix} A_4' \\ \vdots \\ A_{40}' \end{bmatrix} + \begin{bmatrix} B_4' \\ \vdots \\ B_{40}' \end{bmatrix} X_t + \mathbf{v}_t : \mathbf{v}_t \sim N(0, \sigma_v^2 \times I)$$

What is different?

- We have a vector of constants in the measurement equation
- The selector matrix is a function of "deeper" parameter, i.e. $D = f(A, C, \delta_X, \Lambda_0, \Lambda_X)$

We want to estimate $\Theta \in \left\{A, \delta_X, \Lambda_0, \Lambda_X, \sigma_v^2\right\}$

- The average short rate δ_0 can be difficult to identify so we will set $\delta_0 = 0.06$.
- A is normalized to be lower triangular and C is normalized to I.

A State Space System

We need:

1. A program that maps Θ into A, C, D, Σ_{vv} and d where

$$d = \begin{bmatrix} A'_4 \\ \vdots \\ A'_{40} \end{bmatrix}$$

2. A program that evaluates $\mathcal{L}(Z \mid \Theta)$
3. The program that loads the data and calls the simulated annealing maximizer.

Code for 1-3 are called `NoArb3fac.m`, `NoArb3facLL.m` and `NoArb3facSA.m`. The simulated annealing code is called `simannb.m`.

A State Space System

The ML estimates

$$\widehat{A} = \begin{bmatrix} 0.97 & 0 & 0 \\ -0.08 & 0.69 & 0 \\ -0.08 & 0.30 & 0.59 \end{bmatrix}, \widehat{\delta}_x = \begin{bmatrix} 0.42 & 1.32 & 0.12 \end{bmatrix}$$

$$\widehat{\Lambda}_0 = \begin{bmatrix} 2.87 \\ 24.7 \\ -337.6 \end{bmatrix}, \widehat{\Lambda}_X = \begin{bmatrix} 0.058 & 0.022 & -0.0034 \\ -0.0033 & 0.019 & 0.0013 \\ -0.21 & -0.19 & -0.14 \end{bmatrix}$$

$$\widehat{\sigma}_v^2 = 0.01$$

These can be used to compute an implied $\widehat{D}$.

State Space models and Principal Components

Previously, we used principal components to find the factors of a system that was in state space form

- What is the relation ship between the state X_t and the factors F_t
  - Can we find one from the other?
- When is state space models and filtering preferable to principal components?

Principal components factor models

Recall:

$$\begin{aligned} \underset{(r\times 1)}{F_t} &= \Phi F_{t-1} + \mathbf{u}_t^F \\ \underset{(N\times 1)}{Y_t} &= \underset{(N\times r)}{W}\ \underset{(r\times 1)}{F_t} + v_t \end{aligned}$$

- W contains the eigenvectors of $EY_tY_t' = W\Lambda W'$ and $WW' = I$ so that $F_t = W'Y_t$
- Λ is a diagonal matrix containing the ordered eigenvalues

This looks like a special case of a state space system

Can we find a mapping from X_t to F_t?

Yes:

- When N is large or Σ_{vv} is small the following holds:

$$\begin{aligned} F_t &= W' Y_t \\ &= W' D X_t \end{aligned}$$

If D is of rank n (where $n =$ dimension of X), then $(W'D)^{-1}$ exists so the mapping works in both directions.

How can we find W for a state space model?

$$\begin{aligned} EZ_t Z_t' &= D\Sigma_{xx} D' + \Sigma_{vv} \\ &= W \Lambda W' \end{aligned}$$

where Σ_{xx} solves

$$\Sigma_{xx} = A\Sigma_{xx} A' + CC'$$

Doing the eigenvector/value decomposition of $EZ_t Z_t'$ thus gives us W so that the factors can be computed as $F_t = W' D X_t$

How about the dynamics of the factors?

We have:

$$\begin{aligned} F_t &= \Phi F_{t-1} + \mathbf{u}_t^F \\ X_t &= AX_{t-1} + C\mathbf{u}_t \end{aligned}$$

To find Φ, use that $F_t = W'DX_t$ and $X_t = (W'D)^{-1} F_t$

$$\begin{aligned} F_t &= W'DAX_{t-1} + W'DC\mathbf{u}_t \\ &= W'DA\left(W'D\right)^{-1} F_{t-1} + W'DC\mathbf{u}_t \end{aligned}$$

to get

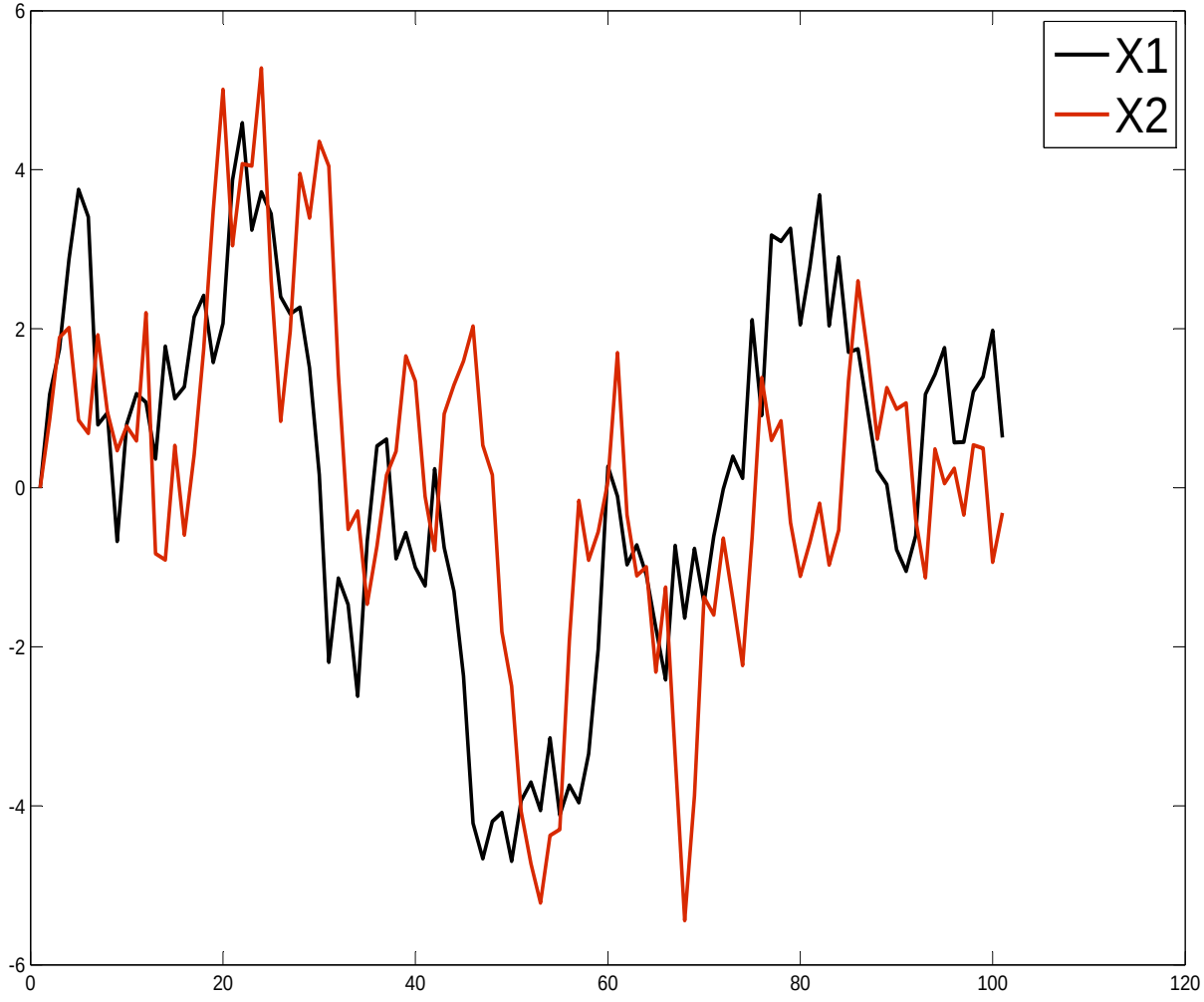
$$\Phi = W'DA\left(W'D\right)^{-1}, E\mathbf{u}_t^F\mathbf{u}_t'^F = W'DC\left(W'DC\right)'$$

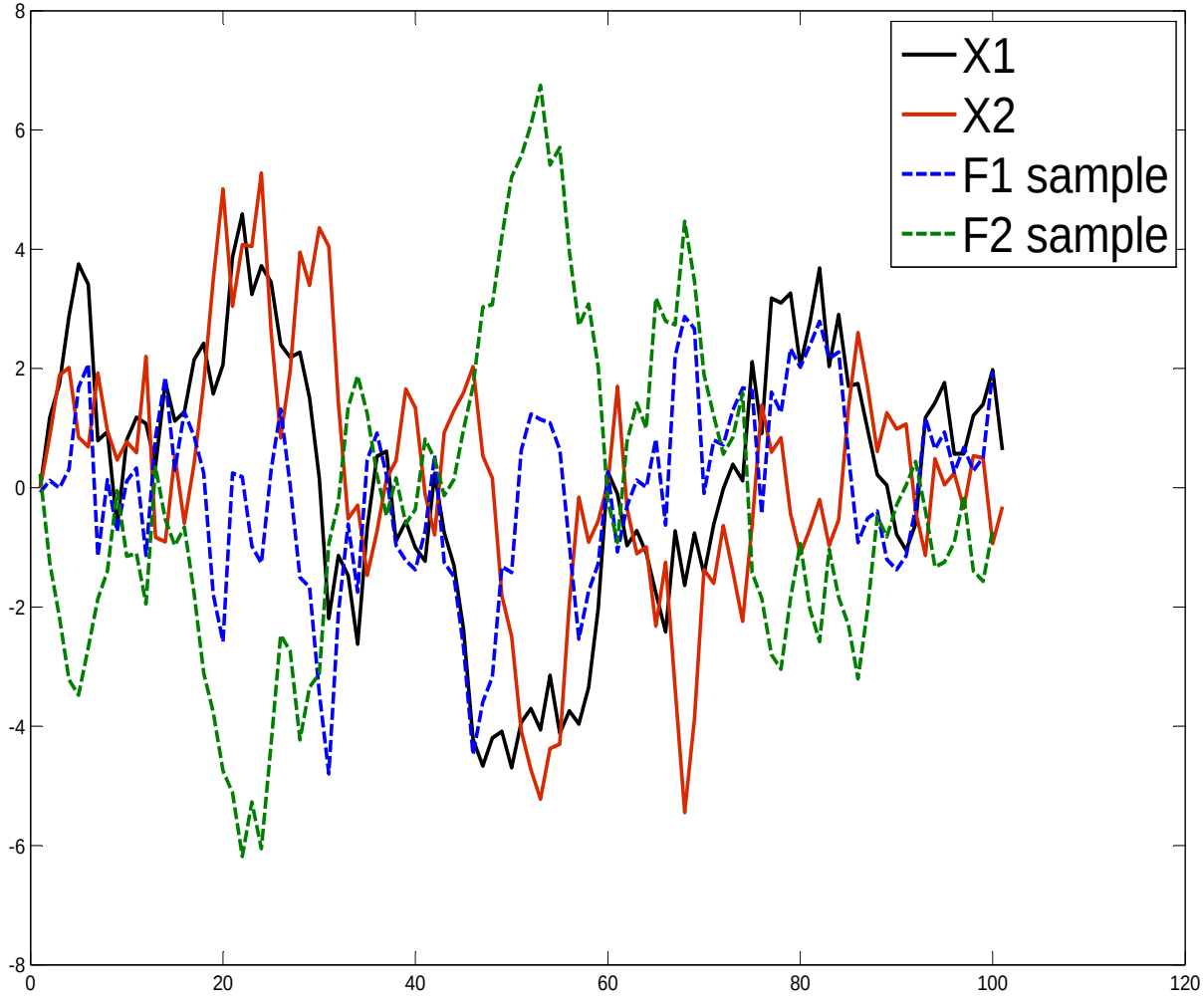
Example

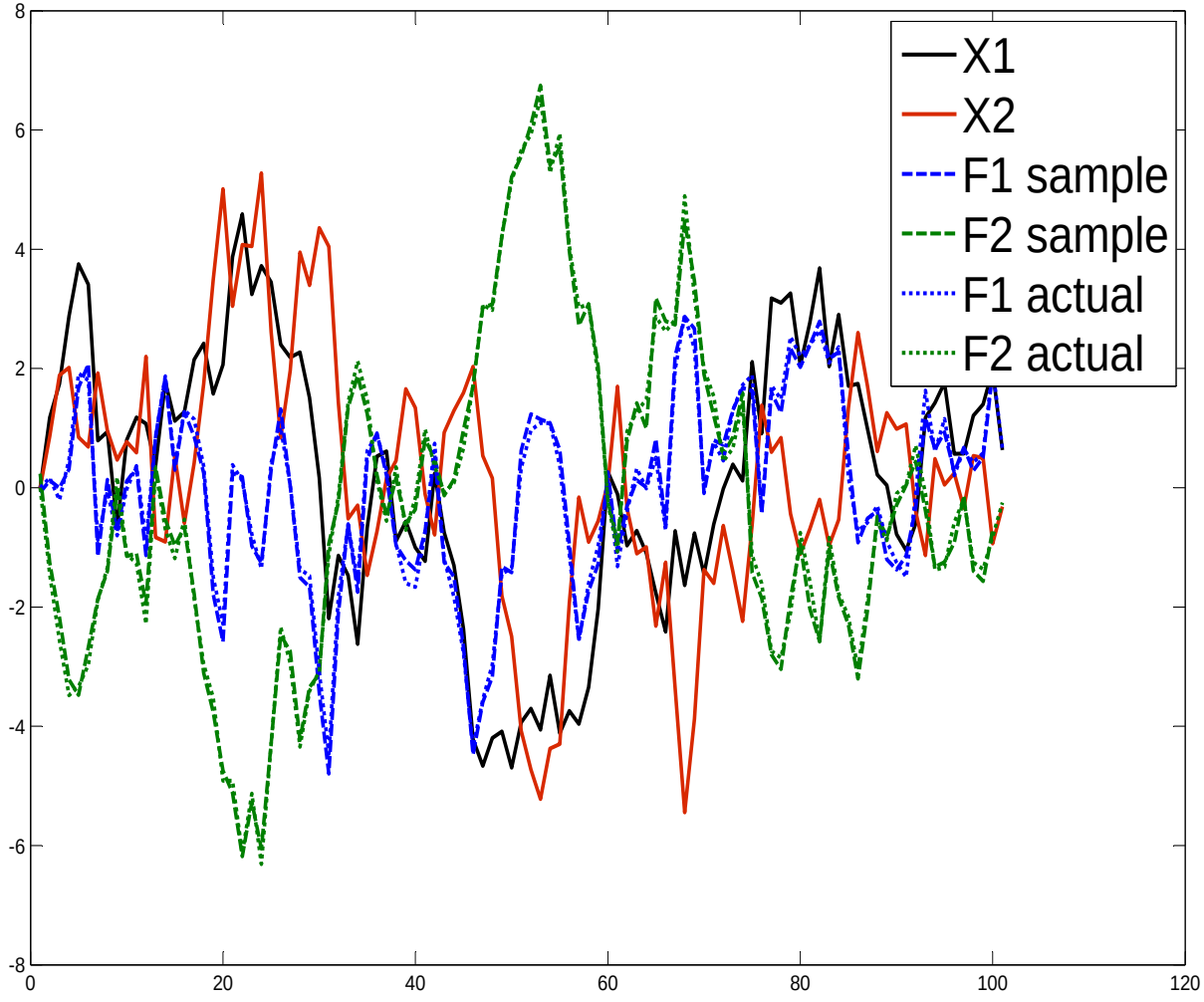
Simulate data using

$$\begin{aligned} A &= \begin{bmatrix} 0.9 & 0 \\ 0.2 & 0.7 \end{bmatrix} \\ C &= I, D = I, \Sigma_{vv} = 0.1 \times I \end{aligned}$$

with $T = 100$.


X1
X2
6
4
2
0
-2
-4
-6
0
20
40
60
80
100
120


X1
X2
F1 sample
F2 sample
8
6
4
2
0
-2
-4
-6
-8
0
20
40
60
80
100
120


X1
X2
F1 sample
F2 sample
F1 actual
F2 actual
8
6
4
2
0
-2
-4
-6
-8
0
20
40
60
80
100
120

T=100 sample and theoretical Ψ

Using $\Phi = W'DA\left(W'D\right)^{-1}$

$$\Phi = \begin{bmatrix} 0.69 & 0.008 \\ -0.19 & 0.91 \end{bmatrix}$$

$$\widehat{\Phi} = \begin{bmatrix} 0.71 & 0.025 \\ -0.19 & 0.93 \end{bmatrix}$$

Differences are due to small sample and non-zero measurement errors

Take homes from PCs and State Space models

State space systems that have the same implications for observables are not unique

- "Rotations" of state variables in X_t can give different interpretations
- Different rotations span the same space, so no difference in predictive content

Is this important?

- Depends on the question.
  - In factor models of the term structure, PC imply that the factors will have the level, slope and curvature interpretation.
  - In macro models, usually too few degrees of freedom to do any rotations since number of deep parameters is lower than free parameters in the state space system.

What is better when?

State space model and Kalman filter is better...

- ...if number of different observable time series is small...
- ...and measurement errors are large.

Time dimension is important when law of large numbers do not work in the cross-section

- If noise is small or number of time series very large, PC and SSM give the same results
- Choose whatever is more convenient

That's it for today.