Cont'-ing w/ the estimation — 25 July 2020

- despite adding SPF, still not identified

so I'm really thinking it must be a code issue

E.g. the arg moments still depend on shocks when N=100? Can that be?

Also start taking notes on cleaning out the TIPS from a lig. premium in blue, and notes on how to include the welfare mistake you make when you ass. RE instead of anchoring to the papers in brown.

Let's define some terms. — 26 July 2020

Breakeven inflation = difference btwn nominal & real yields of the same maturity

Fisher: $r = i - \pi \Rightarrow \pi = i - r$

$\Rightarrow \pi^{be} = i^{Tbill} - r^{TIPS}$

Andreasen et al say that

1. pos. bias in r^{TIPS} → neg. bias in π^{be}.

So the idea is

$$\pi^{be\,true} = i^{Tbill} - (r^{TIPS} - liq\,premium^{TIPS})$$

Now, the FRED π^{be} series (T10YIE) is constructed as

$$T10YIE = WGS10YR - DFII10$$

↑ 10-year Treasury ↑ 10-Year Treasury Inflation-Indexed Security (both constant maturity)

The question is if DFII10 is filtered for the liq. premium or not. Likely not.

A quick check for the liq premium could be

what Andreasen's model is able to match well:
the "model-free measure of the TIPS liq. premium, ... the difference between inflation-swap rates and TIPS-breakeven inflation."

<u>inflation-swap</u> := an agreement between 2 counter-parties to swap fixed-rate payments for a floating-rate payment linked to inflation.
i.e. to swap a fixed-rate to an inflation-indexed rate payment!

It sounds like the swap rate is a measure of expected inflation too, which is why

$$\pi^{be} \stackrel{!}{=} \pi^{swap}$$

so that if they're not equal, then

$$(\pi^{be} + \text{liq prem}) = \pi^{swap} \Rightarrow \pi^{swap} - \pi^{be}$$

= liq. premium.

This would be great if π^{swap} was a good measure of π-Exp. But it's not b/c the swap market is, although growing, very small. Still it's quite liquid (ho?) (Fleming & Sporn, 2013)

Investopedia: a zero-coupon inflation swap (the simplest form) is also known as a breakeven inflation swap

But it doesn't even seem like 27 July 2020
π-swap data is publicly accessible.
(A Cleveland Fed paper has it from Bloomberg, but I guess you need an account.)

Haubrich et al emphasize that risk premia may also contaminate things, and I'm scared that Andreasen et al ignore that.

So cont. w/ Andreasen et al.

liq. risk = identified as the difference between prices of principal & coupon payments

The Andreasen et al ATSM model (Section 3)

Caffine term structure model

$$r_t^N = \rho_0^N + (\rho_x^N)' X_t$$

- r_t^N: nominal short rate
- ρ_0^N: scalar
- ρ_x^N: $N \times 1$
- X_t: N pricing factors

X_t evolves as

$$dX_t = \underset{N\times N}{\kappa_x^Q} \left(\underset{N\times 1}{\theta_x^Q} - X_t\right) dt + \underset{N\times N}{\Sigma_x} \underset{N\times N}{\sqrt{S_{x,t}}}\, d\bar{W}_t^Q$$

where

$\bar{W}_t^Q$ is a standard Wiener process

$$[S_{x,t}]_{k,k} = \underset{N\times 1}{\delta_{0\,k,k}} + \underset{N\times 1}{\delta_{x,k}'} X_t$$

Price of nominal zero-coupon bond maturing at time $t+\tau$

$$P_t^N = \exp\{A^N(\tau) + B^N(\tau)' X_t\} \quad (3)$$

↓ some known ODEs ↙

In principle, TIPS (or other real bonds) could be priced like this, but that's not a good assumption given the low liquidity of the bonds market.

So instead they assume that liq. costs are present, and in fact in the following form:

$$r_t^{R,i} = \rho_0^R + (\rho_x^R)' X_t + \underbrace{h(t-t_0; i)}_{\text{increasing fct of time since issuance, to (latent factor)}} X_t^{liq} \quad (4)$$

↑ a time-varying liq costs

$Z_t \equiv [X_t', X_t^{liq}]$ we have an extended state vector which evolves according to a Wiener process. (5)

⇒ Price of a real zero-coupon bond maturing at T is:

$$P^{R,i}(t_0,t,T) = \exp\{A^{R,i}(t_0,t,T) + B^{R,i}(t_0,t,T)' Z_t\} \quad (6)$$

where A & B are given ODEs.

⇒ implied breakeven inflation rate from (3) & (6)

$$-\frac{1}{\tau}\log P_t^N(\tau) - \left(-\frac{1}{\tau}\log P_t^{R,i}(t_0,t,\underbrace{t+\tau}_{=:T})\right)$$

(which is a fancy way of saying $E\pi = i - r$)

$$= \frac{1}{\tau}\left[A^{R,i}(t_0,t,T) - A(\tau) - B(\tau)' X_t + B^{R,i}(t_0,t,T)' \begin{bmatrix} X_t \\ X_t^{hist} \end{bmatrix} \right]$$

Section 3.2. A Gaussian version of the ATSM w/ liquidity risk w/ closed-form expressions (!) for liquidity adjusted real prices

level factor ← , slope factor ←

$$r_t^N = L_t^N + S_t \quad (5)$$

for the real rate:

$$r_t^{R,i} = \mu^R + \alpha^R S_t + \beta^i \left(1 - e^{-\lambda^{L,i}(t-t_0)}\right) X_t^{liq} \quad (10)$$

- α^R: scales EIR
- $\beta^i > 0$
- $\lambda^{L,i} \geq 0$

functional form for $h(t-t_0; i)$

Interpretation of β^i and $\lambda^{L,i}$

Trading of TIPS happens in 2 phases:

Phase 1: bond i just issued, high supply but also high demand (low liquidity risk)

Phase 2: buy-and-hold investors have acquired their share of TIPS i, are sitting on them contently and the supply of bonds i for trading is scarce (high liq. risk)

$\lambda^{L,i}$ = determines length of Phase 1; a low $\lambda^{L,i}$ implies a long Phase 1, less exposure to X_t^{liq}

β^i := determines maximal exposure of i to X_t^{liq} in Phase 2.

Now, $z_t \equiv [L_t^N, S_t, C_t, L_t^R, X_t^{liq}]'$

$$\Delta z_t = \underbrace{\begin{bmatrix} K_x^Q & 0_{4\times 1} \\ 0_{1\times 4} & K_{liq}^Q \end{bmatrix}}_{K_z^Q \ 4\times 5} \left(\underbrace{\begin{bmatrix} 0_{4\times 1} \\ \theta_{liq}^Q \end{bmatrix}}_{\theta_z^Q \ 5\times 1} - z_t \right) + \Sigma_z \Delta W_t^Q \quad (11)$$

$$K_x^Q = \begin{bmatrix} \underline{0} & \underline{0} & \underline{0} & \underline{0} \\ \underline{0} & \underline{\lambda} & \underline{-\lambda} & \underline{0} \\ \underline{0} & \underline{0} & \underline{\lambda} & \underline{0} \\ \underline{0} & \underline{0} & \underline{0} & \underline{0} \end{bmatrix} \quad 4\times 4$$

Actually, when the model (β^i, λ^{Li} and a bunch of other things for all TIPS $i = 1, \ldots n_{TIPS}$) is estimated w/ an extended Kalman filter when you back out X_t^{liq} as a filtered state... na ne!

→ So let's try to understand the behavior of the liq premium and try to argue that it's not driving the dynamics of my figure 1.

p. 26 Mac

(1) The liq premium on TIPS is low and falling in general b/c i) the market is growing
ii) dealers are expanding their TIPS-trading

(2) The TIPS liq premium is higher in recessions, e.g. 2001 9/11 and 2008, but stabilizes afterwards (also partly thanks to QE)

(3) The 10-year TIPS exhibits a lower average liq premium and a less volatile one than the rest of the TIPS market

$mean(\psi_t^{10y}) = 30$ basis points vs $mean(\psi_t) = 38$

$sd(\psi_t^{10y}) = 13$ bp $\qquad sd(\psi_t) = 34$

(4) Older TIPS have a higher liq premium, but QE

mainly lowers their liq premium since the Fed mainly bought TIPS which were issued long ago.

Their dataset goes till Dec. 27, 2013.

Basis points = $\frac{1}{100}$ %, i.e. 0.01 %

So mean(γ_t^{10y}) = 30 bp → 0.3%

If I add a sd = 13 bp to it, 43 bp = 0.43%

So supp at 2020 Covid-shock, γ^{10y} rose by 2sd, i.e by 26 bp to 56 bp = 0.56%, then π^E is downward biased by 0.56 pp, so instead of π-Exp(·) of 0.5%, it'll be 1%.

1 pp = 100 bp

Since int. rates are usually changed by 25 bp, this liq premium is significant, but it doesn't change the message.

(5) Corr(γ, VIX index) = 0.67

↑ avg liq premium of Indicators at t

↑ # between 0 & 100.

Hnd: regressing γ on VIX gives a sig effect of 0.85**

1↑ in VIX → 0.85 bp↑ in γ

In 2020: VIX ↑ by 60 → 51 bp ↑ in γ.

Here is a VAR(1) 28 July 2020

$$y_t = \rho y_{t-1} + \varepsilon_t$$

What's its autocovariance at lag k?

$$E(y_t y_{t-k}) = ?$$

$$E(y_t y_{t-k}) = E(\rho y_{t-1} + \varepsilon_t)(\rho y_{t-1-k} + \varepsilon_{t-k})$$

$$E(y_t y_{t-k}) = \rho^2 E(y_{t-1} y_{t-1-k}) + \underbrace{E(\varepsilon_t \varepsilon_{t-k})}_{=0 \text{ if iid}}$$

$$\Gamma_k = \rho^2 \Gamma_k \quad \ddot{\sim}$$

Its VC-matrix is

$$E(y_t y_t) = (\rho y_{t-1} + \varepsilon_t)(\rho y_{t-1} + \varepsilon_t)$$

$$= \rho^2 E(y_{t-1} y_{t-1}) + E(\varepsilon_t \varepsilon_t)$$

$$\Sigma = \rho^2 \Sigma + Q$$

In this scalar case $\Sigma = (1-\rho^2)^{-1} Q$

→ Hamilton p. 67 Mac: j^{th} autocov of AR(1)

$$\gamma_j = E(y_t - \mu)(y_{t-j} - \mu)$$

Hamilton uses the MA(∞)-representation as

$$y_t = \rho y_{t-1} + \varepsilon_t$$

$$= \rho^2 y_{t-2} + \rho \varepsilon_{t-1} + \varepsilon_t$$

$$\ldots = \varepsilon_t + \rho \varepsilon_{t-1} + \rho^2 \varepsilon_{t-2} + \ldots$$

$$\gamma_j = \left(\varepsilon_t + \rho \varepsilon_{t-1} + \rho^2 \varepsilon_{t-2} + \ldots\right) \qquad j=1$$

$$\left(\varepsilon_{t-1} + \rho \varepsilon_{t-2} + \ldots\right)$$

$$= \left(\varepsilon_t + \rho\left(\varepsilon_{t-1} + \rho \varepsilon_{t+2} + \ldots\right)\right)\left(\varepsilon_{t-1} + \rho \varepsilon_{t-2} + \ldots\right)$$

$$= 0 + \rho\left(\varepsilon_{t-1} + \rho \varepsilon_{t+2} + \ldots\right)^2$$

$$= \rho^1\left(b^2 + \rho b^2 + \rho^2 b^2 + \ldots\right) = \rho^1 b^2 \left(1 + \rho + \rho^2 + \ldots\right)$$

$$= \frac{\rho \cdot \sigma^2}{1-\rho^2}$$

For AR(1), the j^{th} autocov is

$$\gamma_j = \hat{\rho}^j \cdot \frac{\sigma^2}{1-\rho^2} = F^j \cdot \Sigma \quad \checkmark$$

in the VAR(1) - notation of Hamilton,

eq. [10.2.21]

→ so it seems again that my Matlab code is fine.

→ in the multivariate VAR(1) case this is

$$\left(I_{np^2} - F \otimes F\right)^{-1} \cdot \underbrace{vec(Q)}_{\uparrow \sigma^2}$$

$\uparrow \rho$

The point is, the moments seem to be computed properly.

Peter meeting 28 feb 2020

1) F has to have eig (F) < 1

2) synthetic data: might be an issue of scaling
The logic of Gauss $W^{-1} = \begin{bmatrix} 2 \cdot 10^{-6} & & \\ & 1 \cdot 10^{-6} & \\ & & \ddots \end{bmatrix}$

then what matters is that 1 is two times that of 2. But in Matlab, numerically, this can be so close that it screws up. Take the smallest order of magnitude on diag and rescale w/ that so all diag el's are bigger > 1.

Could also scale the moment vector, altho careful then b/c it might cancel the multiple of W.

3) Take artificial data

4) Try a VAR(1) b/c then the VAR is more misspecied

5) Can instead of estimating the autocov & fit a parametric model to the data

↳ paper by Hansen, Hodrick, Singleton about how to compute the asymptotic std. error (i.e. W) w/o bootstrapping
↳ would require a lot of coding

6) Try again to plot the loss fct as a fct of a single α, when the others are fixed at the truth.

Then you can iteratively to do it with two, and for three just do the blue-blue-yellow-blue plots.

7) W^{-1}

In Matlab $\left(\begin{pmatrix} 1^{-10} & 0 \\ 0 & 100 \end{pmatrix}^{-1}\right)^{-1} \neq \begin{pmatrix} 1^{-10} & 0 \\ 0 & 100 \end{pmatrix}$

Haha! So that could easily cause problems.

Ryan meeting 29 July 2020

(Materials 38)

- Makes sense to explore the calibration of measurement error b/c they shouldn't be visible on the ACF.

  Shut off the moments for $E(\pi)$ and just add the meas. e. → compare ACF and see if meas. er. made things noise on its own.

- Maybe the meas. error causes a wedge between Nestim & Nsimul

- Fig 7. Don't reflect (re-)scaling of W in certain respects, e.g. 10^{-5} is smaller in magn. than the others, vs. bottom panel.

Top panel rejects flat model in ID b/c of flat regions.
Bottom panel ?!

Do Fig 7. of Mod 38 w/o expectations
- Why does the loss hit a hard 0?
- Why does rescaling change the shape?

↳ based on this picture and the others: a bug.

- Can only have as many moments as parameters in F & Q ⇒ so p=4 is better than p=1

2 reasons why stoch. singularity may not happen:

- Model is non-linear, so when you est a VAR w/ fixed coeffs it might not be stoch. singular

The other reason is that observables depend on past lags (of shocks) which the VAR doesn't see, and those act like new shocks

What could happen though is that F is very volatile. (And close to singular.)

What you can do?

Ridge $\lambda = 0.01$ small compared to $X'X$ then singularity

meas error: add a little.

How to hunt for bugs

4 August 2020

1) Investigate whether the measurement error is behind it.

1a) Shut off the Expectations / moments and look at autocovariogram w/ and w/o meas. error.

→ autocovariograms shouldn't change

1b) Same: plot loss for changing α's w/ and w/o meas. error and w/ and w/o rescaling

If the meas. error is behind it, then try getting it out by resorting to a ridge regression approach if stock singularity does become an issue.

No, keep Nsimulations so you're comparable.
~~Use Nsimulations as a default instead of Nstocks.~~

2) I want to do a general combing thru of various codes.

1a): If I take out the expectations-moments, W is no longer tiny, W^{-1} no longer explodes... → the expectations moments are the ones responsible for the "shrinking W" issue.

Another thing I note is that the moment J use the true data w/ meas. error, residuals become super-small. This is true even if you don't use meas. error in the true data, only in the estimation.

The measurement error does show up in the ACFs, → it changes the initial moments, but that's not a surprise. It shouldn't for Nsimul though, and it doesn't really either.

- Should it change the truth? Yes, I think so b/c that's just one simul.

1b) Losses indicate that

i) the rescaling thing is really an issue when $E(\cdot)$ are used → all losses look like the rescaled one.

ii) meas. error is scaling down losses a lot! why?

iii) α's at the edges look ID'd; but at the wrong place. (a little too low)

iv) Middle α's are not ID'd: float in the correct region.

v) $0-\alpha$ looks ID'ed at zero.

↳ why opposite of last week's?

↳ if ID'ed at zero, why isn't estim giving a zero there?

⇒ most likely b/c interaction w/ the ones that aren't ID'd.

↳ again, why is meas. error changing the loss?

⇒ I'm also wondering if actually this means that w/ more info that wouldn't be screwed up, we could ID everything?

I wonder why E(·)-moments have no variance.
Try setting $K=1$ (use only 2 lags, 0 and 1)
as the E(·)-moments have lowest variance at longer lags. Why?
W^{-1} still is ~ 05.

↳ not saving these figs since W^{-1} is still exploding and then I prefer to keep K the same.
Doesn't seem to improve a lot. The problem is that it isn't representative b/c W^{-1} exploding.

Check out the same for $K=4$.

↳ In both case Matlab doesn't complain when I

generate the data, but it does complain maybe six times while estimating that sthg is close to singular.

W^{-1} is on the order of $\epsilon 06$.

k=4 suggest (Nsimul) that the moments are matched very well, so previously it was indeed the meas. error that made them unmatched.

Also for the loss, $\alpha = 0$ seems to be reversed when you add $E(\cdot)$.

Checking codes — 5 Aug 2020

One thing I notice for Nsimul is that (N=100)

$loss(\alpha^{true}) > loss(\hat{\alpha})$

1736 > 944

N = 1000 ?

resnorm(α^{true}) = 1821

N = 10 000 ?

resnorm(α^{true}) = 1834

N = 10?

resnorm(α^{true}) = 1686

→ Is the loss increasing in N? It seems so!

Why?

I've checked and obj_GMM.m/gain_univariate_mean.m computes the moments exactly as command_acf_simulate_univariate.m does. The only diffs are that

i) command_acf... also computes the lag p, which is an input to obj_GMM...

ii) obj_GMM... does the computation N times and then takes a mean.

→ why is $res = (Om^{data} - mean(Om))\, W^{-1}$
increasing in N? → can only be if if mean(Om)
moves away from Om^{data} as N↑.

Check that acf saves the right things! ✓ Done.

obj_GMM_LDM gain_univariate.m is also correct;
it also does exactly the same thing that
command_acf... does.

The last thing to combine is sim_learnLM_dem_
approx_univariate.m
→ I still wonder if taking no inverse gains
would help "unclutter" the learning work
Especially as $k^{-1} = 0$ sometimes, which
set $k = \infty$, which, I believe, is
penalized in obj_GMM_LDM gain_univariate.m

"Uninverting it"

↳ sim_learnLTI_clean_approx_univariate_uninvertk.m

↳ fk_CEMP_uninvertk.m

↳ fk_CUSUM_vector_uninvertk.m

↳ fk_smooth_approx_univariate_uninvertk.m

Test it out in uninvertk.m

Haven't implemented "uninverted IRFs" but I obtain the same exog states, preps, k, shocked k, and convergence diffs.

Now estimate using the uninverting learning code

Nestimations: surprisingly, I get pretty much the same thing (see 1.2.3 in Materials 39).

Nsimulations: same :C, even more exactly the same thing! The loss is exactly the same.

↳ for this at least it got a loss of 740 instead of 780.

So for my discussions w/ Ryan & Peter:

1) Measurement error does impair the ability to match moments. (Fig 1 & Fig 2)
   - Both for Nestim & Nsimul
   - The loss becomes lots smaller if "data" has meas. error in it.
   - The m.e. seems to, strangely, give some ID-info. (seen nicely on Fig. 3)

2) Loss
   - The "W explodes"-issue is only there if E(·) are included. (Fig 4)
   - Fig 5 again: meas. error adds some ID info
   - Fig 6: adding E(·) adds huge problems: loss↑, directions reverse.

Peter meeting Patents 38 5 Aug 2020

- Fig 4: Shape shouldn't change shape from top to bottom.

  scalar / matrix (VC matrix of moments)

  $Y = \alpha X$

Check: $\varphi^{-1} = \frac{1}{\alpha} X^{-1}$

Check: $X^{-1}./Y^{-1}$ = matrix of α's

You are inverting a matrix at one point. If the inversion is done correctly then you should get $X X^{-1} = I$

Need to confirm that this isn't happening at one place.
→ That's where the error is.

- Measurement error: loss goes down

  Could multiply by step to make it go up.

• Fig 4. Supp Panel (a) is an accurate pic of loss fct.

1.) Ignoring the flat parts, the loss takes on a value very close to zero close to the true value

2) loss steeply sloped below truth
↳ do we see a U-shape when increasing the neighborhood of true?
If yes, then 1D had large std error b/c it converges to a diff value everytime.
⇒ check again

1D problem: checks also "absolutely flat" vs reaching a global min at the truth and being slightly bigger off to the right.

1D, bad misspecified

Meas. error:

Loss falls. Typically we think that a lower loss as a better fit.

But a model w/ & w/o m.e. doesn't correspond to each other.

Even if you add noise to data & loss ↓, even that can happen b/c it's a different dataset! Est-ing the same model on different data is not comparable.

2 possibilities:

1) params of model are ID-ed, but shallow bowls on the surface mean that the estim is challenging numerically, std errors large

↳ then let it run 10 000 times and say in an App. all runs. ——w/ diff starting values——→ std errors tell us that

estimated w/ uncertainty

2) not ID-ed w/ this data

then Bayesian est. of DSGEs the prior is doing a lot of the work

Ryan meeting — 5 Aug 2020

Bosted meetings 10.30 Tuesday.

meas error making est variances of moments larger, and therefore fitting the moments worse but giving a smaller loss,

the sense in which something is going wrong is

- if the solver quits too soon
- if the Jmeasure is too high & me. is playing too large a role

Check Fig 2: take α(row 1), plug into row 2

w/ m.l. → moments should look just as good as in row 1, just MatLab exited too soon.

- true data has no action in $E(\cdot)$
  → do a sim w/ more action.

Fig 5: most troublesome: loss is huge for the 0.025 params.

- not-locally: extremely curvy
- locally: very flat

↳ Need to zoom in and see what's happening

- The meas. error isn't behind any of this.
- We still don't know what is.

Work after

From Peter: check for the scaling where it
(P) occurs that $X'X^{-1} \neq I$ b/c that's where
the inversion encountered numerical difficulties.

From Ryan: a) plug params of Fig 2, top row
(R) into loss of bottom row (w/ n.e.)
should give the same loss (994)

b) generate a "truth" w/ scales
such that there's more variation in
expectations → solves w?

From both:
(Both) zoom in on loss plots.

(R) (a) moments do look just as good, except the
autocovariance of τ_c – that one's off.
↳ does this signal that there is stay fishy
w/ meas. error after all?

Worried about point (R)(a). 6 August 2020

(P): My hypothesis is that when I don't rescale, then the inversion causes conclusion. So then $W^{-1} \cdot W \neq I$. But $W^{-1} \cdot W = I$, indicating that the inverse was taken correctly. Ok, but also once I rescaled, also then $W^{-1} \cdot W = I$.

I also checked elementwise mult, and I get the same.

Check: $\varphi^{-1} = \frac{1}{\alpha} X^{-1}$ → not true everywhere!

Check: $X^{-1} ./ Y^{-1}$ = matrix of α's → true!

I'm surprised: it seems like X^{-1} $(= W^{-1})$ causes troubles, but if that's the case, I don't understand why $X^{-1} X = I$ and $X^{-1} / Y^{-1} = \alpha I$

→ Yes, inverting the unrescaled variance-matrix causes numerical issues, which disappear once you rescale.

(R)(b) A truth w/ more variance in E(·).

mg(2), mg(5), mg(0)

I'm also thinking that if this is not sufficient, widening the fe-space for truth may be a way to design larger fe.

Yeah, mg(2) still has small var(fe).
I also had VAR-problems (a lot)

mg(0) has no VAR problems.
Now I've widened fe-space to (-5, 5), mg(6), and now I have a bunch of VAR-problems. that has the same rescaling issue.

By the way, increasing the fe-space for the scene α lowers the movement in $E(\cdot)$ b/c it implies that bigger fe are associated w/ the scene gain.

So try $mg(0)$ w/ fe $\in (-0.5, 0.5)$, a <u>smaller</u> fe-space (which is the analogy of scaling α up)

$\rightarrow$ no VAR complaints

but the scaling issue seems to be there.

back to (P):

X = unscaled VC matrix

Y = rescaled VC matrix $\quad Y = a \cdot X$

$W_X^{-1} = X^{-1}$ weighting matrix from unscaled VC matrix

$W_Y^{-1} = Y^{-1}$ weighting matrix from rescaled VC matrix

The troubling thing is that if I choose any dd

$X = \text{randi}(5,5)$, set $Y = a \cdot X$

then $Y^{-1} \neq \frac{1}{a} X^{-1}$

In other words, for Matlab,

$$\frac{1}{a} X^{-1} \neq (aX)^{-1}$$

Now I'm starting to doubt my linear algebra.

But in a stanford lecture note, I find

$$(\alpha A)^{-1} = \frac{1}{\alpha} A^{-1}$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad-bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

$$\begin{bmatrix} \alpha a & \alpha b \\ \alpha c & \alpha d \end{bmatrix} = \frac{1}{\alpha^2(ad-bc)} \begin{bmatrix} \alpha d & -\alpha b \\ -\alpha c & -\alpha a \end{bmatrix} = \frac{1}{\alpha} A^{-1}$$

Ok. So Matlab has an issue? Or is $\frac{1}{\alpha}$ to close to 0? Yes! If $\frac{1}{\alpha} = 1e-5$, then matlab gets screwed up! ($a = 10^5$, $1e+5$)

If I set $a = 10$, then

$$y^{-1} = \frac{1}{a} x^{-1}$$

but <u>not</u> if I use $\mathrm{inv}(\cdot)$, only if I mk

$$1./Y = \left(\frac{1}{\alpha}\right)./X \qquad (!)$$

Let $X := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$ $\quad X^{-1} = \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$

$$= \frac{1}{4-6}\begin{bmatrix} 4 & -2 \\ -3 & 1 \end{bmatrix}$$

$$= -\frac{1}{2}\begin{bmatrix} 4 & -2 \\ -3 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -2 & 1 \\ 1.5 & -0.5 \end{bmatrix}$$

Ok now I'm super confused!

I've put (P), (R,a) & (R,b) aside 7 Aug 2020
and turn to (both), plotting the loss.

The good news is: the loss has minima everywhere,
and they're not too badly off either:

Now there are several questions I'm curious about:
1) What if I shut off the convexity moment?
2) What if I add the 0 at 0 restriction?
3) What if I add measurement error?
4) What if I rescale?
5) What if I add expectations?

The main message seems to be: w/o the expectations
variable, it seems to be weakly ID-ed if you add
the convexity ass. The meas. error just seems to screw
things up, so don't add it unless you must.

1) Removing convexity moment:

only impacts α_1 & α_5, making them oscillatory

at around 0.008 - 0.012 / 0.015

Why?

I'm surprised at 2 things:

1) Why these oscillations?

2) Why is there an explosive-looking limit at $\alpha_{1,5} = 0.006$

2) 0 at 0 does exactly nothing if convexity is added

- of course b/c $\alpha_3 = 0$ ✓

3) Meas. error: I now have 2 indications that the meas error aren't bookkeeper: 1) (R, a) 2) weird & wrong α_2 & α_4. Explosion and minima weird. So there may be a bug after all? → Avoid if can.

4) Rescaling actually does help: It puts the minima where they should be. But also here Real's danger of Matlab exiting too soon b/c loss becomes very small.

5) I don't know why expectations screw up everything. Is it b/c of lack of rescaling? → That's what Fig 4 of Materials 39 indicates!

To do: figure out the rescaling.

The rescaling helped both here and in Materials 39.

8 Aug 2020

↳ so need to figure out whether I'm doing it right.

↳ I wanna leave the measurement error be b/c I have so many problems, I don't feel I can resolve all of them. Those'll just have to wait.

↳ I'm surprised that screwed-up shift only screw up some α's. E.g. meas. err. screws up α_2 & α_4, removing convexity screws up α_1 & α_5

Why the oscillations and variation in α_1 & α_5 if you remove the convexity moment?

Maybe the oscillations come from the fact that there are a bunch of explosive fe's here. Maybe the loss is "tricked" into being small?

My feeling is that the way I treat explosive learning paths may be incorrect?

Rescaling didn't help there.

Instability around 0.01, local min at 0.02, increasing afterwards.

The problem is that even the instability can't explain why it's strictly increasing after ≈ 0.02.

It could be that that big f_c's don't really show up in the true simulation. It's true that bins of f_c out around ± 2 are small → goes back to Reyn's point (R, b).

↳ Maybe that also explains a part of why the loss is bigger at the truth than at estimated $\hat{\alpha}$: if there were no f_c errors and at the tail, then $\alpha_1, \alpha_3 \uparrow$ would lead the estimation to produce excess volatility compared to the data.

The thing is that this c.p. moving of α's shows that the default setting (w/o expectations) is ID-cl.

(w/o getting into the muddy waters of rescaling or incr. error) Why the estimation isn't getting there, then, is the question. And it might not b/c it sees a very "normal" fc-distribution, so it wants to assign a "mean" gain to those, and nothing bigger in the edges.

That might be it.

Now try: default, rescale W, set funtol = 1e-9.

B/c rescaling peaks at the right thing, just wanna make sure that Matlab doesn't exit too soon.

N=700 yet it stopped prematurely after 19 min, but it sure got closer to the truth.

Rescaling: One way I could bypass this issue is

Two-step feasible GMM:

1. Set $W = I$ and compute $\hat{\theta}^{GMM,1}$
   ↑ weighting matrix ↑ estimated params.
2. Calculate $W(\hat{\theta}^{GMM,1})$ and compute $\hat{\theta}^{GMM,2}$
   using this matrix.

First of all:

Let me redefine things a bit.

Let $\Sigma^{boot} :=$ Variance(moments) (or Σ)

Let $W^{-1} :=$ weighting matrix of GMM.

I've verified that $W^{-1} = \Sigma^{-1} = \begin{bmatrix} b_1 & & 0 \\ & \ddots & \\ 0 & & b_n \end{bmatrix}^{-1}$ for n moments. Note that for diagonal matrix A,

$$A^{-1} = \begin{bmatrix} \frac{1}{a_{11}} & & \\ & \frac{1}{a_{22}} & \\ & & \frac{1}{a_{nn}} \end{bmatrix}$$

Proof on 2x2: $A = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \Rightarrow A^{-1} = \frac{1}{ad-cb}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$

$A^{-1} = \frac{1}{ad}\begin{bmatrix} d & 0 \\ 0 & a \end{bmatrix} = \begin{bmatrix} \frac{1}{a} & 0 \\ 0 & \frac{1}{d} \end{bmatrix}$. So conceptually,

id shouldn't matter whether I take

inv(A) or 1./A in Matlab;

Well, except that the off-diag zeros introduce Infs.

So it should matter, lol.

Let W := rescaled Σ^{boot}.

$$W = 10^{lscale1} \cdot \Sigma^{boot}$$

$$W^{-1} = \left(10^{lscale1} \cdot \Sigma^{boot}\right)^{-1} \quad \text{(No expectations)}$$

$$inv(W^{-1}) \stackrel{!}{=} 10^{lscale1} \cdot \Sigma^{boot} \quad \text{True to e-12.}$$

$$W^{-1} \stackrel{!}{=} 10^{-lscale1} (\Sigma^{boot})^{-1} \quad \text{True to e-16.}$$

What I don't get is that why do I seem to be doing the rescaling right (not change the rel. size of the elements of W^{-1}) and still obtain different loss?

Add expectations:

$$inv(W^{-1}) \stackrel{!}{=} 10^{lscale1} \cdot \Sigma^{boot} \quad \text{True to e-7}$$

$$W^{-1} \stackrel{!}{=} 10^{-lscale1} (\Sigma^{boot})^{-1} \quad \text{True to e-16.}$$

$W^{-1} W - I = 0$ to e-15

$\Sigma^{-1} \Sigma - I = 0$ to e-15

→ but this gives 11 occurances of numerical nonzeros

while this gives 14

W1 · inv(W1) − I gives 10 nonzeros

W inv(W) − I gives 14

W1 · W − I gives 14

But the thing is that if rescaling changes things, and it is done correctly, as it seems to, then it must be taking (Σ^{-1}) screws up things. But $\Sigma^{-1}\Sigma - I = 0$ to e-15, so it doesn't seem to be the case.

Could it be that for higher Funtol, the non-rescaled version could also get the little more precise result? It got down to loss 944 after 1 min

so very quickly. But this is gonna mean that it rushed into the local sink and is now gonna be stuck there. But that means that the rescaling is changing the way it behaves.

Yes - it obtained the 0.44 loss and exited b/c the loss wasn't changing anymore ... so this unrescaled loss really thinks of this as a local sink. Which maybe the other would too but it just never got there.

→ Yes: it could be that the rescaling doesn't actually change much on the loss, except scales it down, and since the loss is so small, it doesn't find the right thing b/c exits or stops prematurely. Scale back up: obtain same $\hat{a}$. ✓

scale w, get w1, scale it back up, do we get unscaled w1 back? W/o expectations, yes to e-12.

W/ expectations: yes to e-11.

→ Something gets screwed up when you add expectations, and it may not just be the rescaling. Although it seems that rescaling when you add exp. has a bigger impact on the loss.

To do:
Print loss when add $E(\cdot)$ & rescale.

Also doing est. w/ Mat ?

9 Aug 2020

There was some complaint of VAR instability

W/o rescaling there's also a VAR instability complaint

→ Rescaling does the same as w/o $E(\cdot)$: it pushes out α_1 & α_5 to 0.05, doesn't change the others much (it does change α_1 a little)

→ Implemented ridge regression, first w/ $\lambda = 0.03$ then w/ $\lambda = 0.001$ b/c the bigger ridge parameter

wasn't supposed to get rid of the complaints.
But now it seems to have worked.

Note: I haven't been able to verify my ridge approach w/ Matlab's b/c the OLS structure of a VAR w/ lags is kinda difficult (eg has several columns).

It might be that the "error" from adding E(·) comes from the VAR-instability that arises from too little variance of the E(·) (-moments)
→ generating a new batch w/ more variance might help. But I still don't know why the rescaling changes the shape of the loss, and why for only some parameters.

↳ If anything, rescaling makes your fit to the E(·)-moments worse, and the other moments better.

→ That makes sense tho b/c rescaling tries to address the problem that Var(E(·)) is really small, so the weight on those moments is high.

→ Then also for data w/o E(·), what rescaling does is it shifts the weights a little away from the strings that have the lowest Var in the data.

Now try to generate the one data w/ ridge and estimate w/ ridge too.

Unfortunately the unrescaled W is still 1e+7.

I think that there isn't enough var in the data, in fact E(·) are too small.

The estimation has huge losses b/c the moments look quite different. Losses are huge and awful.

I'm starting to doubt the VAR-estimation code.

- Ok - so I still don't understand why the OCscaling is changing the <u>shape</u> of the LOO
- A feeling is that generating E(·) such that they're a) bigger b) more Var(·) would get rid of the W-issue, and maybe also the E(·)-issue.
- But the main point is that the E(·) <u>is</u> ID-el, even w/o E(·). The fact that it's not getting there needs to be addressed.

1) Impose 0 at 0 for Nsimul → better
2) Try W=I for Nsimul → kinda better
3) Try Wdiff32 ↑ for Nsimul. → w/ 1) & w/o needs more.
4) Check SVAR code, write own?
5) Generate new truth w/ more E(·)-actor (R,b)

Another issue was that Nsimul didn't look great or robust in the data (Materials38).

4) SVAR code

Looking at Ryan, Lect 5. Could it be that my ordering doesn't make sense? Not really b/c it only matters for A, the impulse matrix. But I don't need that.

I found a code rdeform_var.m, so I'll just do my own, rf_var.m.

Looking at Lecture, Sum Part 2.

- $\hat{\beta}_{GLS} = (X'\Omega^{-1}X)^{-1}(X'\Omega^{-1}Y)$

  $\hat{\beta}_{FGLS} = (X'\hat{\Omega}^{-1}X)^{-1}(X'\hat{\Omega}^{-1}Y)$

  where Ω := VC matrix of errors.

Useful when off-diags of Ω aren't zero, i.e. errors are heteroscedastic and/or correlated.

- SUR is like separate OLS's stacked on top of each other and estimated via feasible GLS.

But I'm not sure I actually need to do that b/c OLS of an AR is biased but consistent.

Suppose from Time Series Lect 5 that

$$z_t = \beta z_{t-1} + \varepsilon_t \qquad AR(1)$$

then $Y_t = z_t \qquad X_t = z_{t-1}$

$$\hat{\beta}^{ols} = (X'X)^{-1}X'Y$$

AR(2) $z_t = \beta_1 z_{t-1} + \beta_2 z_{t-2} + \varepsilon_t$

then $Y_t = z_t \qquad X_t = [z_{t-1}, z_{t-2}]$

$$\hat{\beta}^{ols} = (X'X)^{-1}(X'Y)$$

VAR(1) $\underset{T\times 2}{z_t} = \underset{T\times 2}{\beta}\, z_{t-1} + \varepsilon_t$

$$z_{1,t} = \beta_{11} z_{1,t-1} + \beta_{12} z_{2,t-1} + \varepsilon_{1,t}$$

$$z_{2,t} = \beta_{21} z_{1,t-1} + \beta_{22} z_{2,t-1} + \varepsilon_{2,t}$$

$$\begin{pmatrix} z_{1,1} \\ \vdots \\ z_{1,T} \\ z_{2,1} \\ \vdots \\ z_{2,T} \end{pmatrix} = \left(\begin{array}{ccc:cc} z_{1,0} & z_{2,0} & & & \\ \vdots & \vdots & & 0 & \\ z_{1,T-1} & z_{2,T-1} & & & \\ \hdashline & & & z_{1,0} & z_{2,0} \\ & 0 & & \vdots & \vdots \\ & & & z_{1,T-1} & z_{2,T-1} \end{array}\right) \underset{4\times 4}{\beta} + e$$

In my case, there are 4 eqs and 4 lags → 8 columns = nvar · p

$$\begin{bmatrix} z_{1,5} \\ \vdots \\ z_{1,T} \\ z_{2,5} \\ \vdots \\ z_{2,T} \\ z_{3,5} \\ \vdots \\ z_{3,T} \\ z_{4,5} \\ \vdots \\ z_{4,T} \end{bmatrix} = \begin{bmatrix} z_{1,4} & z_{2,4} & z_{3,4} & z_{4,4} & \text{and so for lag 1, 3 \& 4} \\ \vdots & \vdots & \vdots & \vdots & \\ z_{1,T-1} & z_{2,T-1} & z_{3,T-1} & z_{4,T-1} & \end{bmatrix}$$

$(T-p)\cdot p \times 1$ $\quad (T-p)p \times (nvar \cdot nvar \cdot p)$ $\quad (T-p)p \cdot (nvar \cdot p)$

$(T-p)p \times (nvar^2 \cdot p)$

which would give $(nvar^2 \cdot p) \times (nvar^2 \cdot p)$

B^{ols} ! That can't be!

Ok but what if you recognize that you're repeating regressors?

B^{ols} would be $(nvar \cdot p) \times (nvar \cdot p)$

Now I'm thinking that having a 10 August 2020
2-dimensional Y- matrix in the VAR regressand corresponds to estimating the VAR equation-by-equation via OLS.
But we can try to verify that. ✓
→ Verified that sr_var.m & reduform_var.m are correct. My new code, rf_var.m is the basis for comparison b/c it computes the regressors w/o recourse to eval.
(command_verify_VAR_OLS.m)

Back to (R,b): a truth w/ more action in expectations
Some thoughts before I begin:
1) Widening fe-range on its own *lowers* the action b/c it's equivalent to scaling down α's.

2) Scaling up α's on its own should heighten the action

3) Putting the two together is complex: I guess the action should increase if $\alpha\uparrow$ is strong enough.

Oh shit - ridge results weren't good b/c the true fc-support was $(-0.5, 0.5)$ here!