MOVING THE NEXT NOTES 10 August 2020

to the iPad.

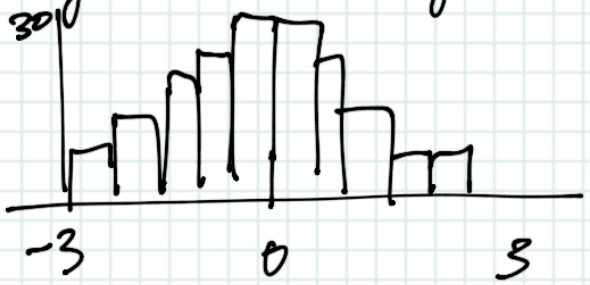
So: trying to generate truth w/ more action in expectations (R, b).

Observed:

1) If you increase the fc-support, fc's in the simulation become more tight around zero, and extrema are smaller. (Eg. $fc \in (-2,2)$ vs. $fc \in (-0.5, 0.5)$.

2) If you scale up α's, fc-distribution becomes more spread out: fatter tails

E.g. 4. α^{true} gives the following fc-distrib



instead of something like

and $\bar{\alpha}$ fluctuates much more.

Estimating 4. α

- loss was initially much smaller 4.95 instead of on the order of 4000.
- estimation gets a lot of "fe was nan" messages and also a new one: "obj function returned NaN, trying a new point". Hm...
  - ↳ I already got this message 9 times
- It took 585 sec.
- Implied k^{-1} is <0 4 times!
  - → maybe that's for negative k?
- Again, $\text{loss}(\hat{\alpha}) = 3.2 < \text{loss}(\alpha^{\text{true}}) = 416$.
- Not possible to evaluate loss function?!
  - ↳ It seems to be nans everywhere?
    Is that why the loss fct returned nan b/c all simulations were nan in the cross-sections?
- Look at the loss: it sucks:
  - for α_1, α_5, it's apparently always nan
  - for α_2, α_4 it blows up
  - for α_3, it's oscillatory.

Any $\alpha > 0.5$ gives all nans. Maybe that's b/c since the truth is higher, it more quickly explodes?

- I'm not thrilled that many histories explode.
- Also very many VAR instability problems.

α_3 do more or less peak at the truth even now. (well, α_3 doesn't and α_1 & α_5 have 2 minima around the one value, but still)

→ Does this indicate that the data-generation is wrong or the estimation?

- True moments changed very much:
  - moments pertaining to x scaled up, ones pertaining to π or i scaled down / were unchanged
  - own moments or cross-moments w/ π or i have the same shape
  - cross-moments w/ x have changed shape: are now more U- or reverse U-shaped.

↳ I think these make sense b/c since x is low, and $i = 1.5\pi$, x is the one that absorbs movements in expectations, while the pass-thru to π and to i is small.

⇒ The data-generation I think is correct. But this kinda suggests that the mean moments are also (more or

less) computed correctly, of which I'm not convinced.

↳ Let's try an estim w/ $\psi_x = 0.5$.

- But suppose the data is generated correctly. Why isn't the "x absorbs all" mechanism matchable by the estimation? I mean, moments didn't move far from initial ones.

  ↳ Explosion % reveals that in the estimation, either 0% or 100% of the N simulations explode. No, sometimes it's 14, 79 or 99%. And it seems that the obj fct returns 'nan' when 100% explode. Yes.

  I learn also, by the way, that each iter of the solver evaluates the loss function exactly 6 times.

  ↳ Why? It's not changing α, or if it is, only one and only marginally

  I'm also noticing that it's only considering α's really close to α_0 ⇒ multistab!

Ask Ryan

I'm also noticing that the simulation is not giving the 'if was nan' error most of the time, even though the solver gets a positive expl_percent. Hm!

When the "fe was nan" error comes, then the explosion-counter does catch it. But apparently there are a bunch of explosions that come not from there.

Ok, it's clear: when fe is nan, it's always caught. The other thing that causes "explosive" simulations is $k^{-1} < 0$. These get caught and register as explosions, but aren't really.

⇒ My check for "global nonnegativity" was wasn't catching everything!

Her ho: that's where the troubles start.

⇒ Initially, fe is nan (maybe the gain is too high) but then the bulk of the supposed explosions come from $k < 0$.

Ok... even when fe is nan it seems it's b/c or at the same time as $k < 0$ ...

⇒ It's at the same time. The initial iteration involves 5% "fe was nan" (5/100 simulations) but 100% have $k < 0$ at one point!

Let's see for the July 6 dataset ... and yes,

the initial iters involve 1% fe nan
69% k<0

and later iters have 2% fe nan
86% k<0

Converges after 23 iters.

Plus no more problems after iter 4.

Why is fe often nan early in the search and not late? Is it b/c of the initial values α_0?

And could a finer, and more broad fe-grid for the global nonnegativity test work?

↳ I introduced the variable broaden=2, which extends the fegrid-fine by ± broaden at both ends. It seems like this catches & subsumes all the previous errors!

It still finds the same min though :S
(But at least it's 2/3 of the time)

If I make the loss fct = nan when k<0 globally, I can engineer the "Obj fct returned nan" error message.

Does this influence the estimate? (Previously I had set the objective to 1e+10) No it doesn't.
→ I don't think so but it does affect the plots of the loss I would think!

Let's go back to the scaled-up truth and plot the loss
Should understand why $k<0$: is it, as I think, that d isn't convex and so a f_e outside the grid is extrapolated to a $k<0$?

The other thing is that this doesn't seem to really affect the behavior of the loss fct. It still converges to the same (wrong) thing. Actually, for the 4.0 truth, it converges to sthg different, but most likely that's b/c it took an avg of a smaller set.

The loss: $k<0$ and f_e=nan return. Does that mean that fegrid-fine needs to be even broader?
Not necessarily b/c now it happens that e.g.
f_e=nan in 14%
but $k<0$ in 0%.
In fact, $k<0$ doesn't happen a lot! f_e is the problem here!

Why is it that for the estimation, $k_E \approx 0$ is the root of all evil, but in plotting the loss, suddenly $f_E = \text{nan}$ becomes the problem #1? Is it that the estim avoids high α's (or maybe also very low ones) that could cause turmoil?

Materials 40, Section 4: Look into behavior of simulation. 12 August 2020

I think I learn a lot from this. Fig 7, which shows means & distros of k^{-1}, $\bar{\pi}$ and f_E are indicative.

1) While k^{-1} spends a lot of time being near 0 (see histogram), $\text{mean}(k^{-1}) \approx 0.0185$ (see means)
↳ If the moments capture the latter aspect, then the estim should have a hard time pushing α_s down. But the moments should more capture the first. Moreover, what does it matter if $\text{mean}(k^{-1}) = 0.019$? It still can enter the lower parts of its state space, and it does! So scrap that.

2) The cross-sectional mean of $\bar{\pi}$ & of f_E ↓ as N↑. Maybe this is why loss↑ in N?
I'm not sure this should happen? ⟶

As $N \to \infty$, $\epsilon \to 0$. So $f_\epsilon \to 0$ and thus $\bar{\alpha}$ too.

But that means that mean moments will reflect a situation in which $\bar{\alpha}$ isn't moving much, and f_ϵ are very small.

Now turn to changing α's: do the following exercises:

- $\eta \cdot \alpha$
- Move edges: α_1 & α_5
- Move middle: α_2 & α_4
- Move 0-point: α_3

Obs. 1: If all α's ≤ 0.1, simulation doesn't blow up. At 0.11, they do. (2 occurances of "f_ϵ was nan", one of which was also $k_{t-1} < 0$)

Obs. 2: $\alpha = 0$ doesn't cause explosions.

Obs 3: In the early scenarios, "f_ϵ was nan" is the only message. In the late ones, it comes together w/ "$k_{t-1} < 0$". Does this mean that $k < 0$ when α_3 is large?

↳ Making the code output α suggests that:

- "f_ϵ was nan" alone occurs when α_1 & $\alpha_5 \geq 0.11$
- "f_ϵ was nan" + "$k_{t-1} < 0$" occurs when α_2 & $\alpha_4 \geq 0.11$

- α_3 doesn't shift any moments: it can also be 0.5, that doesn't cause explosions. (0-neighborhood indifference problem)
- these seem to occur across shock histories.

Now let's look at the plots of the means:

Scenario 1: Varying α_1 and α_5: 0, 0.05, 0.1

- Shifts k^{-1} up from (0.01, 0.012) to (0.02, 0.03)
- Shifts $\bar{\pi}$ from (−0.01, 0.01) to (−0.02, 0.04)
- Barely shifts fe: it remains in the (−0.2, 0.2) range

Scenario 2: Varying α_2 and α_4

- Shifts k^{-1} up from (0.005, 0.001) to (0.05, 0.06)
- Barely shifts $\bar{\pi}$: it remains in the (−0.01, 0.03) range
- Barely shifts fe: it remains in the (−0.2, 0.2) range

Scenario 3: Varying α_3:

- Shifts k^{-1} up from (0.015, 0.02) to (0.05, 0.06)
- Barely shifts $\bar{\pi}$: it remains in the (−0.01, 0.03) range
- Barely shifts fe: it remains in the (−0.2, 0.2) range

↳ So: why aren't $\alpha_{2,3,4}$ moving $\bar{\pi}$ & fe?

↳ is it b/c in that range k^{-1}. fe is a really small number? I.e.: is the 0-neighborhood indifference a problem

here too?

→ I think so! I think what is happening is that the entire $f_E \in (-1, 1)$-region yields too small $b^{-1} \cdot f_E$ products such that they do matter a little bit (the loss does take a min there) but not a lot (the loss is very flat)

⇒ To be identified, you need to consider α's associated w/ $f_E > 1$ (possibly even greater). But: you also need a prior that's not too big, b/c $\alpha > 0.1$ seems to be ill tolerated by the model.

→ To DO!

For Peter meeting:

① Loss does have a min, almost at right spot. (Fig 1)

a) Need convexity assumption – don't get removal (Fig. 2)

b) Rescaling W affects loss, although no indication of inversion issues. (Fig 2.b)

c) Adding $E(\cdot)$ screws things up in a way I don't get. (Fig 4)

(② Initialize w/ more action in $E(\cdot)$) (Fig 6)

I thought that if $E(\cdot)$ aren't moving, then both rescaling

and informativeness of moments might be scattered up.
Explosions hinted at something else.
$\alpha \geq 0.1$ to avoid explosions.

③ Behavior of simulation

a) Loss↑ in N↑ b/c $\bar{\pi}$ & f_e ↓ in α? (Fig 7 b&d)

b) Fig 8: Only α_1 & α_5 can affect $\bar{\pi}$, and even that can't affect f_e. (Fig 8)

Peter meeting

11 Aug 2020

Fig 1. Loss higher at $\hat{\alpha}$ vs α^{true}

Again: b/c data from model is nonlinear, this may not suggest that α^{true} is a local min.

Thinks that a lot of problems come from that est a nonstat model is tricky.

The next issue: holding α's fixed, but estimating more than one α may not work. This would happen if you est $Y = \beta_1 X_1 + \beta_2 X_2$ and X_1 & X_2 multicollinear.

↳ might be some weird interaction between α's so that the convexity restriction helps provide curvature.

Fig 8 conclusions sound exactly right.

Rescaling: still quite strange

- Fix α, check loss w/ and w/o scaling → should recover the scaling factor.

→ Also explains why imposing convexity helps.

- Wanna guard against explaining what is really a coding error. → Should go back to that.

The level of loss fct has no meaning; only the curvature.

Min in Fig 1. don't line up w/ truth: not a problem b/c if you sim using a diff ℓ, you'd get a diff min.

If std. errs large, can be b/c of
- ID → doesn't go away as $N \to \infty$
- sampling error → goes away as $N \to \infty$

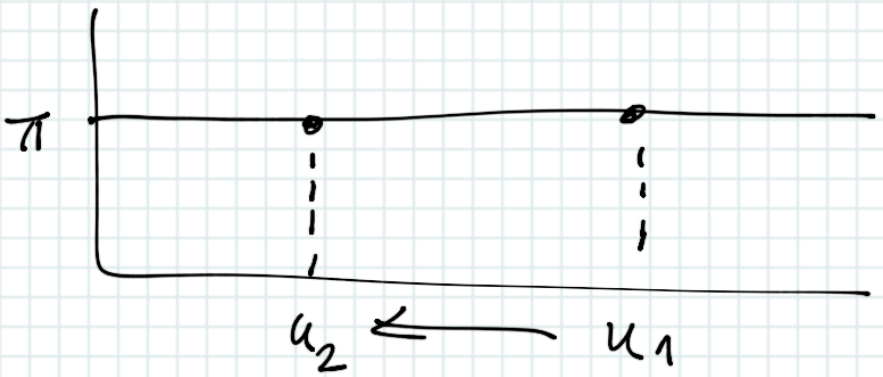
Recall the logistic functional specification for anchoring fct: somehow more structure has to be imposed on the anchoring function to guide the inf in the data

Analogy: Selena?

Silvana Tenreyro: $\hat{PC}$: slope is ≈ 0 → b/c CB is credible! So est a PC w/ these data doesn't tell you what would happen if CB abandoned target. (b/c that's just not in the data!)

→ In obs that have modest fe, we don't learn a lot about how agents learn from forecast errors.

"flat PC": "π isn't moving in tandem w/ u anymore"



↳ he says that the "flat PC" term refers to the idea that π doesn't follow u, not the other way around!

low u doesn't lead to any diff π!

- A full-basis approach: a simpler functional form for anch. fct.
- Or: pick unconditional moments w/o VAR or weighting matrix

We called it an in-between btwn est & calibration.

"Moment-matching": take sample moments & match those.

Successful est. will involve placing restrictions on 1st OR on the anchoring pt to allow to glean info that is in the data.

↳ And I'm not sure why he calls the "moment-matching" by that name: it's not much different than my SMM.

Let's try the thing w/ α's out in the edges. 12 August 2020

Attempt ①: July 6 data, fegrid = [-2, -1.5, 0, 1.5, 2]
(truth same) converged, not much change.

Attempt ②: July 6 data, fegrid = [-4, -3, 0, 3, 4]
Took 500 sec & converged to string guide diff!
Not correct but maybe getting near?

Attempt ③: July 6 data, fegrid = [-4, -3 ; 3, 4] wfe=4
Took 115 sec & converged to string very diff.
Isn't correct either but the thing is that the "truth" doesn't have a lot of fe in the 3-4"6 range.

Attempt ④: July 6 data, fegrid = [-4, -3, 0, 3, 4]
but w/ Wmid=1000.
Took 233 sec. Looks good. fe ±3 & 4 are still off, but I'm hopeful.

Attempt ⑤: generate truth with large forecast errors which you can est. w/ fcgrid = [-4, -3, 0, 3, 4]

First try Avg 10 data ($4 \cdot \alpha^{true}$)

(w/ Wmid = 7000)

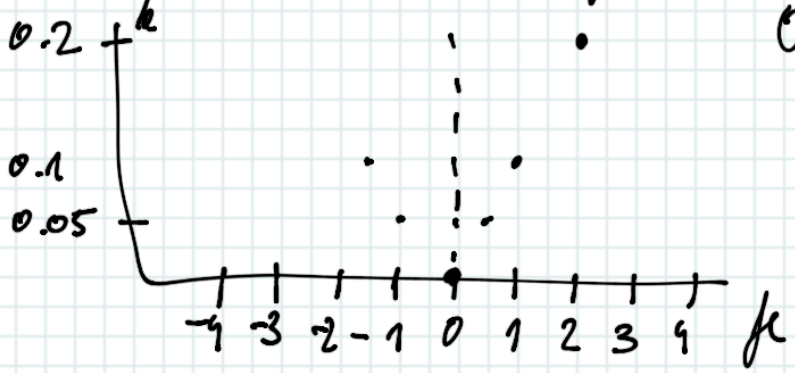
I'm also noting that the connectivity thingy ain't working well.

Took 643 sec.

Got a bunch of "3% of sim histories exploded" messages.

There we go, lookin' good!

~~Attempt~~ ⑥: Make α's associated w/ small fc large, so the truth looks something like:



Oh shit. This is exactly the 4 x times thing!

And $5 \times \alpha^{true}$ explodes already.

I don't think I can generate a dataset w/ bigger fc b/c large fc need large α's, but sufficiently large α's also explode, so...

Real data with Attempt ⑤ settings leads to a 1-2% sim histories exploded sim. 390 sec.

I think I'm going to leave it at that and consider the rescaling issue. Can I recover the scaling factor? Aaand... no. Actually yes, if you recall that the loss involves squaring W.

↳ Ask Ryan: could it make a diff that in Lagrangian you specify the equations residls - W?

Applied Young Economists Webinar (AYEW) 12 May 2022

Nan Hu & Zexi Sun "Vague Talking at Central Banks' Press Conference: News or Noise?" (19 participants)

Hong Kong Institute for Monetary and Financial Research (←node)

- Vague talking that includes words like "risk", "uncertainty", "volatility" and "turbulence" raises stock returns
  - ⇒ b/c investors expect expansionary monpol in the future: the ⊕ news dominate the ⊖ news that "current situ is bad".
  - Expansionary disinflation (Ball 1994)

(independent vs. news vs. noise)

dictionary from Loughran & ..., freq. of uncertain words

Regression: $R_t = b_0 + b_1 \Delta \text{vaguetalking}_t + \int' b X_t + \varepsilon_t$

First diff b/c EMH: stocks should only react to <u>surprise</u>

Prepare Ryan meeting:

① Loss has a min at truth (Fig 1.)

② Loss is indifferent to α's such that $|f_e| \leq 1$ (Fig. 8)

③ Having a more active in f_e truth and est-ing α's corresponding to large f_e modes (Fig 15)

Details & problems:

④ Solver evaluates loss 6 times for the same α?

⑤ Rescaling: changes shape of loss while no sign of inversion issue?

Ryan meeting 12 August 2020

- Need to est ≥ 6 of shocks in order to get model-implied moments closer to those of the data, at least for π. → or play around w/ ϵ_m.
  - ↳ If shocks too small, no f_e ⇒ ID problems
- Isn't the whole story, b/c eigenpairs also off
- Fig 8: would learn more if you plotted the auto-covariograms for these exercises!

→ So what's still amiss? We still don't know.

Weighting matrix: Eq (2): multiply by $\sqrt{\text{diag}(w)}$

→ Take the $\sqrt{\cdot}$ b/c otherwise you may put all weight on one moment.

- My rescaling strategy likely not to solve inversion issues b/c it's the ratio $\frac{\max(\text{element})}{\min(\text{element})}$ of elements of Σ ✓ which matters. If it's more than $10e6$, then problem.

↳ Ryan's strategy: Fix largest el. of Σ and fix all elements to be not smaller than $1/10\,000$ the largest.

- Not aligning the right els of W w/ the right moments could be a potential problem.
- Search alg:

  6 evals b/c it computes gradient & jacobian.

  (by 100 say) You can multiply the step for eval of $\frac{\partial}{\partial}$, then the est. of derivatives is less precise but you get a better idea of the slope.

  "first / fwd difference" or otw in matlab

Search algorithm evaluates loss 6 times for each guess b/c it 1.) evaluates at the guess

2) Changes each α_i a LITTLE bit to estimate

first and second derivatives. It changes one α_i at a time, so this gives 5 additional evaluations.

Main point: the story that we can't use small f_E to learn about how agents learn from f_E is good, but it isn't the whole story. And one may be that shocks are too small, partly responsible for why these are no large f_E in the data.

Work after — 13 Aug 2020

I have 2 open businesses:

1) The ID question: - larger f_E needed in sample → 2 shocks

2) The weighting matrix: - take Sqrt(·) of els
- figure out rescaling

Point 2): The weighting matrix.

Taking sqrt of the elements of W does change the est results. But in a sense these results are kinda better than the flat default Nsimm.

For the "Ks in the edges" specification $\sqrt{W}$ doesn't matter a lot. I guess not b/c that one is better ID-cd anyhow!

What I notice though is that taking $\sqrt{w}$, I <u>don't</u> get $L(\sqrt{w}) = L(w^{rescale}) \cdot$ scaling factor (that I don't get $L(w) = L(w^{rescale}) \cdot scl_f^2$ is no surprise)

Ok but what does hold is:

$$L(\sqrt{w}) = L\left(\sqrt{w^{rescale}}\right) \cdot scl_f.$$ So it still works, as it should.

The thing is: maybe, w/ larger shocks, $Var(E(\cdot))$ will be higher so the weighting matrix won't be an issue at all. Still: I need to understand why the weighting matrix rescaling changes the loss fct.

Comparing to the default NS simul picture, does $\sqrt{w}$ change the shape of the loss? → It does, a little, but not a lot. This is what I expected, given that it weights diff. moments and therefore also obtains diff. estimates.

Now scale W and see if the loss changes.

It does — but again only for α_1 & α_5 ...

I've also checked what Ryan said about the codes being screwed up: that doesn't seem to be happening either.

these are 2 options: either I'm somehow rescaling in a way that does change relationships, or there is something else somewhere.

I think it's Option 2 b/c $\frac{\max(W)}{\min(W)} = \frac{\max(W^{rescale})}{\min(W^{rescale})}$

also we should have

$$W^{rescale} = \frac{1}{scl} W$$ and that holds too!

↳ I'm really starting to think that the change in the shape of the loss plot has sthg weird to do w/ the indifference of $\alpha_2, \alpha_3, \alpha_4$.

The rescaling makes the loss smaller → maybe the non-rescaled one would have the same shape if I zoomed in more?

Erm... not quite. Can it be that it's a numerical thingy that it changes shape?

Honestly, I don't think so.

Try to set $W = I$. Now rescale loss and see if it changes.
→ it does! And the same way too!

I have an idea. Maybe the loss fct is changing shape b/c the convexity moment is in place.

Shut off Wdiffs2 → Wdiffs2 = 0.

Set W = I and compare loss w/ & w/o rescaling!

⇒ They have the same shape!

Redo w/ $W = \Sigma^{-1}$ & Wdiffs2 = 0 ⇒ same thing!

YEAH! That was it! The convexity moment was effectively relatively weighted more when I scaled down the weights on the others!

I also don't get why evaluating the loss fct now (today) takes longer than before. Producing these loss plots takes 35s. instead of 15 sec.

→ Whoa, I can't believe that it was that simple! So I can move on to what matters: the forecast errors and the variance of shocks. Let's think: how best to do this:

A first pass thing is to see whether simulated fc ↑ as ℬ↑. (command_check_simulation_approx.m)

Set $N=1$ (rng(1)). Default $b_z = 1$ $z = r, u, i$

Default: $fe \in (-2,2)$. (-2,2) $\alpha = 2.5 \cdot \alpha^{tme}$.

(-5,5) $\alpha = 4 \cdot \alpha^{tme}$ $\alpha = \alpha^{tme}$

Raise $b_u = 2 \Rightarrow$ explodes.

Raise $b_u = 1.2 \Rightarrow fe \in (-4,4)$. explodes (-2,2)

Raise $b_u = 1.5 \Rightarrow fe \in (-5,5)$. -"- (-4,4)

$b_u = 4 \Rightarrow (10,10)$ $b_u = 1.8 \Rightarrow$ explodes. -"- (-5,5)

Raise $b_i = 1.2 \Rightarrow fe \in (-2,2)$ (-2,2) (-5,5) ($b_i = 2$ too)

$b_i = 4 \Rightarrow$ same. (-2,2) explodes

b_i should have an effect though.

$b_i = 8 \Rightarrow fe \in (-4,4)$ (-2,2) $\rightarrow$ It does, just a much smaller one.

Raise $b_r = 8 \Rightarrow fe \in (-5,5)$ (-4,4) explodes, $b_r = 2$

↳ Also a much smaller effect. I'm a little surprised that b_r apparently has a higher effect than b_i b/c ... oh no, idiot... of course b_r has a higher effect b/c the int. rate works from the IS-curve!

$$\pi = \kappa X + u$$

$$\leftarrow X = -b_i + r^n$$

$$\leftarrow i = \gamma_\pi \pi_t + \bar{i}$$

So the strength of the effect is of course $b_u > b_r > b_i$.

Ok, so that was like initial reconnoitering. I guess the next step is going to be to see how the estimation reacts to 1) true data being generated via higher λ. Maybe the easiest is to do $\lambda_u = 4$ as a first pass. w/ $\alpha^{true} = \alpha^{true}$ (0.05, 0.025, 0, -1/1, -1/4)

↳ this is the Aug 13 data!

Shit. VAR complaints, a lot of them.

Ridge? No complaints.

↳ But then maybe the estimation needs to use ridge too.

↳ Yep, it definitely complains.

Ok now it has some explosions (Je was nan) and it still has VAR issues despite the ridge. Let's try to lower (?) the ridge parameter λ.

(It was 0.001, now it's 0.0001)

↳ immediate VAR error.

So raise it to 0.01

↳ seemed to work well, but then an "input to rank must not contain NaN or Inf" error. Hmm.

↳ Let's try $\lambda_u = 2$. Think I need to rethink the ridge?

Yeah, immediate error. → Can create data w/o ridge? No.

Ok, so the situ is the following: for higher b, data creation and estimation needs ridge. So To DO: look into ridge and confirm that it's correct. Create some replicatable simple example.

Confirming ridge

17 Aug. 2020

The issue of centering & scaling: Matlab's default for ridge is to center & scale: X is standardized (mean 0, var 1)
and y is centered (demeaned I guess?)
There is mention of this on some websites where I check ridge. Why is this important?

Statweb.Stanford.edu

Ridge is related to PCA: sort of weighted X's are jth principal components of X.

→ Ridge projects y on the PC's w/ large variance.

Statlect.com provides a hint as to why:

OLS is scale-invariant: β^{OLS} of $R \cdot X = R^{-1} \cdot \beta^{OLS}$ of X.

β^{ridge} is NOT scale-invariant! Therefore, whether you express variables in m or cm, e.g., matters!

↳ We therefore standardize all variables in ridge to

avoid undesirable scaling effects.

But I think you only need to stdize X, not Y?

Matlab helpfile is confusing me now:

$$B0 = \left[\text{mean}(Y) - \frac{\text{mean}(X) \cdot B1}{\text{std}(X,0,1)}, \frac{B1}{\text{std}(X,0,1)} \right]$$

Maybe Matlab is actually standardizing in each case, it just computes B0 as above?

Stata notes on Ridge (ncss-wpengine.netdna-ssl.com)

- standardize all variables: X and Y
- when final results are presented, they are adjusted back into their original scale. (Matlab's "0").

→ When "scaled=0" "ridge restores the coefficients to the scale of the original data".

↳ It's out: · Matlab standardizes in any case
· scale = 1 or 0 determines whether Matlab scales them back to their original scale.

→ So let's stdize Y & X!

Many observations:

1) still not the same as Matlab.

2) Matlab's ridge.m doesn't seem to be scaling Y.

↳ If I use the original, non-stdized Y, I get the same as Matlab's Ridge, scale=1.

✓ ok, but should one std-ize Y or not?

Let's see:

$\hat{\beta}^{OLS} = (X'X)^{-1} X'Y$. Let $Z = XR$ ← rescales.

$\tilde{\beta}^{OLS} = ((XR)' XR)^{-1} (ZR)'Y$

$= (R'X' XR)^{-1} R'X'Y$

$= (R')^{-1} (X'X)^{-1} (R^{-1} R') X'Y$

$= (R')^{-1} (X'X)^{-1} X'Y = R^{-1} \hat{\beta}^{OLS}$.

$\hat{\beta}^{ridge} = (X'X + \lambda I)^{-1} X'Y$ Again $Z = XR$

$\tilde{\beta}^{ridge} = ((XR)' XR + \lambda I)^{-1} (XR)'Y$

$= (R'X'XR + \lambda I)^{-1} R'X'Y$

$= (R'X'XR + \lambda R'(R')^{-1} R^{-1} R)^{-1} R'X'Y$ | Factor R' and R (from left / from right)

$= [R' (X'X + \lambda (R')^{-1} R^{-1}) R]^{-1} R'X'Y$

$= R^{-1} (X'X + \lambda (R')^{-1} R^{-1})^{-1} (R')^{-1} R'X'Y$

↑ inverse of product switches order.

$$\tilde{\beta}^{ridge} = R^{-1}\left(X'X + \lambda (R')^{-1}R^{-1}\right)^{-1}(R')^{-1}R'X'Y$$

$$= R^{-1}\left(X'X + \lambda (R')^{-1}R^{-1}\right)^{-1}X'Y$$

For this to be equal to $\hat{\beta}^{ridge}$, we need $(R')^{-1}R^{-1} = I$ i.e. R needs to be orthonormal. Generally doesn't hold!

↳ only need to standardize X! (Not Y.)

Do need to standardize but also need to scale back! → NTU stats. stackexchange question & answer: (Philipp Burkhardt) 18 Aug 2020

For when only regressors std-ized.

Let $z_j := \frac{x_j - \bar{x}_j}{s_j}$ standardized regressor j

Then we have the regression line

$$E[Y] = \beta_0 + \sum_{j=1}^{k} \beta_j z_j$$

which gives fitted values

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{k} \hat{\beta}_j z_j \quad . \quad \text{Write this as}$$

$$\hat{Y} = \hat{\beta}_0 + \sum_{j=1}^{k} \hat{\beta}_j \left(\frac{x_j - \bar{x}_j}{s_j}\right) \quad | \text{Rearrange}$$

Statistics has $\bar{Y}$ in her too.

$$\hat{Y} = \left(\hat{\beta}_0 - \sum_{j=1}^{k} \hat{\beta}_j \frac{\bar{x}_j}{s_j}\right) + \sum_{j=1}^{k} \hat{\beta}_j \frac{x_j}{s_j}$$

Scaled-back intercept

β scaled back (= what Statistics has)

If both regressor and regressand were scaled:

$$\hat{y}_{scaled} = \frac{\hat{y}_{unscaled} - \bar{y}}{s_y} = \hat{\beta}_0 + \sum_{j=1}^{k} \hat{\beta}_j \left(\frac{x_j - \bar{x}_j}{s_j} \right) \quad \Big| \cdot s_y + \bar{y}$$

$$\hat{y}_{unscaled} = \bar{y} + s_y \hat{\beta}_0 + s_y \sum_{j=1}^{k} \hat{\beta}_j \left(\frac{x_j - \bar{x}_j}{s_j} \right)$$

$$= \left(\bar{y} + s_y \underbrace{\hat{\beta}_0}_{=0} - s_y \sum_{j=1}^{k} \frac{\hat{\beta}_j}{s_j} \bar{x}_j \right) + s_y \sum_{j=1}^{k} \frac{\hat{\beta}_j}{s_j} x_j$$

This doesn't correspond to Matlab either: extra terms absent in Matlab.

I don't think Matlab is scaling y. I suspect mean(y) is added back to take care of $E(y)$ somehow?

Actually, Matlab's help for ridge.m, "Coefficient Scaling" explains it. Matlab standardizes X and centers y (uses $\hat{Y} := Y - \text{mean}(y)$), which is why mean($y$) shows up in the intercept.

Ok: at least whether you use demeaned Y or not doesn't matter for $\beta_1, \ldots, \beta_k$, only for β_0, the intercept.

Ha! If you don't use demeaned Y, & scale back w/o $\bar{y}$, you get the same thing as if you used demeaned $\bar{y}$ and scaled back w/ $\bar{y}$.

Ah - I think ridge is scale-invariant in Y:

$$\hat{\beta}^{ridge} = (X'X + \lambda I)^{-1} X'Y$$

Now let $z = YR$

$$\hat{\beta}^{ridge}(z) = (X'X + \lambda I)^{-1} X'YR$$

$$= \hat{\beta}^{ridge} \cdot R$$

So just to be like Matlab, demean Y and standardize X.

Wait - why is $\hat{\beta}^{ridge}(Y) = \hat{\beta}^{ridge}(\underbrace{Y - mean(Y)}_{=: \hat{Y}})$?

$\hat{\beta}_0^{rescaled}$ (adding back $\bar{Y}$) $= \hat{\beta}_0^{rescaled}$ (not) $+ \bar{Y}$.

0 : intercept

which makes sense. But there shouldn't be this entire business about scaling Y if it doesn't matter for $\hat{\beta}^{ridge}$ ...

Ah ... I think I know $X'Y$ is kinda $Cov(X,Y)$

$$Cov(X,Y) = Cov(X, Y - \bar{Y})$$

↑ b/c $\bar{Y}$ is a constant ($Var(\bar{Y}) = 0$)

$$Cov(X,Y) = E[(X - E(X))(Y - E(Y))]$$

$$Cov(X, Y-\bar{Y}) = E[(X - E(X))(Y - \bar{Y} - E(Y - \bar{Y}))]$$

$$= E[(X - E(X))(Y - E(Y) - \bar{Y} + \bar{Y})] = Cov(X,Y).$$

→ So demeaning Y gives you the same $\hat{\beta}^{ridge}$ as not demeaning.
So it seems like in any case you adjust the intercept, adding $\bar{Y}$ to it. Ok...

Finally the last ridge-thing of interest:
Y $n \times k$ (for VAR)
↳ Matlab can't do it, but doing it by hand, and then separately for each regressor gives the same thing.

↳ So move on: generate data w/ the new ridge commands and see if that solves the VAR-instability issues.
Note: the data-generation itself of 13 August ($\beta_u = 2$) complains of VAR-instability. That had the old ridge.
Data-generation complains w/ new ridge code?
(rf_var_ridge.m)
Nope, it didn't! :) ($\beta_u = 2$, $\lambda = 0.004$)
⇒ acf_sim_univariate_data_18_Aug_2020.

Ok, so finally do LST w/ 18 Aug data

- ridge $\lambda = 0.001$
- $b_u = 2$
- $\sqrt{w}$ (! this guy needs to become the new reference fig, as in Fig 3, Materials 41)
- $\lambda = 0.0001$

Estimation: instability in VAR: 6 times

↳ It seems like it's the initial 6 evals that cause problems. Warning never occurs afterwards.

→ Fig 4, Materials 41 (Flag = 3)

- Try same thing for TolFun = 1e-9 → same, Flag = 2.
- same, TolX = 1e-9. → same, Flag = 3.

- Now: do 'manual' grid out in far end [TolFun = TolX = e-9]
  But really one should have TolX = TolFun = 1e-6 (default) and add Wmid = 1000. Yeah this one's not great.

↳ Doing that now.
  - tols back at defaults
  - $b_u = 2$ (ridge, $\lambda = 0.001$)
  - $\sqrt{w}$
  - fegrid = [-4, -3, 0, 3, 4]

Not bad but not great either. What avenue to pursue? Raise b_u more?

Prep for Peter:

- ZEN interview
- Presentations: Aug 25 12.30 - 2pm (BU macro read)
  Sep 1 3.30 - 4.30 pm (Diss Workshop)
- Content: Just Fig 15 from Materials 40

stand talk!

Peter meeting

18 Aug 2020

Main issue: works, but not so well in the neighborhood of 0 f_E.

↳ Ready to go back to true data? Yes?
But what you wanna do for sure is to constrain the parameters of f_E close to zero.

For real data: proceed in steps:

- first identify α (large f_E) and constrain α(small f_E) and see how well you can do
- gradually expand # of α to expand to α(small f_E)

- Volatility of stocks important AND data is likely to be informative about them
  ↳ could est them
  ↳ or calib them such that model-sim data matches volatility of data.

. b shocks are simply harder to calibrate!

→ 2 questions:

1. In real world data, you wanna have a good feel for how ambitious you can be in $\ell 80\ \alpha$ and whether this depends on b
2. b is simply hard to assign values to.

↳ Talk after BN seminar!

10:30 - 11:45 Tue & Thu Febio 9 - 20.15

Do an email exchange after DiffsWork to talk on Sep 2 or the next days.

Work after

Ok - so see if we can estimate b.

I wanna do a new obj function which complements

obj_GMM_LOMgain - univariate_mean.m

in that it also estimates b_j $j = r, u, i$ - shocks.m

I also wanna

✓. change names of figures so that they're not as long

→ Try it on Aug 10 data b/c that's the 4x scaled-up data.

! And TW is a default now! ! Wmid=1000 too! Nsimul too!

The code is running... Took 713 sec. And it sucks: ↪ 70 iters
$b_j = 0.1$ $\forall j$ (while truth is 1).
Why didn't it move? Error in code or? Is it time to try Ryan's thing of making it take larger steps when evaling the loss?

After 70 iterations, it looks like Matlab never tries anything else than b_0. It doesn't seem like it's a problem of getting the values into "param".

↳ I think it's something like "forward finite differences"

As usual, some background: finite differences seems to be a numerical approximation method for derivatives.

Wait you idiot: of course it's not trying 19 Aug 2020
out any values of b: b is irrelevant in the model as it now stands. I think I need to modify the obj. fct.
→ Yeah, now I can see Matlab trying different things, that's good.

Finite difference := an expression of the form $f(x+b) - f(x+a)$

difference quotient := $\dfrac{f(x+b) - f(x+a)}{b-a}$ "finite diff approximation"

Finite differences are used in approximating derivatives, or for "finite difference methods" for solving ODE's.

finite difference quotients are used in approximating derivatives.

3 types:

forward difference: $\Delta_h[f](x) := f(x+h) - f(x)$

backward difference: $\nabla_h[f](x) := f(x) - f(x-h)$

central difference: $\delta_h[f](x) := f(x+\frac{1}{2}h) - f(x-\frac{1}{2}h)$

! Authors who use the term "finite differences" to mean "finite difference approximation of derivatives" also mean "forward difference quotient" when they say "forward difference".

Relation w/ derivatives

$$f'(x) := \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}$$

For a fixed h, this is

$$\frac{f(x+h) - f(x)}{h} =: \frac{\Delta_h[f](x)}{h}$$

the forward difference quotient!

And I think the 15 July correction to the convexity moment does this: $\Delta\alpha = f(x+h) - f(x)$, $h = \Delta f_{grid}$.

Exciting stuff: in Matlab's solvers, there's the

FiniteDifferenceStepSize: I guess this is "h".
The default is sqrt(eps) for forward
(for optimset, "FinDiffRelStep") eps^(1/3) for central finite diffs.

FiniteDifferenceType: "forward" (default) or "central"
central is more accurate but takes twice as many function evaluations.
(for optimset, "FinDiffType")

I'm trying: same (i.e. MaxFunEvals = 2200 (instead of 700)) → 45 min!
but FiniteDifferenceStepsize = 100 · sqrt(eps).
Same. $\hat{b} = (0.43, 0.25, 0.56)$.
Let's go back to MaxFunEvals = 700 (12 min).
and set FinDiffStepSize = 100 000 · sqrt(eps). awful! timed out & didn't move
↳ Better but still timed out. Try 1M · sqrt(eps)

Let's make a little: why is the solver iterating out now w/o convergence? Am I starting too far from the truth? Is this complementarity between α and b which comes from the ⊕ feedback loop in the model?

What is really weird is that, compared to the case where

I fixed $b = 1$, now the estimates for $\hat{\alpha}$ are lower, while $\hat{b} < b^{true} = 1$. I would have expected $\hat{\alpha}$ to be higher, b/c the true data is more volatile (b/c $b^{true} > \hat{b}$) so to capture that, if the estimation doesn't raise b, it should raise $\hat{\alpha}$! Analogously, the moments aren't matched well at all.

Interestingly, increasing the findiff stepsize does allow the estimation to move away from the initial values more and fits moments better too. But why doesn't the solver continue on this path?

With findiff stepsize = $100k \cdot \sqrt{\varepsilon}$, $\hat{b} = (3.5, 0.9, 0.75)$ and $\hat{\alpha} < \alpha^{true}$, but closer. So, in a sense, I see the tradeoff I was talking about (instead of $\hat{\alpha} \uparrow$, $\hat{b} \uparrow$, in fact above b^{true}), but in a way, $\hat{b} \uparrow$ was accompanied by $\hat{\alpha} \uparrow$, so the complementarity is kinda also not there.

I also wonder whether there's a kinda "symmetry indifference" in the sense that in half of simulations, $fc < 0$, in the other half, $fc > 0$, so $\alpha_{1,2} \gg 0$, $\alpha_{4,5} = 0$ matches moments just as well as $\alpha_{1,2} = 0$, $\alpha_{4,5} \gg 0$. Similarly, $b_r = 0$ & $b_i \gg 0$ may be undistinguishable from $b_r \gg 0$, $b_i = 0$.

Or is the estimation saying that x is more volatile and therefore $\hat{b}_r > \hat{b}_i$ or $\hat{b}_u$? But why isn't it raising $\hat{\alpha}$?

And is 1M · default fordiff stepsize not moving at all b/c

i) jacobian is so imprecise

ii) or the loss is really flat over large surfaces?

$\frac{f(x+h) - f(x)}{h}$ is initially <0 as $\hat{b}$ is raised,

and then it becomes consistently $= \varepsilon < 0$, small.

Maybe what is going on, is that if $h :=$ fordiff stepsize is small, it's not getting to large b, so it times out, but if h large, it gets to large b, but then the steps in α are too large to make a diff.

Let's try $h = \begin{bmatrix} h_b \\ h_\alpha \end{bmatrix} = \begin{bmatrix} 100k \text{ for all the } b\text{'s} \\ 1 \text{ for all the } \alpha\text{'s} \end{bmatrix}$.

Interesting: this leads to $\hat{\alpha} < \hat{\alpha}$ ($h = 100k$ · default $\forall$ params)

$\hat{b} < \hat{b}$ ($-11-$)

Reverse to $h = \begin{bmatrix} h_b \\ h_\alpha \end{bmatrix} = \begin{bmatrix} 1 \text{ for all } b\text{'s} \\ 100k \text{ for all } \alpha\text{'s} \end{bmatrix}$

Interesting: this gets me almost as far as $h = 100k$ for all!

↳ So it's the h in α that matters, not so much the one in b.

What does that mean? → The loss is locally flatter in α than in b.

Try: $h = 100k \times$ default for all (uniformly), but initialize at truth: $\gamma_0 = 1$.

↳ Much better (see Mediocrals 41) → seems to suggest that some of the "small step issue" may be coming from the initialization. But maybe the same holds true for α?

↳ Trying $\gamma_0 = 2$. Much more explosions and $k^{-1} < 0$. But best so far! If I don't make the estimation "experience volatility", it doesn't.

↳ Maybe should do the same for α: initialize at truth or at higher values.

Also: I'm wondering if I should penalize $k^{-1} < 0$ in the estimation?

Prep for Ryan

- ~~ZEW~~ interview → not avail.
- Presentations: Aug 25 12.30 - 2pm (BU macro read)
  Sep 1 3.30 - 4.30 pm (Diss Workshop)
- Teaching: coordinate → no rules or things to abide by.

Ryan meeting — 19 August 2020

Makes sense to $\gamma_0 > \gamma^{true}$.

- Pick $\mathcal{D}_j$ such that you list moments of data.

- Fixed-point problem: make a truth that looks like the data.
  ↳ fix some params. ⇒ Est. on real data. Take $\hat{\alpha}^{real}$ and put it into the fake data est. ⇒ Iterate until convergence.

Loss is
- Not continuous ← capital crime
- Kinky ← misdemeanor

↳ 2 crimes you can make

↳ cutting simulation that exploded introduces a jump in the obj. fct.

↳ Any gain that's < 0, set it to zero and go on w/ the simulation

↳ for exploding fc, you could set fc to a max

↳ clean sol.

Could explain diff to NEstim

Worst-case scenario: vs projection facility which introduces kinks!
Can take 3×3 sigmas, est α for 9 cases, take $\arg(\hat{\alpha})$ or the one w/ the best loss.

Try to calibrate $\mathcal{D}$ so as to match moments of data. See if est then is better ID-ed. Can send Lucas an email to get feedback.

Take-aways:

① 1st priority is to calibrate b's to match data moments (for a given α) and see if this higher volatility world helps ID-ing the α's.

② Need to address how you deal w/ exploding f_e and $b^{-1}<0$ in individual simulations. Throwing those out introduces jumps in the objective fct which is the biggest crime you can commit.

↳ doing this could explain the difference to the "N estimations" strategy b/c that one doesn't have these jumps, it has a cross-section of $N=200$ anyway!

③ For the "ceteris paribus problem" in b and α, can start w/ a laboratory w/ fixed α, b. Estimate those for fake data. Does it work? Estimate on real data. Use $\hat{\alpha}$ & $\hat{b}$ as α^{true} & b^{true} in the next round of fake data. Iterate.