Work after — 26 May 2020

Trying to map VFI results to those of parametric expectations.

The VFI sol is $V(X)$, where $X = (k_{-1}, \bar{\pi}, r^n, u)$

So I'm trying to input X^{PE} at the sol of PE.

Ok, that gives me the value function at the sol.

But I want the policy function, $\{i_t\}_{t=1}^{T}$.

$$V = L_t(x_{t+1}, \pi_t) + \beta V$$

So $(1-\beta)V = L_t(x_{t+1}, \pi_t)$

↳ period loss given optimal policy

Since I know the form of the function L_t ($\pi_t^2 + \lambda x_t^2$, and I know how π_t and x_t depend on i_t), I can solve for i_t from $(1-\beta)V = L(i)$

$$L = \pi^2 + \lambda x^2$$

$$= (\kappa x + s_3 f_a + s_4 s_t)^2 + \lambda_x x^2$$

$$= \left(\kappa\left(-\delta i + s_1 f_b + s_2 s\right) + s_3 f_a + s_4 s\right) + \lambda_x x^2$$

$$L = \left(R\left(-2i + s_1 f_b + s_2 s\right) + s_3 f_n + s_4 s\right)^2$$
$$+ \lambda_x \left(-2i + s_1 f_b + s_2 s\right)^2$$

→ I just need to solve this for i

→ The matlab sol ($\min_i L$ using fmincon) works very better than the analytical sol using Mathematica (matrixlab31.nb) Beware!

Note that you also need to initialize the matlab solver at i^{PE} so it finds that one.

Peter meeting — 26 May 2020

- s_{t-1}: the right thing to do: 6-dim state-vector b/c $s_t - s_{t-1}$ are separate states
  → which would allow for serial corr in the shocks.

- x^{PE} in $V(x^{PE})$ on RHS in (3)
  → $V(x_t^{PE}) - \beta V(x_{t+1}^{PE}) = L$

- Check whether a finer grid makes a diff w/o estimation, a small grid is fine. The moment you need quantitative exercises, then you need a larger grid.

$$V(X_t^{PE}) = L(i) + \beta E V(X_{t+1}^{PE})$$

If they are iid, then Peter is right to say that $E X_{t+1}^{PE} = X_t^{PE}$. To encompass the case of AR(1), we can write

$$E X_{t+1}^{PE} = h_X X_t^{PE}$$

but that's not cool for the endog. states in X.

How to compute $E_t(k_{t+1}^{-1})$ and $E_t(\bar{\pi}_{t+1})$?

$$E_t(k_{t+1}^{-1}) = E_t[\rho_k k_t^{-1} + \gamma f e_{t+1|t}^2]$$

$$= \rho_k k_t^{-1} + \gamma E_t[(\pi_{t+1} - (\bar{\pi}_t + b_1 s_t))^2]$$

$$E_t(\bar{\pi}_{t+1}) = E_t[\bar{\pi}_t + k_{t+1}^{-1} f e_{t+1|t}]$$

$$= \bar{\pi}_t + E_t[k_{t+1}^{-1} f e_{t+1|t}]$$

→ It all seems to hinge on $E_t[f e_{t+1|t}]$

$$= E_t[\pi_{t+1} - \bar{\pi}_t - b_1 s_t]$$

$$= E_t[\pi_{t+1}] - \bar{\pi}_t - b_1 s_t$$

↑ E[AUM] (1,1)

You know what: better idea: 27 May 2020

$$L = \pi^2 + \partial x^2 \qquad =: fas$$

$$= (Kx + \underbrace{s_3 fa + s_4 s_t}_{=: fas})^2 + \partial_x x^2$$

$$= \Big(K\big(-bi + \underbrace{s_1 fb + s_2 s}_{=: fbs}\big) + s_3 fa + s_4 s\Big) + \partial_x x^2$$

$$\Rightarrow L = \big(K(-bi + fbs) + fas\big)^2 + \partial_x (-bi + fbs)^2$$

$$= K^2(-bi + fbs)^2 + fas^2 + 2K(-bi + fbs)fas$$

$$+ \partial_x (-bi + fbs)^2$$

$$= (K^2 + \partial_x)(-bi + fbs)^2 + 2K \cdot fas(-bi + fbs) + fas^2$$

$$= (K^2 + \partial_x)(b^2 i - 2b\, fbs\, i + fbs^2) - 2bK \cdot fas \cdot i + 2K\, fas\, fbs$$

$$+ fas^2$$

$$L = (K^2 + \partial_x) b^2 i^2 - (K2b(K^2 + \partial_x) fbs + 2bK \cdot fas) i$$

$$+ 2K\, fas\, fbs + fas^2 + (K^2 + \partial_x) fbs^2$$

$$(K^2 + \partial_x) b^2 i^2 - (K2b(K^2 + \partial_x) fbs + 2bK \cdot fas) i$$

$$+ 2K\, fas\, fbs + fas^2 + (K^2 + \partial_x) fbs^2 - L = 0$$

$$(k^2+\lambda x)b^2 i^2 - (2b(k^2+\lambda x)fbs + 2bk \cdot fas)i$$

$$+2k\,fas\,fbs + fas^2 + (k^2+\lambda x)fbs^2 - L = 0$$

$$x_{1,2} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$i_{1,2}$

$$= \frac{2b(k^2+\lambda x)fbs + 2bk \cdot fas \pm \Delta}{2(k^2+\lambda x)b^2}$$

$$\Delta = (2b(k^2+\lambda x)fbs + 2bk\,fas)^2 - 4(k^2+\lambda x)b^2(c)$$

$$= (2b)^2\left((k^2+\lambda x)fbs + k\,fas\right)^2 - 4b^2(k^2+\lambda x)c$$

$$= 4b^2\left[\left((k^2+\lambda x)fbs + k\,fas\right)^2 - (k^2+\lambda x)c\right]$$

$$i_{1,2} = \frac{2b\left[(k^2+\lambda x)fbs + k\,fas\right] \pm 2b\sqrt{\Delta}}{2(k^2+\lambda x)b^2}$$

$$= \frac{b(k^2+\lambda x)fbs + k\,fas\,b \pm b\sqrt{\Delta}}{(k^2+\lambda x)b^2}$$ ✓

$$\Delta = \left((k^2+\lambda x)fbs + k \cdot fas\right)^2 - (k^2+\lambda x)\left[2k\,fas\,fbs + fas^2 + (k^2+\lambda x)fbs^2 - L\right]$$

~~$(k^2+\lambda x)^2 fbs^2 + k^2 fas^2 + 2k(k^2+\lambda x)fas\,fbs$~~

~~$-(k^2+\lambda x)2k\,fas\,fbs$~~ $-(k^2+\lambda x)fas^2$ ~~$-(k^2+\lambda x)^2 fbs^2$~~ $+(k^2+\lambda x)L$

$$\Delta = b^2(k^2+\lambda_x)L - \lambda_x f_{as}^2 b^2$$ ✓

So actually

$$i_{1,2} = \frac{(k^2+\lambda_x) f_{bs} + k f_{as} \pm \sqrt{(k^2+\lambda_x)L - \lambda_x f_{as}^2}}{(k^2+\lambda_x) b}$$

Ok so what have I learned:

1. Since I know there are 2 sols, doing it analytically makes more sense than w/ fsolve.

2. Depending on the values of L, f_a, f_b, s, some i may be complex, in which case all i show up as complex, some w/ 0 imaginary part. Then one needs a good way to discriminate between the two sols, for every t!

Brian Doumbeck macro lunch

Shadow price learning & Expectationally Driven BCs

main diff btwn econ & nat. science: expectations

→ nice!

Evans & McGough: shadow price learning
"SP learning"

SP learning: agents learn about shadow prices
→ using an RLS scheme (so it's a special case of adaptive learning)

- ↳ ask for slides!

  Ulrich: ppl have been saying $\pi > 2\%$ for a long time! → hard to reconcile w/ RE.

- if news shocks can be incorporated into the info set of agents, then maybe one can drive agents into responding to promises of the gov.

Is "news shocks" a proxy for adaptive learning?
Eusepi & Preston AER says that adaptive learning looks a lot like news-shock-model.

- Nick Bloom "TV volatility is exog"

. news & noise are equiv under RE
→ not obvious if they are outside RE
Brian thinks they should be equiv outside RE b/c they are exog processes.

News shock in RE: can only generate comovement if $\beta < 0.25$ instead of 1.

- SP-learning emphasizes the distinction between market clearing conditions vs. model eqs.
→ so it seems like an improvement over EE learning

SP - learners are less sophisticated than

EE - adaptive - learners.

- Is it really an RPE?

What's a state shadow price?

value to me today of an additional unit

of a state

Ryan: → λ multipliers on intertemporal constraints

(multipliers on static constraints don't

matter b/c you can sub 'em out)

He lets them regress on MSV sol.

He simulated T=500.

- job market materials?

willing to read?

- These models are usually not treated in their non-linear form!

Ryan meeting 27 May 2020

Can approx the policy fct i the same way b/c you have generated values for i at every point in the state space.

Trick to save time: convex combo of policy iter
VFI
faster than VFI but more stable than PI.
Update policy fct every 5th iter.

Keep the policy the same for some iterations

↳ saves you the max step.

The distinguishing elements of global methods
- gridpoints
- how you're approximating
- how you find zeros

Value function is also a projection method, but in any case, policy iteration is a projection bc it works directly w/ FOCs.

Brian meeting 27 May 2020

will send job market materials.

EH 2001 : misspecified models & RPE

Chap. 13

Evans & McGough Shadow Price Learning Paper

↳ sends!

End of it is internship! NK!

fwd guidance as a news shock

Offshoot: currency crisis models untethered

targeting underpinning it

everything is fine, boom!

→ exog shift in E(·)

Schmitt-Grohe & Uribe OpenEcon Textbooks

pdf online!

Let's talk periodically! Figure out over email!

Ok so mixed news: 29 May 2020

1) VFI = PEA...

2) ... only if you approx the policy function using a sufficiently large $\bar{\pi}$-grid: $\bar{\pi} \in [-0.2, 0.2]$ is clearly too small! $\bar{\pi} \in [-4, 4]$ is the smallest that seems approximately ok. The problem is that VFI w/ $\bar{\pi} \in [-10, 10]$ started diverging at one point. But I suppose I should use the same grid for VFI as for i-approx...?

Or not?

Saying that they should have the same grid is to say that we're approximating the value function on the same grid as the policy fct.

I don't know. The good news is that it's not the acceleration that screws up the results. It's the $\bar{\pi}$-grid. My concern is that the grids should be the same b/c we use i-values generated

from a grid X_1, so $i_t = \hat{V}(X_1)$
and then the coefficients that approximate
$\hat{i}(\alpha, i_t(X_1), X_2)$ shouldn't be the function of
a different grid.

Let's see if the $\bar{\pi} = [-4, 4]$ grid thus works for
VFI.

Let's try to figure out why my value function
sometimes diverges (it has a kind of spectacular
reversal). In theory (and it looks like, in practice too)
VFI is guaranteed to converge, albeit slowly.
So I'm wondering if there's something about the
learning that can screw things up: for a large
value of $|\bar{\pi}|$, some unpermitted policy is chosen
that causes say a $k_t^{-1} < 0$ or stop. Maybe
it has to do w/ a potential incorrect (or lucky)
specification of the problem constraints? But if that's
so then only doing the i-update every j^{th} iter

may actually be helpful b/c you don't do the max step so many times.

But $k_t^{-1} = \rho_k k_{t-1}^{-1} + \gamma z_t^2$

definitely keeps $k_t > 0$. What it can do though is that suddenly z_t can become huge: in fact: the bigger & more coarse the grid, the more likely that is to happen b/c it changes a lot from one iteration to the next. On the other hand, since we're on a grid, the result of every iteration is $V(X)$ where X is all combination of all states. Divergence can only happen if somewhere along X, $V^{l+1}(X)$ is very different from $V^l(X)$ Can it be that k_t reaches 0? No b/c it's safely constrained by the grid. The only thing that can happen is that $V(X_{t+1})$ is such that can i is chosen which gives a very diff L_t and thus TV.

Ha! What I'm finding from this $\bar{\pi} = [-4, 4]$ VFI is that now I need to use a much bigger grid for $\hat{i}$ (e.g. $\bar{\pi} \in [-40, 40]$ is too small, $\bar{\pi} \in [-100, 100]$ is fine)

for it to have the same shape as PEA.

Let's make the $\bar{\pi}$ grid start at 0.0001 instead of 0.

↳ doesn't change it

Obs:

- The bigger $\bar{\pi}$ grid, the bigger are the changes in V initially. Why? B/c more "data variation?"
- For $\bar{\pi} \in [-4, 4]$, the i^* are huge: $\in [-300, 300]$. I'm wondering whether this is what makes it desirable to scale up the $\bar{\pi}$ grid afterwards for $\hat{i}$. → seems to suggest that VFI suffers from a scaling issue somewhere, especially as the problem is right, just the scale off.

Ok so I've set up command_vfi & command_pea to investigate the two methods; compare_value_pea_results compares the two. Now at this point I'm really wondering if the PEA is wrong.

→ Continue there!

- Initialize PEA at different rand — 30 May 2020
  → gives the same
- Try 1e-10 as eps instead of 1e-6 in PEA
  → gives the same, took 20 min.
- Now input a different sequence of X and see if the scaling issues remain the same
  → it's diff of course but in the grand scheme of things it's similar in that
    i) it has big up/down oscillations initially
    ii) later in the sample it moves around 0 in the ±5% band.

⇒ the scaling issue in VFI seems to be there, no actually,

no; the $\bar{\pi}$-grid used for VFI was $[-4, 4]$, and loading it for the $\hat{\pi}$-approx, I get the same thing.

value_outputs_server32-accelerated

pea-outputs_30-May-2020-10_12_28

Now try Bob's w/ value_outputs-w/ a $[-4,4]$ grid!

→ it works, holy smokes! I must have used non-compatible grids. Yeah!

To do for First Draft (=Draft4) 31 May 2020

✓ • Rewrite the analysis part fundamentally

✓ • Once you have that, rewrite the intro in the spirit of Adam

✓ • Once you have that, rewrite the abstract.

- Analysis
  - ✓ - Target criterion
  - ✓ - Implementing: optimal int. rate sequence vs. a std Taylor rule.
    - ✓ • policy fct as a function of X:
      - function of $\bar{\pi}$, of b_t^{-1}
    - X • reaction fct wrt observables, π & X?
      - → relation to TR
      - → numbers to the opt. policy response!
    - ✓ • Corr btw b & π and response of $\hat{i}$ to unanticip.

0.0 0 8.8981e-04 = 0.00088981

- ✓ • App PEA & VFI
- ✓ • Conclusion
- ✓ Intro: Related lit ← 2 June start here!
  - ✓ • Go over references
- ✓ • Go over App. ←
- ✓ • visual identity for figures ↙ working on it
  - ✓ • Bigger grid server

For after:

- Basu's liq. premium correction
- TR w/ smooth crit → maybe once 1st is done.
- TV TR
- more relation of policy function to TR
- estimation

A first read on tablet. Notes: 2 June 2020

Nope.

Ryan meeting 5 June 2020

- interpretation of Lorn-k: concave/convex, slope need derivative, so you only need to plot it.
- point of TR: a simple, verifiable description of non-pol ⇒ a TV-TR has all the complications of a TC, so no need to do.

⇒ I'm at an inflection point in my work. Ryan's prior was: write up things, and then estimate, but

now it seems like the prob (estimation changes the results & the pitch of the paper) is fairly high. Therefore:

→ spend up to a month estimating

If in 2 weeks I see that I won't get any quick results from estimating, then I can still revert to sending this draft.

NB. "I need 2 weeks of lead time." in the contract means "expect to have 2 weeks until he gives you feedback", not "give him a heads-up 2 weeks before submitting."

Work after Today is June 5.

+ 2 weeks is June 19.

+ 4 weeks is July 3.

Work after

Ok so the draft is done for now. The only thing it will need to do is incorporate Ryan's idea of why a time-invariant TR exercise makes sense. Don't apologize: instead, say that the whole point of a TR is a simple, easily computable and verifiable implementation of policy.

As for the estimation, I return to the existing code command_GMM_anchoring_function.m, but replace the objective function.

- Need to redo bootstrap as a VAR
- Need to redo objective function

  ↳ incorporating a shape-preserving spline

→ Actually, I decided to leave that old file be. Instead, I moved stuff to materials33.m

1) First step is to make the 7 June 2020
bootstrap and the weighting matrix work

2) Then the GMM will have EXACTLY the same structure as before, except LOM_k will be replaced by $B \cdot s(X)$ where B are the coeffs and $s(X)$ is the basis (hopefully a shape-preserving spline or simply a piecewise-linear approx).

3) ⇒ shape-preserving spline.

1) I'm scrapping bootstrap_resample.m b/c it's useless: it's just resampling the errors w/o building a new (VAR) sample. Instead, I'm using the old code data_boot.m from the IT-project.
So I have dataset_boot, which is the nboot × T bootstrap-resampled VAR dataset.
→ Now what I need is a procedure to estimate the ACF of a VAR. Hamilton, p. 280 MaC seems

to provide such an approach.

The j^{th} autocovariance Γ_j of the original process y_t is the first n colums and n rows of

$$\Sigma_j = F\Sigma_{j-1} \quad j=1,2,\ldots \qquad [10.2.20]$$
$$= F^j\Sigma$$

which is

$$\Gamma_j = \Phi_1\Gamma_{j-1} + \Phi\Gamma_{j-2} + \ldots + \Phi_p\Gamma_{j-p}$$

for $j = p, p+1, p+2, \ldots$ [10.2.22]

The VAR-notation Hamilton uses is

$$y_t = c + \Phi_1 y_{t-1} + \Phi_2 y_{t-2} + \ldots + \Phi_p y_{t-p} + \varepsilon_t$$

VAR(p) [10.1.4]

Rewrite this in terms of deviations from the mean as

$$\mu = (I_n - \Phi_1 - \Phi_2 - \ldots - \Phi_p)^{-1} c$$

$$(y_t - \mu) = \Phi_1(y_{t-1} - \mu) + \Phi_2(y_{t-2} - \mu) + \ldots + \Phi_p(y_{t-p} - \mu) + \varepsilon_t$$

Use that to rewrite as — Q = VC matrix of v_t

$$\xi_t = F\xi_{t-1} + v_t \qquad \text{VAR(1)} \quad [10.1.11]$$

where $\xi_t = \underset{[np\times 1]}{\begin{bmatrix} y_t - \mu \\ y_{t-1} - \mu \\ \text{etc} \end{bmatrix}}$ (demeaned y_t) [10.1.9]

The j^{th} Autocovariance Matrix

$$\Gamma_j := E[(y_t - \mu)(y_{t-j} - \mu)'] \qquad [10.2.1]$$
$$= E[\xi_t \xi_{t-j}']$$

Then $\Sigma := E[\xi_t \xi_t']$.

Note that

$$\Sigma = \begin{bmatrix} \Gamma_0 & \Gamma_1 & \cdots & \Gamma_{p-1} \\ \Gamma_1' & \Gamma_0 & & \Gamma_{p-2} \\ \vdots & & \ddots & \vdots \\ \Gamma_{p-1}' & \Gamma_{p-2}' & \cdots & \Gamma_0 \end{bmatrix} \qquad [10.2.12]$$

Ah I see. I need to rewrite my play VAR(p) as a VAR(1).

Recall from [10.1.11] that

$\xi_t = F\xi_{t-1} + v_t$. Thus $\Sigma = E[\xi_t \xi_t']$

$= E[(F\xi_{t-1} + v_t)(F\xi_{t-1} + v_t)']$

$= E[(F\xi_{t-1} + v_t)(\xi_{t-1}'F' + v_t')]$

$= E[F\xi_{t-1}\xi_{t-1}'F' + v_t v_t' + \text{cross terms}]$

$= FE[\xi_{t-1}\xi_{t-1}']F' + E[v_t v_t']$

$$\Sigma = F \Sigma F' + Q \qquad [10.2.13]$$

This can be solved for Σ using vec as

$$vec(\Sigma) = \underset{r^2 \times r^2}{\left[I_{r^2} - (F \otimes F) \right]^{-1}} vec(Q)$$

Then, having Σ, we find the ACF's (Γ_j) by computing first Σ_j, the ACF of the VAR(1).

$$\Sigma_j = F^j \Sigma . \qquad [10.2.21]$$

Finally, we obtain the ACF of the VAR(p), Γ_j, as the first n rows and cols of Σ_j.

What is n, p, r?

p = order of VAR.

F is $np \times np$ so n must be the # of variables

Maybe $r^2 := (np)^2$? Yep!

→ solved bootstrap and weighting matrix by using the estimate of the ACF from Hamilton!

A day in Provi – in honor of which 8 June 2020
I'm using Brown ;-)
⇒ focus on shape-preserving splines
In Matlab: pchip seems to be the way to go
Since, believe it or not, it actually is a one-dimensional object.
I just wanna find out in what relationship pchip stands w/ Schumaker.

Well that was it ;) 9 June 2020
See Splines and pchips from Clere's Corner
(saved as pdf in numerical_methods in next.
The point is that Hermite cubics and splines are both piecewise cubic polynomial interpolants.
The difference is that Pchips (Hermites) is the 1st & 2nd derivative conditions & spline (at least in Matlab) imposes 2nd derivative conditions

at the knots and at the endpoints. The Hermite interpolating polynomial only uses 1st derivative conditions. (For this reason, the 2nd derivative may have kinks and jumps, while for a spline it's smooth & continuous. Even the 1st derivative of the Hermite exhibits kinks, while that of the spline is smooth.)

In particular, the Hermite requires that at interior points, the slope is a <u>weighted harmonic mean</u> of the slopes of the piecewise linear interpolant, while at the endpoints, one-sided slope conditions are imposed. That's it, and for this reason, it's less smooth but instead <u>shape-preserving</u> than the spline.

<u>Harmonic mean</u>

$$HM(x_1, \ldots, x_n) = \left(\frac{x_1^{-1} + \ldots + x_n^{-1}}{n} \right)^{-1}$$

<u>Arithmetic mean</u> = $AM(x_1, \ldots, x_n) = \frac{x_1 + \ldots x_n}{n}$

It looks like for ND data, the only 10 June 2020
hope is spapi, where

1) you can specify the order of the interpolating polynomial

2) it may be that the cubic is shape-preserving

But spapi is in B-form which is, technically,
a pp (piecewise polynomial), w/ some differences.
The coefficients are a d x n w/ breaks t(i) (n+k)+1

Formally, the spline f is defined as

$$f = \sum_{i=1}^{n} B_{i,k}\, a(:,i)$$

↓

i^{th} B-spline of order k for the given knot sequence $t(i), \ldots, t(i+k)$

→ Need to use spmak to construct the B-form / pp.

Gaetano Gaballo: Learning by Shopping

Micro Survey of π-E(·):

- huge dispersion

  but disp(π-E(·)) < disp(actual π)

  i.e. disp(price level at individual level)

  HHs

→ . very heterogen π-E(·)

  . very influenced by their limited shopping experience

Ryan meeting — 10 June 2020

example for ndim_simplex.m! → He sent it.

↳ test_simplex2d.m

Use a grid for k_t, fe_{t+1-1}, then use A.6 to get initial values for b_t, fe_{t+1-1}.

Actually, this is important b/c if you simulate,

You may not cover the entire domain of the function and therefore you may be underidentified (b/c a finite-element approach needs data from all elements, otherwise it's singular, vs. global interpol /spectral methods like Cheby.)

The other thing is the inputs to ndim_simplex: x and xx

X = is the whole grid: it stays constant for every evaluation of the fct.

XX = is the current input point; it changes for every evaluation.

Triangulation is about the efficient choice of triangles, once you have a grid. But for Ryan, the choice of a (uniform) grid is tied to the

triangulation b/c he just picks a triangulation that he knows exists for a uniform grid. But it's not efficient, and it seems efficient methods aren't implemented for ND (>3).

Also note that in the binning step, ndsim_simplex.m stretches the triangles at the edges so that if you have a data point outside the grid, you still get attributed to that edge-triangle, so you still obtain a value (this is extrapolation)

Work after

I wonder if I should do log selection for
1) all bootstrap samples
2) model-implied datasets individually or
should I use the log selected for the initial data?

→ I always (or very often) have singularity issues w/ my regressor and I think it might come from having too many legs

Cond here!

↳ Need to solve singularity issues w/ regressors.

→ Once that is done, maybe the coeffs will update properly. → See Peter's email, saved into notes as Peter_email_11_June_2020.pdf.

stochastic singularity.

I'm still dropping that issue 11 June 2020

for now and turning to the ndim_simplex issue.

I think it's happening b/c the gain goes sufficiently close to -Inf.

⇒ How can I tell my approx not to result in negative gains??

→ If I allow the α-coefficients to move too far from α_0, I get this issue.

Gaetano Gaballo meeting 11 June 2020

- don't fight rational expectation
  ↳ Be specific about language
  ⇒ smoother ways of saying you are a member of the other crowd
  → produce large deviations in $\bar{\pi}$ is very hard but we need very small departures in the model (too much!)

You use a lot of the language of the adaptive learning
e.g. the understanding of RE.
↳ Guesserie → you may not end up in REE even if you know the model b/c of coordination failure

Marcet & Adam & Evans & Preston are the ones successful in selling the argument: how large is the departure that can explain → we

want the smallest departure that can still explain the stuff.

" W/ a small departure in modelling I can achieve deviations from a fixed target and nice intuitions about behavior "

There should be a mapping from k_t to a Kalman gain.

Graph to explain the target criterion
→ relevant graph here: Utility indifference curve of planner
Maybe simplify into 2 periods

Ppl will not be surprised that w/o RE, distortion = commitment. So the contrib. is how the learning changes the problem.

should sell to CB b/c it's "insensitive to a non-linear world"

⇒ emphasize not only what is opt pol, but what's the mistake I make when I follow a RE min pol.

- RE is great in LR
- but crisis has shown that things move quickly
- so how costly is it to ass plain vanilla RE

Recall that discretion = commitment

But the sol differs depending on the model

So I do 3 things:

1. Est. model
2. solve opt. policy
3. Assess the mistake I do by ass RE

Add numbers.

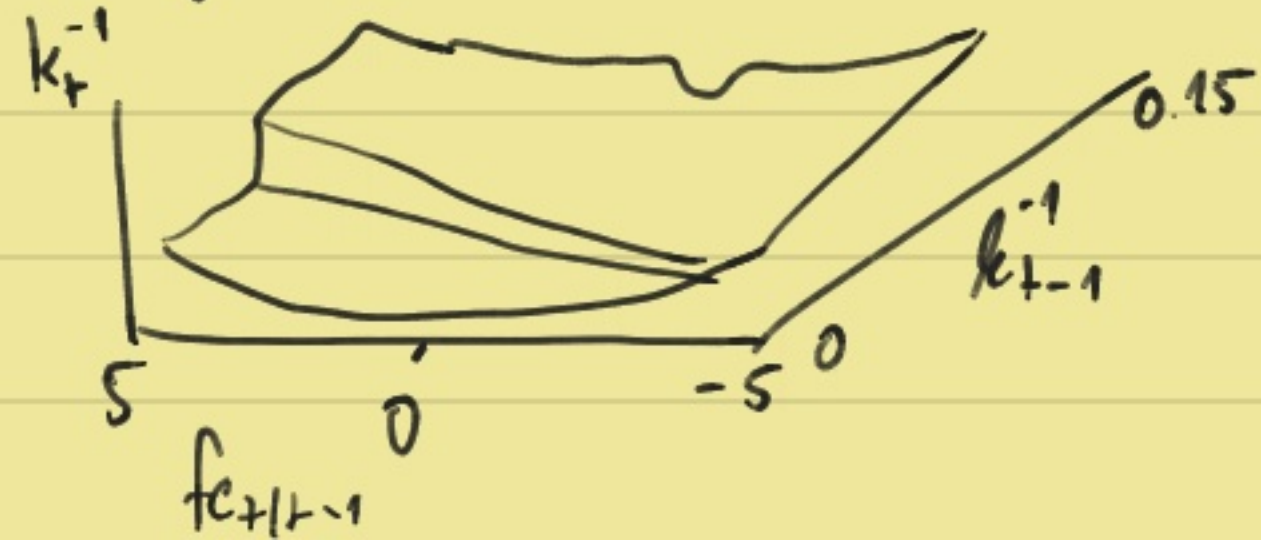
A two-period problem w/ an intertemporal price ppl can learn about, you can already show the mistake ppl make by using the RE ass.

He said it's fine to talk at the end up summer.

Work after 12 June 2020

I needed to make sure the simulation doesn't explode for certain values of α, so whenever an explosion happens, I set the GMM loss to 1e+10. (It's otherwise on an order of 3e+02.)

The estimated relationship (when $\alpha > 0$ constrained) is something like



1) For a small k_{t-1}^{-1}, a given k_{t-1}^{-1} is associated w/ a larger k_t^{-1} if the $fe_{t|t-1} < 0$ than if it is $fe_{t|t-1} > 0$.

NB $fe_{t|t-1} < 0 \rightarrow \pi_t - fcst < 0 \Rightarrow \pi_t < fcst$

$\Rightarrow$ surprise deflation / disinflation

"negative inflation surprise"

"overestimated inflation"

2) This is no longer true for high k_{t-1}^{-1}: then a given k_{t-1}^{-1} is associated w/ higher k_t^{-1} if $|fe|$ higher except for when fe is small and negative.

→ Let's rephrase in words

0) E(·) are always more likely to unanchor for larger FE.

1) When expectations are anchored, they are more likely to unanchor for negative inflation surprises.

2) When E(·) are unanchored, then small negative inflation surprises are associated w/ faster anchoring.

$\Rightarrow$ This may be driven by my sample which spans

1959 - 2020 (Jan 1 to Jan 1), so the Great Inflation is in there, under which $E(\cdot)$ was unanchored, and seeing π come in lower than expected was stabilizing. Once $E(\cdot)$ became anchored in the 90s and especially in the 2010s (after 2008), inflation shortfalls are more likely to unanchor $E(\cdot)$.

⇒ Neat!

Redo PEA w/ this?

For a distance D, we seek 13 June 2020

the stepsize s that splits it into $(n_g - 1)$ parts (n_g knots) where one node is xx.

$$s \cdot (n_g - 1) = D \qquad (1)$$

$$y \cdot s = xx \qquad (2)$$

(1) in (2): $y \frac{D}{n_g - 1} = xx \Rightarrow y = \frac{n_g - 1}{D} xx$

$$\Rightarrow s = \frac{xx}{y}$$

This is fine but it doesn't lead to $y \in \mathbb{N}$
b/c nothing guarantees that $\frac{n_g}{D} \cdot xx \in \mathbb{N}$.

It'd be better to choose n_g such that $y \in \mathbb{N}$ both for $\vec{k}$ and for f_c.

So I have 2 distances, D_1 and D_2, which I want to split into n_g points w/ stepsizes s_1 and s_2, such that $xx(1)$ and $xx(2)$ lie on the grid.
So there must exist $(y_1, y_2) \in \mathbb{N}^2$ such that
$y_1 \cdot s_1 = xx(1)$ and $y_2 \, s_2 = xx(2)$.
while the stepsizes satisfy
$(n_g - 1) \cdot s_1 = D_1$ and $(n_g - 1) \, s_2 = D_2$

$$\Rightarrow y_1 \frac{D_1}{(n_g-1)} = xx(1) \quad \text{and} \quad y_2 \frac{D_2}{(n_g-1)} = xx(2)$$

$$\Leftrightarrow \quad y_1 = \frac{n_g - 1}{D_1} xx(1) \; ; \quad y_2 = \frac{n_g - 1}{D_2} xx(2)$$

↓ How to restrict these to be integers? ↙

→ I solved it by simply approximating the derivs g_π and $g_{\bar{\pi}}$ at the closest gridpoint.

Now I have the problem that the PEA is stalling at the fsolve step. More precisely, the estimation of the Jacobian takes long (1 min) And then the rest seems long too...
I wonder why? The fsolve step shouldn't be so much more difficult only b/c of the dam estimated LOM-gain.

→ No it ain't stuck, fsolve is just now having a hard time to solve. In the 1st iter, it found no sol. Also, it's sooo slow now that I can't possibly do this on my Mac.
By contrast, the VFI (accelerated, 4x4x2x2 grid) takes 30 min.

Let's also pbs it.

Ok, one feeling why it's so slow is b/c of
fe_smooth_approx.m, b/c this involves
computing a bunch of gradients, which is costly.
Can we try to do it outside?
Yes, that speeds things up considerably (40 min
on server) The problem is that now fsolve isn't
solving, and therefore $\|\beta^{i+1} - \beta\|$ is jumping around
all over the place.

↳ Can I est. (ρ_k, γ_k) from est. LDM grid?
Do ρ_k, γ_k suggest too high values such that
fsolve doesn't solve?

I'm concerned b/c technically, 14 June 2020
I should have done PEA w/ the PLM "constant only
tt only" instead of w/ "constant only".

The reason $k^{-1} < 0$ in PEA is b/c k^{-1} goes outside

the grid I previously used in GMM. In particular, $k^{-1} < 0.001$ which was my lower bound for k1grid.

15 June 2020

Ok, but GMM can't find min for k1grid = [0, ... $\bar{g}$] it can for k1min = 0 + ε, where ε very small. Maybe more troubling is that GMM can't solve for PLM = "constant-only, π-only". Since that is the view of the model on the world, it should be that it finds a LOM gain for that view.

Discovery: my "copo" PLM sim_learning code (constant-only-pi-only) diverges in <u>any case</u> The constant-only doesn't, slope & constant is ok too!

⇒ the problem then shows up all over the sim_learn LH_clean.m - family!

⇒ back to clean_up_sim_learn.m

I did! It was only that you need to set smart so that agents know the TR! This only becomes a problem for copo b/c otherwise they just learn it ;)

What have I learnt today?

1) Need to be very careful about smart.
   → may also be important for how you initialize the PEA b/c the TR simulation should be unstable unless you use the co-PLM (constant-obj) or you have them know the TR → might provoke calling smart
2) Somehow in the PEA, $k^{-1} < 0$ even if I have an approx w/ only > 0. So it shouldn't happen. Maybe I'm misunderstanding ndim Simplex after all?

3) Estim seems to depend on grid too
→ another reason to understand ndim_simplex better. Strange: odd or even matters a lot!
→ more calls to understand how to restrict coeffs α such that $k^{-1} > 0$.

To do's for tomorrow

analyze_estim_LONgain.m

- rework calling smat
- investigate the behavior of simLH_clean_approx.m to see where k^{-1} might turn negative
  same goes for simLH_clean_given_seq_approx.m
- go back to the estimation
  understand how ndim_simplex works
  understand how to keep $k^{-1} > 0$

Ok, I think that reworking smat is 16 June 2020
the time to migrate PEA from the input seq

folder b/c we can't have 2 versions of everything lying around

⇒ command_pca_main.m is the migrated file and it works!

⇒ command_pca_approx_main.m is the other migrated one and it also works!

I've changed addpaths so that the inputs for Ryan folder is no longer added so that there should be no duplicate files issue

Now the next step is to create a smaller smat that allows me to select whether I'm 1) allowing them to know the TR
2) including a nonpol sketch

→ In fact, I might not have to change smat too much. It's the ALM codes (ALM.m, ASA10.m) and

the learning codes (sim_learLH-clean.m,
sim_learLH-clean_approx.m
and their _given_seq.m versions)

ALM.m & AGA10.m (→ the 4 sim-codes)
get 2 new inputs
knowTR = 1 or 0
mpshock = 1 or 0
⇒ which they pass on to smat.m
(it's the easiest to edit in smat.m directly)
For AGA10, since it's for the case when i (at least) is input, there's never a monpol shock.
So I just set mpshock = 0, and it doesn't need to be input in AGA10, and thus also not in the given_seq codes.
Ok, command_pca_main → sim_learLH_clean, sim_learLH_clean_given_seq, ALM & AGA10 work!

command_pea_approx_main, as well as its children sim... approx & green_seq work too!

Peter meeting — 16 June 2020

- Maybe the basis s(x) should include fc^2

- Am I flipping the sign of the equation?
  → generate IRF: ⊕ shock should be the same as a negative shock in the other

- Identification issue: it looks like if b/c you don't move much from the original
  Lack of ID means: 2 points in the param space generate ≈ eq. model behavior
  ⇒ the and. fit. may be overparameterized.
- There could be a tradeoff btwn fitting two sets of moments: one fits one better, the other

parameters fit the other set of moments better
maybe restricting α coarses
the GMM to choose one over the other.

- maybe multicoll btwn k+ & fe

- what I should do is:
do this routine (GMM) from many starting values ⇒ repeat for 100 times.
Or 250 times. Make sure you always get the same.

Nburn:

- One way is to settle on a benchmark
Actually, you could estimate nburn!

Or "numerical experimentation led me to believe that effect of initialization had died out after t=25 so I can focus on the true effects of E( )"

- Keeping $k^{-1} > 0$

  $\log(k^{-1})$ But shifting units is an issue.

  Finally fit approach is better than taking logs b/c logs forces you to think of %-deviations instead of absolute deviations.

Instead: impose $k^{-1} > 0$ as a global property

⇒ calculate k_+^{-1} for all the possible values of k_{+-1}^{-1}, fe and impose that to be > 0.

Work after ⇒ it is clear that my first step is to simulate the model for $\hat{\alpha}$ and see what behavior it implies. Calculate IRFs & so.

1. Sim model using α_0:

   - knownTR = 1 is fine, = 0 explodes → but that's fine, PEA can deal w/ that.
   - mpshock or not doesn't make a diff.
   - k^{-1} is <u>never</u> < 0 in any case

I settled for ndrop=5. If α unrestricted, the picture is fairly consistent, but uncertain. If I restrict $\alpha > 0$, then ngrid has a huge effect. Grr...

$\alpha > 0$ → implies a gain simulation that stays high, but α unrestricted an oscillatory gain, even worse.

Some thoughts:

1. fe^2 instead of fe → doesn't make sense to me b/c if the relationship is a square, then a piecewise linear approx should tell me!
2. Anchoring for overparameterized would mean that there's one (or more) param too many.
   ↳ Aha! Want: α is (ngrid · ngrid) x 1
   and I have 45 moments... So $ngrid^2 \leq 45 \Rightarrow$

→ ngrid should be 6 at most.
But strange; I seem to have similar problems
across ngrid = 4, 5, 6.

Tomorrow: try generating IRFs using $\hat{\alpha}$ and
see if you're flipping things.
Maybe generate more moments?

- I've imposed $k^{-1} > 0$ as a "global" property and it seems to work. 17 June 2020
- I don't get the overparameterized issue: if I have less coeffs than moments, shouldn't I be fine? Maybe not. Wait: no identification is not the same as overparameterization.
  "No ID" := more coeffs than moments
  Overparameterization := you fit more coefficients than what describes the true DGP. → but I'm

surprised b/c if some α's are "unnecessary",
then the estimation should set those to 0, no?

I also recall now suddenly that for PEA,
I maintained $k_+^{-1} = g(f_c)$ only, i.e. no k_{+-1}^{-1},
b/c that's the assumption I used for the TC.
→ so I'd need an estimated fct form in function
of f_c only!
↳ So. I need to try the 1D estimation anyway!

Also: could it be that I'm forcing overparameterization
to happen by setting $\alpha > 0$? Since I'm not
allowing the useless params to drop out?
→ And indeed, α^* has a 0 if unrestricted.
(at least for $n_g = 4$, not for $n_g = 6$ or $n_g = 5$)
↳ No, and in this sense overparameterization
and no 1D may be related. Take the one

Jon Reyan meeting 22 Jan 2020: overparameterize means to "overfit" a model: you estimate too many things. The problem w/ that is that the model has too many degrees of freedom which means that you'll have a good fit and yet forecast poorly.

⇒ this makes perfect sense in relation to what Peter said: you have bad ID b/c you find several configs of parameters that fit the moments. But they can imply different underlying model dynamics, which is why you forecast poorly.

Wiki: an overfitted model contains more params than can be justified by the data.

Ok, I'm just confused b/c estimating a functional approx should still kinda tell me when I'm overfitting [i.e. if k_{t-1} doesn't belong in the model]

its coeffs should be set to zero?)

Let's try to understand $\alpha = (\text{ngrid}^2 \cdot 1)$
We could $\alpha^r := \text{reshape}(\alpha, \text{ngrid}, \text{ngrid})$
to get 4 "points", each characterized by
4 values, 2 pertaining to k_{l+1}^{-1}, 2 to fe.
I'm tempted to interpret these two points as
"start" and "end" of the finite-element.
Wait: maybe actually α contains just the
heights (z-values) associated w/ each
(x,y)-point contained in ngrid x ngrid.
⇒ yes! That's it. what proves it is setting

$\alpha =$ vec

1	1	1	1
0.66	0.66	0.66	0.66
0.33	0.33	0.33	0.33
0	0	0	0

$fe_{\text{dim}+1}$-dimension (y)

k_{l+1}^{-1}-dimension (x)

So $\alpha(1,4)$ says that for $(\text{felgrid}(1), \text{fegrid}(4))$, $k_l^{-1} = 1$.

⇒ This has several implications:

1) $\alpha \geq 0$ makes sense b/c k_+^{-1} (= z-value) may never be negative, and $\alpha < 1$ too

2) If I write reshape(α, ngrid, ngrid), I'd expect α^Γ to be

   2.1) increasing in the row dim (as k_+^{-1} is higher the higher k_{+-1}^{-1})

   2.2) a parabola in the column dim (as k_r^{-1} is higher the higher $|k_{+1+-1}|$)

So in principle, if I restrict $\alpha \in (0,1)$,

- k^{-1} should never be negative ✓
- k_Γ can still be neg b/c k^{-1} might still explode

Ryan meeting — 17 June 2020

- Need curvature in fe dim, not in k-dim
  → 2 points in k-dim may be enough.
- Increase k^{-1} support → k_{min} ⇒ should be bound on α.
- Plot ACFs at starting point vs. endpoint and see if the solver pushes in the right direction

- Lsqnonlin better → b/c it uses info on every moment, not just the quadratic form

↳ take weighting matrix and multiply residuals one-by-one.

- do fe & k go off the grid a lot?
  Extrapolation isn't a big problem here.
  In higher dimensions (3D grids or more) extrap. could have jumps.

- Simulate model to get data and see if you can match those.

Add to the target vector all the gridpoints at 0
add a weight to α_0 so the only ones are
a updated when data really wants it to.
When a bump appears, then the robot
only has info on that part of the grid!

↳ Only add a weak (small weights) prior of upon which
the AR(1) is a bad description of the
anchoring function.

↳ This is to examine if the robot only
visits certain areas of the grid and therefore
does whatever it wants in the other areas.

↳ This is the finite elements problem: it needs info
on points in every element to be identified.

- Ryan also agreed that w/ sufficient # of finite
elements, you can match any pdf-al form,
fe^2 or whatever.

- He also agreed about my interpretation of α: they are the heights (z-values).

- He said one more thing which I seem to have forgotten... :S Ah I know: he said that the ACFs are likely to be correlated at different lags, so even if I have 45 moments, many of those may not be actually contributing (much) info. Therefore, adding more moments by adding more ACF lags won't add much independent info. For this reason, if you have 45 moments, you want to be <u>well</u> under that in terms of params b/c you may have much less truly independent moments.

Work after → no more explosions

✓ 1) Increase kdgrid to [lb, ub] on $\alpha = [0, 1]$

✓ 2) Should decide on fe: what values does it take in a normal simulation? For α_0, [-2.6, 3.6]

✓ 3) Increase gridpoints in fe-dim, decrease in kd-dim

nfe = 10 nkd = 2 → more robustness

Experiment w/ nfe.

✓ 4) Plot ACFs ⇒ Pushing in the right direction, just not enough.

✓ 5) Implement bsgnonlin

✓ 6) Add weights to the prior, α_0 in bsgnonlin.

✓ 7) Estimate α on model-simulated data

→ There aren't really bumps, it's more like these ridges might be there b/c that's where we had data? But the flaps at the sides persist.

18 June 2020

→ It seems to do a great job initially, but when the

norm is around 0.12, it keeps converging, but very slowly.
It stops prematurely, but it's really close to matching the moments perfectly.
Yet the implied model dynamics are quite diff.
It's not quite matching the moments and it's also not quite at the true coeffs, meaning that $\alpha - \alpha^{true}$ is a vector w/ sig entries
Let's try to increase MaxFunEvals first to see if w/ more time the solver can get there.
Mm. The fct-val is decreasing but doing so very slowly. What does that mean?
It might mean that the objective function is very flat: since the deriv is close to zero, the solver is only taking small steps but it most likely has a long stretch of almost flat surface ahead of it, so it won't quite get there.

Better conclusions to see where potential instabes lie:

1) Close but not close enough, doesn't converge
2) ACFs: pushing in the right direction, but not enough
3) I don't really see bumps when I assign weights
   → what I do see are ridges
   ⇒ it might be that certain f_E-values do not occur in the simulation ⇒ check!
   ↳ It does seem like not all values in the f_E-range do occur: outside of $[-3.8, 3.8]$ seems to maybe not occur.

- Restrict to $f_E \in [-4, 4]$ ⇒ it doesn't fare much better, but a little, doesn't converge, loss=0.06.
- Restrict to $f_E \in [-3, 3]$: now you get "f_E was nan" error ⇒ apparently f_E is exiting the grid. It's not doing an awful job though.
- Restrict to $f_E \in [-3.9, 3.9]$: also not awful, but not great either.

⇒ I don't think we have the FEM-issue that we don't have values, potentially at the outer edges of the fe-grid, but that's ole ... or?

⇒ Can it be that I can't estimate the α's associated w/ the ends of the grid if those simply don't show up?

I'm now thinking that if the true coefficients α^{true} are 15×1 over fegrid, then maybe I can do better by estimating <15 coeffs on a smaller fegrid.

(nfe_true = 15)

loss = 0.1

- nfe = 12: does match moments better. doesn't converge
- nfe = 9: -11- maybe
- nfe = 3: -11- but the approx is rough and so it delivers a sim gain history that's quite different from the original.
- nfe = 12 and fe [-4, 4]: actually worse

The smaller the (fx- or kt-) grid, the more the simulated k^{-1} will look like this: ⋀⋀⋀
b/c the more points will belong to a particular set that generates a particular k^{-1} value.

Let's now plot α^{true}, α_0 and α^{opt} and then think
Hm wait - looking at this plot, it seems to me that in many instances, the α's are moving in the right direction, and in many, they remain close to the original. It seems like the solver may have gotten itself stuck in a local sink. Let's try random starting points

- same thing
- also, I notice that a random set of α's already does a good job at replicating the ACF
  a 0.5 starting point: same thing.
  ⇒ the α^{true}, α^0 and α^{opt} graph is very help-

Just here: you can see that things are moving in pretty wrong directions. The fact that I'm still close to matching the moments does kinda suggest that I need to reduce the # coeffs b/c I'm under-identified.

- Smaller fe-support? $fe \in [-4, 4]$
  → this gets closest to matching the moments, resnorm ≈ 0.01 but doesn't converge.
  → and indeed αs move in the right direction
- ... and smaller kilgrid: $k^{-1} \in [0, 0.5]$
  → nope.
- Smaller fe-support & nfe = 12
  → It seems to me like this is closer to α^{true} at the edges (b/c fe-support smaller), but of course, it matches moments less.
- — II — & nfe = 9 ⇒ finds a min, but it's a higher resnorm ≈ 0.24. the problem now

is that at this low nfe, the estim is imprecise.

Let me try a last thing: nfe = 3 and nfe = 11

- nfe = 11 ($fe \in [-4, 4]$)

  seems worse than fe = 12 ...

- nfe = 3 ($fe \in [-4, 4]$)

  Exactly what I thought: it converges to a min in < 10 sec and it seems to span the truth nicely. The issue is that nfe is so small that the approx is very rough.

- nfe = 6 ($fe \in [-4, 4]$)

  Finds a min too, same stuff.

- nfe = 20 ($fe \in [-4, 4]$)

  Let me bet that it can fit moments well but won't converge. It didn't, at resnorm = 0.08

⇒ Signs are really suggesting that I have too many params estimated: overparametrization

One more thing: I may be using the ACFs wrong.

Since I'm using for a lag k the entire ACF matrix

$$\gamma_k = \begin{bmatrix} \gamma_k(\pi) & \gamma_k(\pi, x) & \gamma_k(\pi, i) \\ \gamma_k(x, \pi) & \gamma_k(x) & \gamma_k(x, i) \\ \gamma_k(i, \pi) & \gamma_k(i, x) & \gamma_k(i) \end{bmatrix},$$

at each lag I thought I had 9 moments, but actually, I only have 6 independent ones (3 are repeated). So I have $k = 0, 1, \ldots, 4$, i.e. 5 ACF matrices, each w/ 6 moments, so at best, I have $5 \cdot 6 = 30$ moments. That's why $nkl \times nfe = 2 \cdot 15$ may be too many params to estimate. And if I only use the "own ACFs", then I only have $5 \cdot 3 = 15$ moments.

⇒ So at first let's try to only take the upper triangle of the moment matrix.

Interesting: Ω only has the "repeated" 19 June 2020
feature at lag 0. I have a suspicion why that
may be the case. Hamilton, p. 266

$$\Gamma_j \equiv E[(y_t - \mu)(y_{t-j} - \mu)']$$

So e.g. $E[(x_t - \mu_x)(i_{t-j} - \mu_i)']$

$$\neq E[(i_t - \mu_i)(x_{t-j} - \mu_x)] \quad \text{if } j \neq 0.$$

This is only true at lag 0, i.e. when $j = 0$.

⇒ this means that using k lags, I have $(k \cdot 9) + 6$
moments b/c I need to exclude 3 at lag 0.
$4 \cdot 9 + 6 = 42$.
But you know, they could still be correlated.

Ok. Now: let's simulate some data using very
few params nfe = 9. Now the estimation should
be able to uncover α^{true} perfectly.

- Overfit: nfe = 15. → doesn't converge
- nfe = 9 doesn't work either!

Try w/ nfe = 6. Sim true data w/ this and try to fit it.

- estim w/ nfe = 6: it still doesn't converge!
  → maybe I do need to take out some moments after all!

→ using the own ACFs does improve the situ.
Let's shrink fe-support to $[-4, 4]$
→ doesn't help a lot. None converge.
See the issue is that already initially, although I have an agnostic initial point of 0.5 everywhere, I match the moments quite well. So in a way, the moments aren't super informative.

- Estim w/ nfe = 3. (back to fe $[-5, 5]$)
  → same as yesterday: converges, but it's a rough approx and I don't know if it's unique.

Let's go back to truth w/ nfe = 15 and see how the smaller moments vector affects it.
→ Ah I can't b/c there a larger moments vector was created. Grr...

Why is the moments vector well matched for any old α? nfe=6 so I est. 12 params w/ 15 moments → maybe I'm still overfitting?

The thing through is that the estim behaves the same way for less moments as for more, except I guess the moments are matched better now (w/ fewer moments) (on the order of resnorm ≈ 0.004). But the behavior is the same: very fast jumps initially, then a long, painful, flat objective plot that never converges, unless I underfit considerably, but then the approx is crude.

→ Why are the moments uninformative? Can it be that somehow the gain doesn't make a big diff for model dynamics?

What might be the case is that the moments in the model are very highly corr, since

$$x_t = -bi_t + f_p + \varepsilon_t^n$$

$$\pi_t = \kappa x_t + f_a + u_t$$

$$i_t = \psi_\pi \pi_t + \psi_x x_t + i_t$$

Recall also that f_p moves a lot more than f_a, and κ is small, so basically the model "ignores" the effects of k^{-1} on x_t, not on π_t, and since ψ_x small, it doesn't respond either. Let's try 1) a higher ψ_x

2) a lower α (higher κ) e.g. 0.5, 0.6 or 0.7

1) $\Psi_x = 1$ Regen true data

→ doesn't converge

- still close to matching moments
- but it doesn't get α right

2) $\Psi_x = 1$, ~~$\alpha = 0.5$~~, regen true data

↳ sim explodes, try $\alpha = 0.7$

- Now it found a min
- matches moments not bad
- but $\hat{\alpha}$ are kind of uniform

It's interesting. I wonder if this is suggesting that f_E & k_{t-1}^{-1} are multicoll b/c clearly the code sees α^{true} 

as equivalent w/ $\hat{\alpha}$ 

That is, in terms of matching moments, it doesn't matter if k_t^{-1} responds strongly to

- fe, and not at all to k_{t-1} (or vice versa)
- or medium to both,

→ the same moments are the result

The strange thing though is that ysim(k^{true}) the same moments as ysim($\hat{k}$), even though $\hat{k}$ is basically constant while k^{true} fluctuates a lot and the two are not equal?

→ does that mean that ACFs aren't very informative, b/c even very different ysims exhibit a similar ACF-structure in the model?

As a next step, I guess I should estimate the univariate anchoring function. If multicoll is the problem, then that should work.

Estimate univariate anchoring function — 21 June 2020

- Real data:
  - nfe = 6 : finds min
    It doesn't move a lot from the initial moments,
    but it does move some
  - nfe = 6, rand start: finds min (rng(10))
    Moves a lot from initial moments towards
    the true ones, but it does not seem to get
    quite the same α's, although close.
  - nfe = 6, start at 1.1 : doesn't find min
  - nfe = 6, start at 0.9 : doesn't find min, even
    less than before.
    start at 0.2 : finds sol and moves
    towards moments
    and it finds pretty much the same sol.

This $\alpha^* \approx [0.8, 0.13, 0.0002, 0.005, 0.006, 0.98]$

- nfe = 6, start at another rand (rng(5)) : finds min
  Yes! Same sol!

- Sim data: (nfe = 6)
  - nfe = 6, start at α_0: no sol.
    didn't move in the right dir, whether in movements nor in α.
  - nfe = 6, start at rand rng(10): no sol
    same.
  - nfe = 6, start at rand rng(5): local min
    pinned down both very closely!
    Both in α and in ACFs.
    α_0 was similar to α^{true}.
  - nfe = 6, start at 1.1: no sol, moved a bit.
  - —\(— 0.9: local min
    going in the right dir, but not close enough
  - nfe = 6, start at 0.2: no sol, moved a bit

⇒ Interesting that real data vs sim_data behaves so differently, real data is robust, while sim-data not.

Do we see the same for 2D anchoring fct?

Real data: 2D

- wfc = 6, start at α_0: local min, move in right
- wfc = 6, start at neg(10): local min, —11—
- wfc = 6, start at neg(5): local min, —11—
- wfc = 6, start at 1.1 : local min
  thinks more in the right dir, get
  similar thing
- wfc = 6, start at 0.9: no sol. & barely moved.
  loss was also huge: seems like far from sol.
- wfc = 6, start at 0.2: local min, moved in the right dir. but has big flaps at the edges.
  Otherwise looks like the others.

Sim data: 2D

- nfe = 6, start at α_0 : no sol, though moved in right dir
- nfe = 6, start at arg(10) : —"— : in both, not enough
- nfe = 6, start at arg(5) : local min
  It finds well, b/c α_0 is close to α^{true}.
- nfe = 6, start at 1.1 : no sol, but boy, did it get close! Both in α & in ACF.
- nfe = 6, start at 0.9 : no sol, but again, darn close! Here, as at 1.1, the middle of the fe-space was identified well, while the the edges, no. It's α_1, α_2 and α_{end-1}, α_{end} that are badly identified.
- nfe = 6, start at 0.2 : no sol. but again, not bad fit in the middle.

⇒ Here's my take on things: things work like a charm on real data, are robust. Not on simulated data. Both for 1D & 2D anchoring fct.

→ so this means that the sim data doesn't generate as much fe's in the "edges" of the fe-support as real data. Therefore the sim data aren't ID-ed for the "outer" (first & last) α's.

→ There is no diff btwn 1D or 2D anchoring fct.

- 2D, nfe=6, start at 0.2, fe ∈ [-4,4]: no sol
- -11- + fe1 ∈ [0,0.2] : no sol
- -11- + fe1 ∈ [0.1,0.9] : no sol
- -11- + fe1 ∈ [0.2,0.5] : no sol and this was bad.
- -11-, fe1 ∈ [0.1,0.9], fe ∈ [-4,4]: local min but not a great fit.
- -11- but add lb = fe1min, ub = fe1max: same
- -11- fe1 ∈ [0.1,0.8]: same.

same: but start at 1.1: no sol. Worse fit now.

⇒ Somehow my feeling is that the sim data doesn't generate enough variation in ACFs to pin down the corrs deterministically at the edges. Actually, using 6 corrs on a smaller fc-support doesn't make sense b/c the fit will by def deteriorate: e.g. $\hat{\alpha}_1$ will never fit α_1^{true} b/c they pertain to a different fc-space. (element)

Starting from 10 diff starting points 22 June 2020

1) The conclusion that real data behaves better than sim data is also here: things are D-cd w/ real data, while for sim data they're not, at least in terms of estimated $\hat{\alpha}$.

2) common across all groups is that resids are very similar too, indicating weak identification.

3) Also common is that ID seems weakest at the edges of the supports, but not strongly.

4) A difference between the 2 cases is that for 1D, the sim-data estimates are very consistent, just wrong. Is there a local sink?

↓ increased MaxIter & MaxFunEvals and tol and...

... sim-data 2D converged!

sim-data 2D visually suggests I might be inverting something somewhere...

... sim-data 1D converged!

I'm not sure if this suggests an inverting issue, but it might be

→ just checked thru all my objective fcts and simulating fcts, and they don't seem to invert anything wrongly. The issue is k^{-1} vs k. But it doesn't seem to be happening. So how about the ACF-generating codes?

→ They look totally fine!

Cont w/ doing moments for the 2D case!

Now I want to see: 23 June 2020

1) Do we move a lot from initial α's & moments?
2) Do we move in the right direction?
3) Can we actually get an exitflag of 1 if we increase Rvings?

Let's always use nfe=6, start at rand, rng(5)

2D Real data

1) Yes!
2) Yes!
3) No.

Gff the step tol is stricter, it iterates out.

2D sim data

1) Yes
2) yes, not always
3) It initially gets flag=3

→ No: either the step is too small or it iterates itself out

→ My feeling is still that it's the end when the movement stops

1D Real data

1) yes! Very sim to 2D.
2) yes! —11—
3) Well, you can change flag 2 to flag 3 and get essentially the same result.

And no, otherwise it times out.

1D sim data

1) Yes, but α_0 is close
2) yes, but q_0 is close
3) → also here you can get 3 instead of 2 by lowering TolX, and get the same result.

→ Moments aren't informative: decrease nfe!

But the problem for the sim data is that you're underparameterized so you have a hard time fitting.

Exitflags in lsqnonlin

1 = the 1st order optimality measure was < stky

2 = change in X (stepsize) was < stky

3 = change in objfct components were < stky

In optimset, you can only set TolX (→ 2) and TolFun (→ 3). You need optimoptions to set "OptimalityTolerance" (→ 1).

⇒ You get exitflag = 1 if the first-order optimeasure < crit. This means for the fact the gradient is never close enough to zero, even if the fct value is only changing a little bit. ⇒

→ this really means that there are several things you can do to be just as close to the moments, but you can't get any closer. You're stuck at an indifference curve between several α's, and you can't get to a lower indifference curve.

So let's try on 1D Real data w/ nfe ↓

nfe = 4 Flag: 2, 3, 0, 0, 0

(I always increase the stepsize tol of func tol or MaxFunEvals or MaxIter)

⟶ exitflag = 0

nfe = 2 Flag: 3, 2, 3, 2, 2, 2, 0

⇒ even nfe ↓ isn't enough. So that must mean that there's a combined issue: not only are the moments not informative, maybe the support of f_ε is too small/big.

- nfe = 6. fe ∈ [-6, 6] Flag : 3

6	[-5, 5]	3	
6	[-4, 4]	3	
6	[-3, 3]	error	
6	[-3.5, 3.5]	3	
4	[-3.5, 3.5]	3	but Opt T-1 was lower than ever (0.0253)

→ Decrease Tol : Flag : 2, 2, 3

4	[-3, 3]	error.	
4	[-3.2, 3.2]	0	
4	[-10, 10]	2	0.0235
2	[-10, 10]	2	
2	[-3.5, 3.5]	2	

But nfe = 2 is very-very rough.

Let's try the same thing w/ the sim data w/ a strong weight on the prior, and starting w/ a diff. rand that's further from the truth

(Btw from rng(5), nfe = 6 fe ∈ [-5, 5]
has a better fit than fe ∈ [-3.5, 3.5].)

rng(8): its initial coinflag = 0, b/c it moves a lot but partly in the wrong dir, although mostly in the right.

Now Wprior = 100, flag = 0.

→ it stays on the prior everywhere.

Wprior = 1 flag = 0

→ stays on it almost everywhere.

Wprior = 0.01, flag = 0

→ moves pretty much everywhere (except at 4)

Wprior = 0.1, flag = 0

→ same.

⇒ There aren't really signs of the FEM issue:
it's more just that I can't quite match the moments. It seems like the model-implied moments don't offer as much wiggle-room.

An approach can be to crank up TolX and among the estimates, only keep those where flag is 3.

1e-15

- Indeed, cranking up TolX and TolFun, for 1Dsim_data, you get all 0 flags.
- Same TolX & TolFun, real data: all 0 flags.
  ⇒ set TolX = 1e-15 but TolFun to stay nicer like 1e-9 or 1e-11.

Peter meeting 23 June 2020

A DSGE estim w/o a Bayesian prior you'd see this same thing: flat loss portions, ridges, multiple peaks

Suggestions:

- recover true params from sim data
  ↳ obj_fct evaluated at true α better be better than its value at any $\hat{\alpha}$.
  If not, then the problem has to do w/ the obj fct.
  A consistent estimator requires the loss to have a peak at α^{true}.
  - coding error
  - or ID problem
  → If yes, then the trouble is just coaxing the solver to the sol. → Then you can just code

along the obj. fct-values associated w/ the candidate $\hat{\alpha}$.

Same resids: Sum obj fct is very flat but has curvature. That's fine b/c it's hard to estimate a large # of coeffs w/ the amount of moments.

Or: multiple peaks w/ very flat surfaces

Try: fix all α at α^{true} except a single one, α_i. Then eval the obj fct at diff α_i; and see what the fct looks like.
→ Do this for all the elements α_i.

If there is indeed weak ID, then need to bring in more info. Either by adding Ts or $E(\cdot)$ or by restricting the coeffs when the use of

info outside the model.

Scharfstein & some else: Sequential Monte Carlo

→ they try to replicate Schmiff-Grose & Uribe's paper on News shocks (Econometrica)

→ Find that MCMC of S-G & U got stuck at a local max

Another case: Monika Pinzzesi's paper

In the orig paper, the max was a local one.

Guard against the embarrassment better than S-G & U by being honest about it, say "the likelihood is very flat", run for 2 days find max and we have the feeling that if you run it for 2 months, they'd get a better max, then put in a footnote that explains how you got it.

Work after

1.) obj (α^{true})

2) obj (α^{true} except at one)

Let's restrict ourselves to the 1D case.

So my strategy now is to do a bunch of reruns and compare for values. Overnight are running 2 pbs-es w/ n=1000 and n=10 000. Let's see what we get!

They are not bad, but not quite 24 June 2020

There either: the minimum loss I can get is $\simeq 30$.

→ I tried a "smart" coaxing where you search in the neighborhood of the best results so far, but it's actually not doing great b/c the point is that each local min appears to be a vast flatland; a very mean local sink. You don't find anything

better around there. That's why I think that a brute force search w/ $n = 100\,000$ (or more? my goodness)

Presentation stuff learned from Alessandro Villa:

1.) Nice serif-font
2) Black header, white slide
3) Blue bar that shows progress
4) Content: simple model → full dynamic model
5) Work w/ "parse"
6) Emphasize "behavioral"

he uses OpenMP & Cuda to use GPU computing and he parallelizes on CPU & GPU

Approxes "max" w/ sigmoids but looks like the kernels in nonparametric stats or like the neurons in a neural network.

Ryan meeting 29 June 2020

Can do a gridsearch that's guaranteed to converge
- take a grid for each α
- optimize over that α, keeping the others fixed, and then do that for each

$\Rightarrow$ could help to falsify whether there is a min.

Other thing: take a 6-dim grid
evaluate the likelihood at the $6^6 = 46000$
to see how the likelihood looks

- Fig 9. Lines don't hit 0 $\Rightarrow$ suggests that the truth is a global opt.

Sim data:
- Is the truth too exotic? α^{true} are non-conv.
Try to pick a nicer point.

Worry: We might punch the PC to get the global optimum but there are a bunch of points that look similar.

↳ Scrap that: I don't worry about the local minima b/c they are not similar to the global.

When GMM doesn't work:

- add moments
- remove params
- change model

1) Warm see $\alpha \in (0, 0.1)$ → simulate model at that
2) Run $n = 100\,000$, investigate the top 10
   - policy fct
   - residuals
   - loss function for those top ten points.

Dream scenario:

After $n = 100\,000$, you compare the top 10,
then only A is close to zero

$\Rightarrow$ Do for real data, same, get $\alpha^{true, data}$

$\Rightarrow$ Redo simulation w/ $\alpha^{real data, true\ as\ truth}$ to prove that estimation works.

↳ Like a proof of concept that my estim strategy works.

Add moments:

- can be calibrated moments
- or strict priors e.g. function should be monotonic or convex

convexity: take α's

take diffs twice

$\Rightarrow$ should be positive

2^{nd} term $> 0 \Rightarrow$ penalize the loss fct for that.

Make the penalty convex (increasing in

deviation) so that loss doesn't become
kinky

calibrated moment: e.g. average $k = 0.05$
in the simulation

Work after Materials 35

2.1) Evaluate loss on a 6^6 grid → gsubbed.

2.2) $\alpha^{true} \in (0, 0.1)$ and convex → whoa! much better!

2.3) 100 000 starting points, top 10 → been running on the server since morning

2.4) Add moments → strict priors convex → calibrated moments

→ Only the $f_6 = -5$ coefficient seems to be unidentified.
Shrinking $f_6 \in [-3.5, 3.5]$ helps get a better fit

Good news: (1D function) 25 June 2020

making the α^{true} i) convex ii) $\in (0, 0.1)$ iii) pertain to $f_\alpha \in [-3.5, 3.5]$ makes the estimation able to nail $\hat{\alpha} = \alpha^{true}$ exactly.

↳ So the estimation is able to recover the truth.

But for real data, the sim is still "bad".

I have reason to suppose that the truth is convex, and I can shrink the f_α-support I consider to $[-2.5, 2.5]$ [but not further!], but I still seem do be unidentified b/c starting points really matter, and the solver mostly iterates itself out.

⇒ Add moments! Try it out on the sim data, which now is working brilliantly!

1) Convexity is the first moment I should add.

$$\hat{\alpha}'' > 0$$

But the moment condition itself should be convex (:P) in the sense that the more it's violated, the higher should the penalty be

→ what I'm trying is

$$\text{sum}\left[\left(\hat{\alpha}'' < 0\right)^2\right]$$

Dam. That doesn't converge.

Try max instead of sum. → Not good!
Doesn't impose convexity. Actually sum doesn't work either! It doesn't impose convexity.

Try to discard $\hat{\alpha}$ immediately if not convex.

Nope that's even worse.

→ Yay! It works! ⇒ moment = sum(secondDiffs(secondDiffs<0)^2)

Try on real data. Still iterates itself out.
Also: a nonconvex one still made it.
Need to penalize more. (Multiplied by 10, checked and it still works for sim_data.)

So try to add the average gain-value calibrated moment: $\text{mean}(k^{-1}) = 0.05$
Now the sim data doesn't quite hit the truth (of course b/c I'm imposing sthg that ain't true in the "fake true world"). But it's still damn close.

Let's see the real data: it still iterates out.
Is it robust at least? No.
And convexity penalty still seems too low.
↳ Try 100. (checked that sim data still works!)
↳ Try 1000 (seems to finally

work on the real data, but now the simdata iterates out, although it's still at the correct thing.
↳ W/o the calibrated moment, it works.

Now real data can't tolerate $f_e < 3.5$.
Hmmm.... But that's not useful. My problem rather is that whatever I do I still get the same features as before: difficulty in converging, iterating out. I mean, it's ok if the moments aren't well matched if the optimum is unique. But I don't find a lot of optima and they seem not unique.

Can't here tomorrow

16 June 2020

As a first step I'll clean up a bit so whatever we can truncate can be tweaked at the same place.

Now I'm seeing that even for the sim-data, not all seed starting points are able to find the optimum. The solver iterates itself out and again it's mainly the lowest fe-valve that seems unidentified.

In this case, adding the calibrated moment helps getting closer to the truth a lot, but unfortunately you don't see this in the residuals b/c of the penalty.

rng(0), for example, does really badly!

resnorm = 75.

Again, adding the calib. moment helps a lot, even though you still iterate out.

rng(4) too, calib moment helps.

rng(7) gives $k_1^{-1} < 0$ and therefore initial point is bad.

rng(8) too!

Out of rng(0) - rng(9), 4 don't report success! Good to

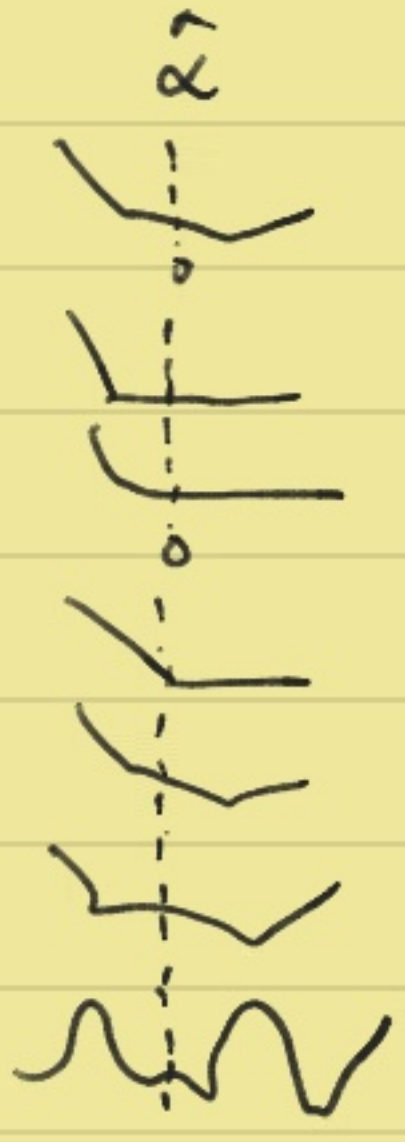
know for real data b/c then adding the calib moment helps nudge the $\hat{\alpha}$ close to α^{true}.

So this may suggest that for true data we can get there too as long as we nudge enough.

So turn to the <u>real data</u> and play around w/ the things we have one-by-one:

Defaults and:

	flag	resnorm
· nfe = 6	0	resnorm = 551.
· nfe = 5	0	463
· nfe = 4	2	, resnorm = 469
· nfe = 3	2	resnorm = 471
· nfe = 8	0	549
· nfe = 12	0	551
· nfe = 30	0	535



⇒ This seems to suggest that we have about 4 informative

moments. Add calib moment to nfe = 5

→ still 0 flag. ∩

↑ Wmean from 10 to 100 → still 0 flag ∩

Defaults and fe-support

fe ∈ (·,·)	flag	rcsnorm	shape
[-4, 4]	0	553	
[-5, 5]	0	557	
[-6, 6]	0	560	
[-10, 10]	0	560	
[-3, 3]	0	526	
[-2, 2]	0	461	
[-1, 1]	0	508	

↳ You get some fe=nan

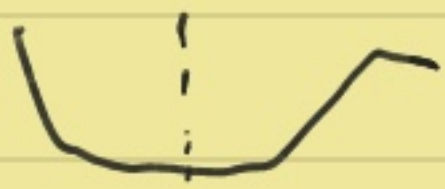
- [-1.5, 1.5] 2 501
  ↳ still
- [-1.8, 1.8] 2 500
  ↳ still

key terms

"flipped classroom" := students do all the talking

→ but you can finesse that!

↳ Sorry it's been a busy day just thinking about Covid and teaching and all ...

fe	flag	resnorm	
? [-0.5, 0.5] (barely moved from initial)	2	610	
[-2, 2] & add calib moment:	0	523	

Defaults and ub & lb

lb & ub		flag	resnorm	shape
0	1	0	551	
0	0.5	0	550	-11-
0	0.1	0	551	-(1-

0.001 1 0 559

0.01 1 initial point is a min 4.5e+21

↳ it shifted the initial point to comply w/ lb!

default & play w/ moments.

Default is Wprior = 0

Wconvexity = 1000

flag 0
res= 551

Wmean = 0

w/ change the mean moment from $|(k^{-1}-0.05)|$ to $(k^{-1}-0.05)^2_j$

Wmean = 10 0 552

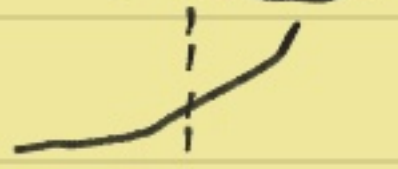
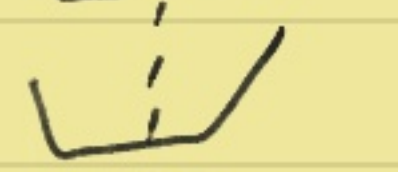
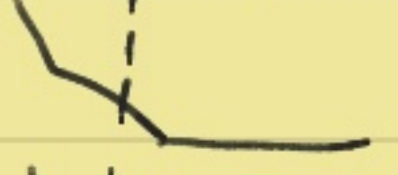
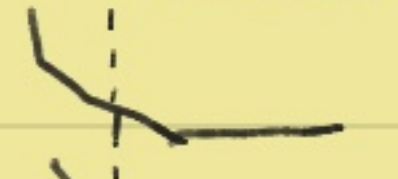
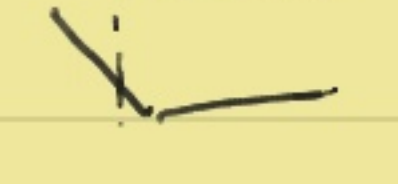
Wmean = 100 0 567

Wmean = 100, Wconvexity = 10000, 0,582.

Wmean = 1000, Wconvexity = 1000. 0,639

Back to default, now change starting points

Default is the U-shaped start by fk_smooth

Start	flag	resnorm	shape
0.2	0	491	
0.05	0	521	
rng(0)	0	653	
rng(1)	0	468	
rng(2)	0	563	
rng(3)	0	453	
rng(4)	0	505	
rng(5)	0	528	
rng(6)	0	490	
rng(7)	1	initial point is min, 495+21	
rng(8)	-11-		
rng(9)	0	467	
rng(9) + Wmean=10	0	466	
rng(9) + Wmean=100	0	507	

Some random attempts

Default + Wmean = 100 0 567

Default + Wmean = 100 + fe ∈ [-2, 2]

0 479

Default + Wmean = 100 + fe ∈ [-1, 2]

2 522

Default + Wmean = 100 + fe ∈ [-0.5, 2]

now it complains fe nan

Default + Wmean = 100 + fe ∈ [-0.8, 2]

fe was nan again

Default + Wmean = 100 + fe ∈ [-1, 1.5]

fe was nan.

But ok, we can go down to fe ∈ [-1, 2]

Default + Wmean = 100, Wconvex = 10000, fe ∈ [-1, 2]

0 486

same but nfe = 5 0 485

—||— nfe = 4 2 487

—||— nfe = 3 3 482

I again conclude that I seem to have 4 informative moments (incl. the convexity moment)

Default, nfe = 4 flag 2

Default, nfe = 4, Wconvex = 0. Gives almost the same thing, so it doesn't seem to constrain a lot. (still flag 2)

Default, nfe = 4, Wconvex = 0, Wmean = 200 flag 3 & almost unchanged results

→ So even w/o the extra 2 moments, nfe = 4 solves. So we seem to have 4 moments. The convexity moment and the mean moment don't add much binding info b/c they seem to be fulfilled by the sol anyway. But this would in principle mean that nfe = 6 (or 5) should solve w/ the 2 extra moments.

Try wfe = 4 w/ diff starting points

Default, $W_{mean} = W_{kurt} = 0$, wfe = 4 0.52, 0.01, 0, 0.0026

3 462

same, but $\alpha_0 = 0.05$

3 464 [0.43, 0.01, 0, 0.0044]

same, but $\alpha_0 = 0.2$

3 462 [0.51, 0.01, 0, 0.0024]

same, but $\alpha_0 = rng(9)$

0 467 [0.37, 0.001, 0.0002, 0.0074]

same, but $\alpha_0 = rng(5)$

3 464 [0.42, 0, 0, 0.0037]

same, but $\alpha_0 = rng(0)$

3 464 [0.42, 0, 0, 0.0057]

Ok, the estimates are very similar. In all of them, weak convexity holds, so that moment doesn't add more info

You don't believe it, but 27 June 2020

$n = 100\,000$ finished on the server after running for almost 70 hours!

Here's the best it found:

α^{true}	$\hat{\alpha}^{best}$
0.8147	0.9741
0.9058	1
0.1270	0.8881
0.9134	0.9325
0.6324	0
0.0975	

w/ a resnorm = 34.5273 and min_idx = 41885

(rng('default'))

Not bad but not great. Especially as it didn't improve on boot 10 000, it's the same point!

↳ I'm gonna return to real data instead!

Real data Default, nfe = 4, no additional moments

Now add Wconvex = 1000: 3, 464 [0.43, 0.0005, 0 0.0041

nfe = 5, Wconvex = Wmean = 0: 2, 471, [0.38, 0.29, 0, 0, 0.0015]

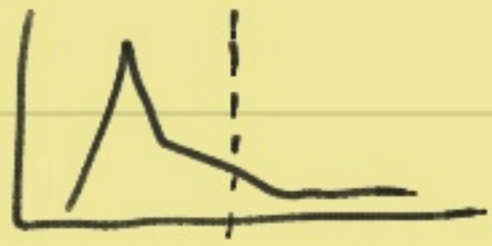
Ha!

nfe = 5, Wconvex = 1000: 0, 463 [0.71, 0.0018, 0.002, 0.005, 0.01

nfe = 6, Wc = Wm = 0: 2, 462 [0.0065, 0.31, 0.05, 0, 0, 0.0025]

→ this seems like it's

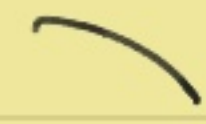
overfitting



nfe = 6, Wconvex = 1000 (default): 0

nfe = 6, Wconvex = 1000, Wmean = 1000: 0

nfe = 6, Wconvex = 10000, Wmean = 1000: 0



Fine but it seems like we have 5 moments initially b/c we can match nfe = 5. What is surprising is that these convexity & mean moments don't seem to be very informative.

nfe = 4, Wc = Wm = 0: 3, 462 [0.52, 0.01, 0, 0.0026]

nfe = 4, Wc = 0, Wm = 100: 3, 478 [0.67, 0.03, 0, 0.0006]

nfe=4, Wc=100, Wm=100 : 3, 479 [0.58, 0.018, 0, 0.0027]

let's keep this and add nfe

nfe=5 : 3, 478, [0.88, 0, 0.0008, 0.0061, 0]

↳ the convexity moment is clearly not penalizing enough.

nfe=5, Wc=1000, Wm=100 : 0, 481, [0.77, 0.0026, 0, 0.005, 0.01]

↳ and when it is, it no longer converges.

same, MaxFunEvals = 1000 (instead of default = 500):

2, 481, [0.77, 0.0026, 0, 0.006, 0.01]

same, nfe=6 : 0, 563, [0.07, 0.04, 0.02, 0.01, 0.0039, 0.173]

↳ Given that the two new moments may not always be informative, it may be that we're still only able to estimate nfe=5 consistently.

same, nfe=7 : same.

But how robust is this sol?

nfe=5, Wconvex=Wmean=0, MaxFunEvals=1000

different starting points:

α_0	flag	nonon	$\hat{\alpha}$
fk_smooth	2	471	[0.99, 0.24, 0, 0, 0.0015]
0.2	3	457	[0.86, 0, 0, 0.0061, 0] ! not convex
0,05	—	"	—
ruglo)	—	"	—
rug(2)	—	"	—
rug(5)	0	461	[0.72, 0.0001, 0, 0.0068, 0.0013] not
rug(8)	3	457	[0.86, 0, 0, 0.0061, 0] not convex

⇒ If it finds stop, then it's nearly the same thing!

Old Wconvex = 1000

α_0	flag	nonon	$\hat{\alpha}$
fk_smooth	0	463	
0.2	0	460	
0,05	0	463	
rug(0)	3	462	[0.73, 0.0013, 0, 0.0054, 0.01]
rug(2)	0	?	
rug(5)	3	563	[0.03, 0, 0.03, 0.06, 0.16] (!)
rug(8)	0	460	

Add Wmean = 700

α_0	flag	nonorm	$\hat{\alpha}$
fk_smooth	2	481	[0.77, 0.026, 0, 0.0058, 0.0107]
0.2	2	480	[0.78, 0, 0.0005, 0.0052, 0.0082]
0.05	0 but nearly converged	481	[0.78, 0, 0.002, 0.0005, 0.0058, 0.01]
rng(0)	0 -11-	491	[0.79, 0.0024, 0.009, 0.0058, 0.009]
rng(2)	0 -11-	574	[0.0258, 0, 0.03, 0.07, 0.1276]
rng(5)	2	482	[0.76, 0.0025, 0.001, 0.0058, 0.01]
rng(8)	0	572	[0.0268, 0, 0.03, 0.0653, 0.7742]

⇒ There are some abominations, (like rng(2)) but normally it converges to the same thing, either close enough, or its neighborhood.

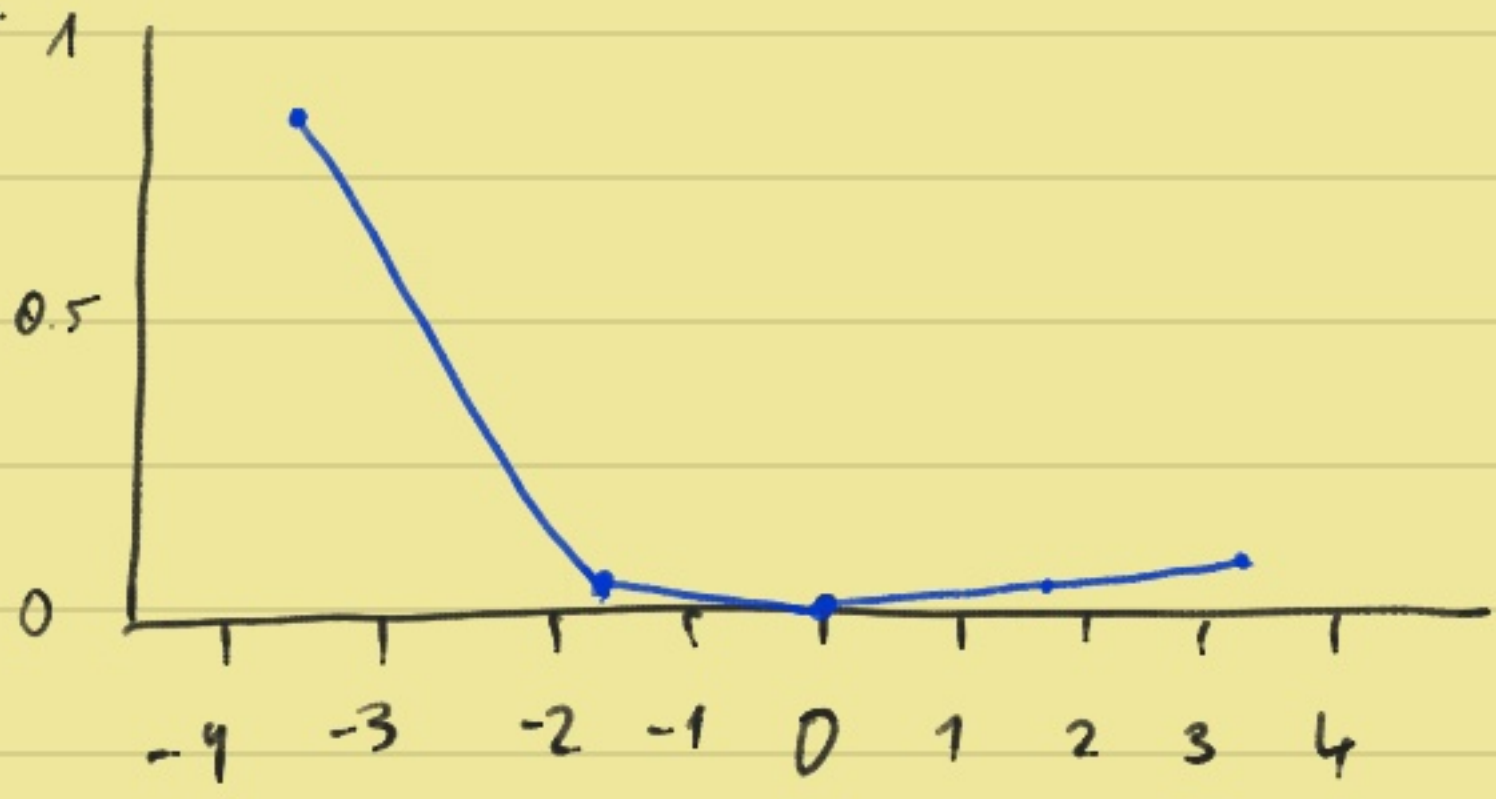
- You need to impose the convexity moment to exclude non-convex sols in the nk=5 case.
- You need to impose the mean moment to make convex sols converged.

→ But w/ these impositions, the following config produces a quite robust result:

n/c = 5, fe ∈ [-3.5, 3.5], [lb, ub] = [0, 1], Wlower = 1000, Vmean = 100, starting point = fk_smooth (or some others)

The sol is for fk_smooth:

$$\hat{\alpha} = [0.77, 0.026, 0, 0.0058, 0.0107]$$



"a laid-back L"

↳ This would mean that people react extremely strongly to big ⊖ surprises, but not so strongly to others. Robust? Trying to set up a server to check!

Peter meeting — 30 June 2020

JPÉ Acemoglu, Robinson & BH

econ growth & democracy

Really nice paper! Use a bunch of diff models.

Dem → growth?

Motivate in 2 ways:

1. "Dem is a weakness b/c slow & dissenting"

↳ for this reason we care about this

2. Mixed results in data:

· on avg, growth ↓ before econs become democratic

⇒ coming up w/ a way to control for that is key for causal effect

⇒ So bunch of ways/models

↳ Get around w/o a simple graph or two-period model

A couple of annoying news/details about the optimal TR w/ estim params. Now I get a loss that's always below $loss^{RE}$. 1 July 2020

- I can't do CUSUM b/c it always explodes (why?)
- I can't do CEMP anit b/c it relies on the ALM formulated as A_a, A_b, A_s, which I don't have any more.

↪ I'm changing these plots a bit, so both losses are next to each other.

Guess I should rework CUSUM & CEMP :S

Wait: I've uncovered a parameterization issue:

1) What should $\alpha =$?
2) Is $\alpha = \frac{(1-\alpha)(1-\alpha\beta)}{\alpha}\zeta \approx \frac{1-\alpha\beta}{\alpha}\zeta$?

↪ I'm relatively sure it's the former!

Try to find in experimentation notes!

Rosen et al Exchange Rate Disconnect Redux
seems to offer evidence for expectations
being rational but moving (learning) slowly

Ryan meeting 1 July 2020

- How much does the loss function change as you go from wk = 5 to 4 to 3?

↗ Ryan & Robert, p. 49 (Mac)

- Show autocorrelogram (covariogram)
  so we can describe where it succeeds
  and where it suffers.

  One might be a good fit for π, the other for output

- Try to not generate the same shocks
- Should it be stationary (i.e. go back to 0)
  when the gain is falling? Eq.

- dolish: I should be worried if ppl adjust their E(·) downward.
- This Friday to 15th July: he's on vacation.
  - 16th/17th is the earliest we can talk if I send an email
  - can send paper/draft, he won't read it during holiday.
  - can still email him

  e.g. Covariograms and some thoughts

As for others' comments: this is a complete paper, with the estimation, it will be. Need to convince ppl externally and internally that it is.

Work after

I see in my Notes for experimentation,
Notes 3_CEMP_estimation_Preston_simulation, p. 61

The Calvo param α

$\frac{1}{1-\alpha}$ = avg (= expected) duration of a chosen price (in periods)

So if $\alpha = 0.9$, then 10 periods (10 quarters)

$\alpha = 0.7$, then $3.\overline{33}$ periods ($3.\overline{33}$ quarters)

$\alpha = 0.66$, then 2.94 periods (3 quarters)

$\alpha = 0.5$, then 2 periods (2 quarters)

Surveys who: Prices change every __ months:

Blinder & Klenow (1998) 4.3 months

Klenow & Kryvstov (2008) mean: 7-9 months

median: 4-7 months

Nakamura & Steinsson (2008) 7-9 months

Klenow & Malin (2010): adjust for sales 6.9 months

Eichenbaum, Jaimovich & Rebelo (2008) 10.6 months

Collard slides, 5, p. 43.

But where do I have this info from? Wiki?

But actually, see Hristov (calvo_hristov.pdf in next > Literature)

He says: $\theta := \alpha :=$ Prob(unchanged)

Expected lifetime of a price is for how many periods this price is expected to remain unchanged = $\frac{1}{1-\theta}$

⇒ So at least that's correct!

Only Calvo is wrong to say that $\frac{1}{4} = \alpha = 0.66$ or $0.75 \Rightarrow$ adjust prices every 3 to 4 months, etc it's only months if that's the period of the model. It's period. Not months.

⇒ We need α's ranging from 4 to 9 months, i.e. 2 quarters seems ideal, or 3 at most.

⇒ $\alpha = 0.5$ or $\alpha = 0.66$.

Maybe $\alpha = 0.5$ is actually fine: prices change on average every 6 months. Seems reasonable.

Now to α: finally I've found where I discuss this in my next Notes 2 "after DW", starting on p. 67 (Mac), bottom.

Mean of empirical evidence is 7.56 months. This is 7.56/3 = 2.52 quarters 2 July 2020

$$\frac{1}{1-\alpha} = 2.52 \text{ quarters} \Rightarrow \frac{1}{2.52} = 1-\alpha$$

$$\alpha = 1 - \frac{1}{2.52} = 1 - 0.397 = 0.603$$

$$\underline{\alpha = 0.6}$$

So we can do $\alpha = 0.5$ as a flex price extreme, $\alpha = 0.7$ as a sticky price extreme, or we can settle for $\alpha = 0.6$ which corresponds to the mean of estimates and to an average duration of ≈ 7.5 months.

$$K = \frac{(1-\alpha)(1-\alpha\beta)}{\alpha} \cdot \zeta$$

and the question is what is ζ?

In CEMP, $\zeta = 1$ b/c there's no demand side.

In Basu, $\zeta = \eta + b$ b/c there's a demand side but common factor markets

In Preston, $\zeta = \frac{\omega + \theta^{-1}}{1+\omega\theta}$ b/c demand side and specific factor markets

ω = elasticity of a firm's real marg. cost function wrt its own output, $y_t(i)$ "

θ = elasticity of demand?

Woodford: ζ = measure of strat comp. in price setting

when $\zeta\uparrow$, strat comp $\downarrow$

Ok but the point is that in any case, λ prop the (1-d) up front.

Prop 3.3 in Woodford

$$g = \frac{(1-\mu s_m)(s_y + s_Y)}{1 + \theta[\epsilon_\mu + (1-\mu s_m) s_y]} \qquad (1.43)$$

So if you ass. no intermediate inputs ($s_m = 0$)

- common factor markets ($s_y = 0$)
- constant desired markups ($\epsilon_\mu = 0$)

$g = s_Y$ and $s_Y = \omega + b^{-1}$ ← my I guess.

Supp there are no intermediate inputs ($s_m = 0$).

Then $g = \frac{s_y + s_Y}{1 + \theta[\epsilon_\mu + s_y]}$

Supp there are constant desired markups wrt levels of output produced ($\epsilon_\mu = 0$)

Then $g = \frac{s_y + s_Y}{1 + \theta s_y}$

Then the only things we need to understand are

1) When specific factor markets, $s_y = \omega$, $s_Y = \beta^{-1}$

while for common factor markets, $s_y = 0$, $s_Y = \omega + \beta^{-1}$

$$\zeta = \frac{\omega + \beta^{-1}}{1 + \omega\theta} \quad \text{vs.} \quad \zeta = \frac{\omega + \beta^{-1}}{1}$$

specific common (Basu)

2) What is ω and how does it subsume the Frisch and other stuff?

1) Ok, I think I see. Recall that a log-lin of the MC-function is

$$\hat{s}_t = \omega \hat{y}_t(i) + \beta^{-1} \hat{Y}_t - (\omega + \beta^{-1}) \hat{Y}_t^n \qquad (1.15)$$

where ω = El of s wrt $\hat{y}_t(i)$

When common factor markets, s_t is the same $\forall i$ and so it's as if $\hat{y}_t(i) = \hat{Y}_t$ in this regard, so $s_y = 0$, $s_Y = \omega + \beta^{-1}$. No distinction.

When specific factor markets, $s_y = \omega$ and $s_Y = \beta^{-1}$.

2) What is ω? It's $\omega = \omega_w + \omega_p$ (1.16)

$$\omega = \omega_p + \omega_w \qquad (1.16)$$

↑ ω_p: - elasticity of MPL wrt output → price demands

↑ ω_w: disutility of work (Frisch?) → wage demands wrt output

Here we have it: Woodford p. 165 — 3 July 2020

W/ common factor markets, the wage is independent from the amount firm i produces (that is, from $y_t(i)$)

and so $S_t(i) = P_t\, s(X_t, Y_t; \xi_t)$

where $X_t = \int_0^1 y_t(i)\, di$. So technically, $X_t = Y_t$ and there's no distinction between $y_t(i)$ and Y_t in terms of effects on MC, s_t. $\Rightarrow s_y = 0$, $s_Y = \omega + \sigma^{-1}$

$\Rightarrow$ this allows me to think of ω as the elasticity of MC to "some relevant output" coming from labor.

$$w = w_w + w_p$$

elasticity of disutility of work (Frisch, see Barro Sala 1, p. 44 Mac) wrt output

elasticity of Y (1.9, p. 148) ↳ = MPL

w_p = El MPL wrt output

So $w = \eta + w_p$

Frisch elasticity of disutility of work

elasticity of MPL w/ Cobb-Douglas $y_t = A_t h_t^{\alpha}$ (and K is in fixed supply)

$$MPL = \alpha A_t h_t^{\alpha-1}$$

$$\frac{\partial MPL}{\partial y_t} = 0 \Rightarrow w_p = 0$$

$w = \eta$ Voilà!

One last thing is η vs η^{-1}

Basu Sum 1, p. 56 Mac

$$\hat{H}_t = \varepsilon_{HW}(\hat{\lambda}_t + W_t) \qquad \text{(log-lin LS)}$$

↕ Frisch elasticity

Needs to be $\varepsilon_{HW} = 4$,

estimates say $\varepsilon_{HW} \leq 1$

Basu Sum 2, p. 46 Mac:

$$\gamma_t H_t^{\eta} = \lambda_t \frac{W_t}{P_t} \qquad \text{(LS)}$$

$$\Rightarrow \hat{\chi}_t + \eta \hat{H}_t = \hat{\lambda}_t + \hat{w}_t - \hat{p}_t$$

Supposing $\hat{\chi}_t = 0$,

$$\hat{H}_t = \frac{1}{\eta}\left[\hat{\lambda}_t + \hat{w}_t - \hat{p}_t\right]$$

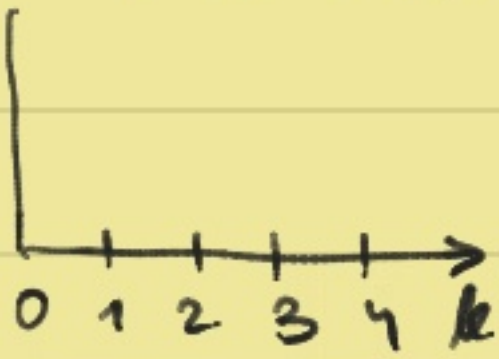
$$\varepsilon_{HW} = \frac{1}{\eta} = 4 \Rightarrow \eta = \frac{1}{4}$$

If I use the number $\omega = 1.25 = \frac{5}{4} = \eta$,

then $\varepsilon_{HW} = \frac{4}{5} < 1$ ok!

ok, so autocorrelogram:

$Cor(\pi_t, \pi_{t-k})$	$Cor(\pi_t, x_{t-k})$	$Cor(\pi_t, i_{t-k})$
$Cor(x_t, \pi_{t-k})$	$Cor(x_t, x_{t-k})$	$Cor(x_t, i_{t-k})$
$Cor(i_t, \pi_{t-k})$	$Cor(i_t, x_{t-k})$	$Cor(i_t, i_{t-k})$



Let's try to extract these from $\Omega = 45 \times 1$

$\Omega = vec(\Gamma)$ where $\Gamma = 3 \times 3 \times (K+1)$

For this purpose, create $A = 3 \times 3 \times 2$

with $A(:,:,1) = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$ $A(:,:,2) = \begin{bmatrix} 10 & 13 & 16 \\ 11 & 14 & 17 \\ 12 & 15 & 18 \end{bmatrix}$

Then vec(A) produces = $vec(A) = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \\ 8 \\ 9 \\ 10 \\ 11 \\ \vdots \end{bmatrix}$

This means that reshape(vec(A), 3,3,2) should give you back A: yep!

⇒ So then I can get back Γ = reshape(Ω, 3,3,K-1)

The autocovariogram is beautiful. The problem is that I'd need to regenerate the truths if I change calibrated params, and the real data estimation doesn't look great either in response to params change. But resnorm is smaller, so maybe those new params are good?

Now creating truth using the new param values ⇒ it's not able to solve. What about w/ convexity moment? Not quite. We still seem to have a FEDI problem, potentially an underID problem too.

4 July 2020

Ok so let's be systematic.

Truth: nfe = 6, fe ∈ [-3.5, 3.5], convex.

I started out w/ est. 6 α's, on fe∈[-3.5, 3.5] w/o additional moments.

Then w/ the convex moment. They don't find a sol and aren't convex either.

Wconvex = 0, 1000, 10000 and 100000 doesn't work.

I'm wondering if I can impose this moment differently somehow, in a more legendre fashion.

Try Wconvex = 1000 (defaults but Wmean = 0)

	flag	resnorm	convex?
default	0	1046	no
nfe=5	3	1082	no
fe∈[-2.5, 2.5]	2	1037	no
fe∈[-1.5, 1.5]	2	828	No!
fe∈[-1, 1]	2	887	No
fe∈[-0.5, 0.5]	3	1096	No

	flag	reconm	convex?
$fc \in [-0.1, 0.1]$	0	10 98	No
$nfc=5$, $fc \in [-0.1, 0.1]$	2	1111	No!

Back to default, w/ Wmean = 0.

→ Try to fix Wconvex.

Instead of summing and adding onto previous moments, I now add a separate moment so res is (45 + # of additional moments) x 1.

But it still doesn't seem to be strong enough to impose convexity!

Cranking Wconvex = 10e+10 is now enough to do it, but barely moves from the initial. I need to change it further.

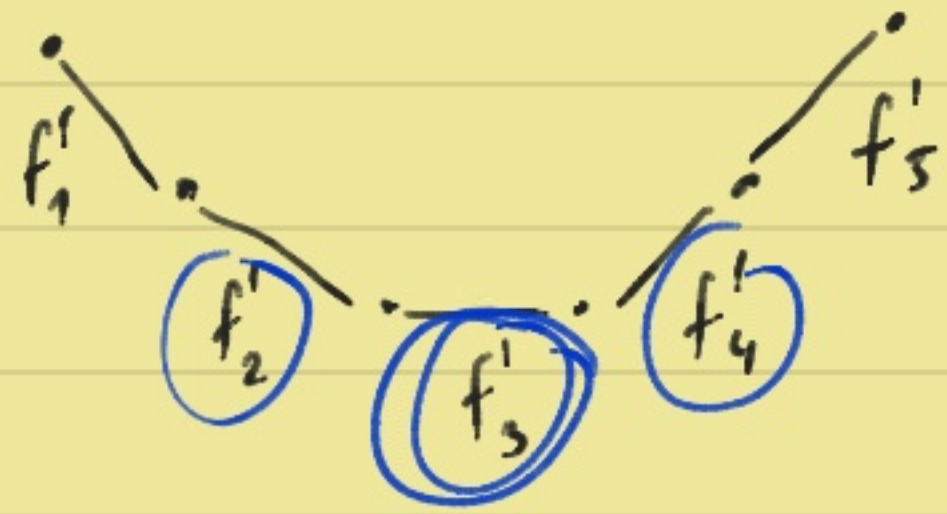
Maybe I need to add a moment that says treat the first set of 1st derivs should

5 July 2020

be <0, while the 2nd set >0.
The problem is that a shape like

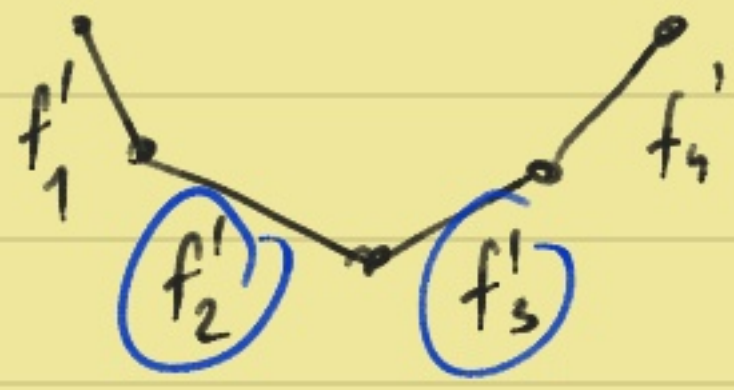
is currently only a problem here b/c that's the only point where the 2nd derivative is <0.
But if I also tell it that the first few 1st derivatives ≤0 and the last few ≥0 then I can restrict the shape to be at least weakly convex.
If nfe=6, then there are nfe-1 first derivs and nfe-2 second derivs.

α



$(nfc=6)$

α



$(nfc=5)$

I feel that my priors are less strong on what should happen in the middle — in particular, I don't know when the first derivative f' changes sign. So maybe the least restrictive thing to do is to not restrict the middle f's. Instead, just restrict $f_1' \leq 0$ and $f_{end}' \geq 0$.

That doesn't seem to be enough. So let's do the following: $\frac{(nfc-1)}{2}$ is the first half of the first derivatives. If $\frac{nfc-1}{2}$ is integer,

then take those. Else take floor($\frac{\text{orig}-1}{2}$).

Wait - right now $\text{obj}_{\text{chi}}(\alpha^{\text{true}}) \neq 0$!
For rng(0) - generated shocks, $\text{obj}(\alpha^{\text{true}}) = 0$.

Ok, so rng(3) generated shocks can actually do a pretty good job, w/o additional moments.
ronom = 865. flag = 0 though :(

Doing a couple of diff rng(·)'s seems to suggest that the additional moments aren't changing the sol. a lot → they're just making it a little pretties, but ultimately, it's the same.
→ it depends more on the (rng(·) of the) shocks than on the moments. That's bad news and sounds like lack of ID.
→ You know what? Maybe each derivative

should be a moment!

So need to be very smart. 6 July 2020

See the problem might not be that we're under-ID
b/c if that was so, we should be able to ID
smaller nfc perfectly.

The estimated $\hat{\alpha}$ don't seem robust to nfc
- but that doesn't really mean anything b/c
if I initially estimate nfc = 6

then for nfc = 2 it's not easy to say which fits it
better:

Even wfc > 6 (the results for sim_data) converge to 8th, although I don't think it's robust, but so isn't the lower wfc estimates.

→ So I kind of wonder if the problem is not as much a GMM-problem (a moment problem) as I thought? Could it be that it's more a FEM- or a grain-value problem? E.g. I see that the grain in the simulation using α^{true} is always < 0.05, while the α^{true}-values exceed 0.05 3 times! I'm also concerned about the FE-values which really depend on the specific shades. Also wondering if $\bar{q}$, the initial value of the grain matters? (Btw, should begin calling it h_0?)

So, looking at the grain k^{-1} and k implied in the sim by $\hat{\alpha}$ (the one that looks like /_)

I see that k^{-1} is bunching around 0.07 with little variance; f_e is mainly btwn $[-0.5, 0.5]$. So my feeling is that GAN is choosing, in the f_e-support that's available, the grid that fits the one that it observes, more or less. That's why the $\hat{\alpha}$ look the opposite of the truth:

while the truth is

$\Rightarrow$ try to create a truth that

i) is defined on a smaller f_e-support so that this support can actually be visited by most sims

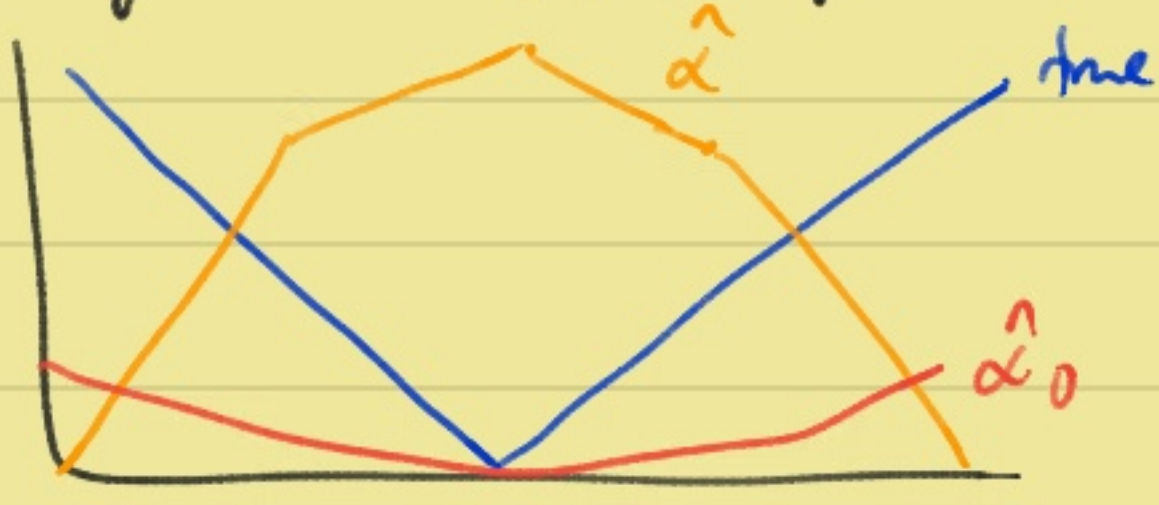
ii) has $\max(k^{-1})$ values that are actually generated by sims using α^{true}

1) $n/k = 6$, α^{true} unchanged, $f_e^{true} \in [-2, 2]$

Well, that initially doesn't work b/c doing the

estim on anything smaller than $fc \in [-3, 3.5]$ yields errors → but it doesn't work even for $mg(3)$ where the shoulders allow me to go do $fc \in [-2, 2]$.
Same w/ $mg(2)$; it doesn't converge at all!

Let's try another bump that's convex, but flatter: and pushes from 0 in the middle (which I suppose is the more frequently visited range)
In particular, let wfc be odd, so 0 is a knot.
$wfc = 5$ $fc \in [-2, 2] \rightarrow -2, -1, 0, 1, 2$
Extremely interesting, you get:



Now, if you mirror $\hat{\alpha}$ (multiply by -1 and all

the lowest value to all to push it up) you nearly get the truth.

Ok listen here, what I try for this scenario is plug α^{true}, $\hat{\alpha}^{inverse}$ and $\hat{\alpha}$ into obj.

$obj(\hat{\alpha}) = 710$

$obj(\hat{\alpha}^{inv}) = 1652$

$obj(\alpha^{true}) = 1514$

And even if I do $mg(0)$, $obj(\alpha^{true}) > obj(\hat{\alpha})$ what?! I thought I'd checked that and that it was ok?! → Ok, it was b/c I wasn't considering the right fe-space, but now it's good!

So I need to think about the role 7 July 2020
of random shocks. Maybe also IRFs. The nice thing about IRFs is that they don't depend on shocks.

Peter meeting — 7 July 2020

Boston Fed
Estim

"retirement from teaching duties is to be full at a crucial point in my research : it would make a huge difference to me"

→ generate 100 samples of artificial data and in each case generate the moments and then avg those over the 100 sims
→ and match avg of moments
⇒ LLN to wash out the randomness

SMM (sim method of moments) invented by Tony Smith, 1993
→ see the paper "indirect inference"
journal of applied econometrics

If I had a stationary model, one long ts would also

be good to wash out the contamination of shocks.

In a sense, I was previously controlling for sampling error by reg(o)-by everything
→ it's telling me that sampling error is so large that it can't handle.

Put in a footnote: "the consistency properties of the est. I'm using are presented in "xyz.

Work after

H looks like Smith (1993) simulates N=1000 samples, estimates the params on each and then looks at the mean.

Duffie & Singleton 1990: don't mention a cross-section, just that the moments themselves are time-averages

Lee & Ingram (1991): same (0.3, Mac) ⇒ shows cons. & asymptotic N.

They all seem to agree that the sim data should be longer than the true data.

Fuck knows! I have no choice but to try.
But: first, mend the fk_CEMP criterion using smat.
It works! Verified in clean_up_sim_learn.m
and in command_IRFs_approx_pretty.m.
I've also "re-implemented" the scalar CUSUM and interestingly, it looks a lot like the CEMP-crit in terms of IRFs.

The "scalar CEMP" of fk_cemp_scalar.m implements a criterion where $\theta_1 = \max(\text{diffs btwn } E(ALM) - PLM \text{ for inflation only})$. This is in analogy to the CUSUM-scalar that allows for fk coming from wherever to show up and

drive up θ_+; also here I allow diffs not just in the constant a, but also the slope b to show up. This may not actually have an impact as long as I consider a "copo" (constant-only-pi-only) PLM model.

→ Yep: in `command_IRFs_approx_pretty.m` the LCMP-thing (scalar or vector) looks the same as the scalar CUSUM. Makes sense b/c

1) there's no distinction btwn scalar & vector LCMP in a copo-PLM model
2) there still is btwn scalar & vector CUSUM b/c firms still make fcst errors in x & i as well.

Let's use the vector versions as defaults for COMP & CUSUM cmd.

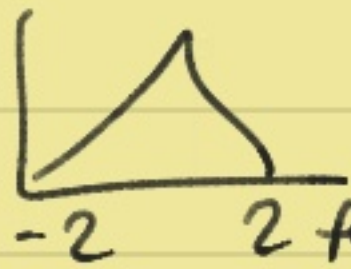
What I'm now looking at is the IRF generated by α^{true} (06 July): it looks qualitatively and quantitatively like the one generated by CONP, but the gain looks like a cgain, fluctuating around a small value.

Another thing I see is that the smooth-crd, but also CONP (not cusum) look like dgain learning.
↳ but it can move to look like cgain by manipulating α and fe-support.

CONT HERE and w/ the cross-section!

Ok, so cross-section. 8 July 2020
What I need to answer is whether to do the estim N times, or whether to average the data. It's the former.

Problem: so I do 10 different histories ($N=10$) and the mean looks like



and these non-convex ones dominate the α-candidates that are closer to the truth! So in a way it's of course the same problem I've been observing. The reason it's worrying is b/c it suggests that taking a big cross-section may not be able to wash out the effects of shocks b/c wrong sols have a lower residual than correct ones.

The problem is seen again at $N=100$. (rng(1))
Here's the best $\hat{\alpha}$ against α^{true}

But this is partly encouraging: it seems to suggest that we have a problem I've thought of before, which is that fe's out in the "edges" don't occur frequently, and so the data "pushes" that up which occurs: the middle. If this is so, restricting fe-support or adding the convexity-moments might help. Let's try it out in $N=20$.

But if it's the α^{true} has a very high residual at some draws for the shocks, that only leaves me w/ one option: draw as many shocks as possible and hope to get one that's close. The problem w/ that is that it's seeing that the LLN fails.

- $fe \in (-1,1)$
  Nope, that didn't help much.

- Ah I forgot - I should increase T for the sim.
  $T = 2 \cdot T^{data}$
  (Honestly I don't hope for much here - I mean, the "true data" is T^{data} long, having more T won't help in matching moments.)

→ Does have helped ... a bit ...?
  It made $\hat{\alpha}$ "flatter", less concave.

- Wconvex: $Wdiffs1 = Wdiffs2 = 100$
  → didn't help at all!
- $Wdiffs1 = Wdiffs2 = 1000$
  → makes $\hat{\alpha}$ flatter, but still concave.
- $Wdiffs1 = 0$, $Wdiffs2 = 1000$
  → Nope!

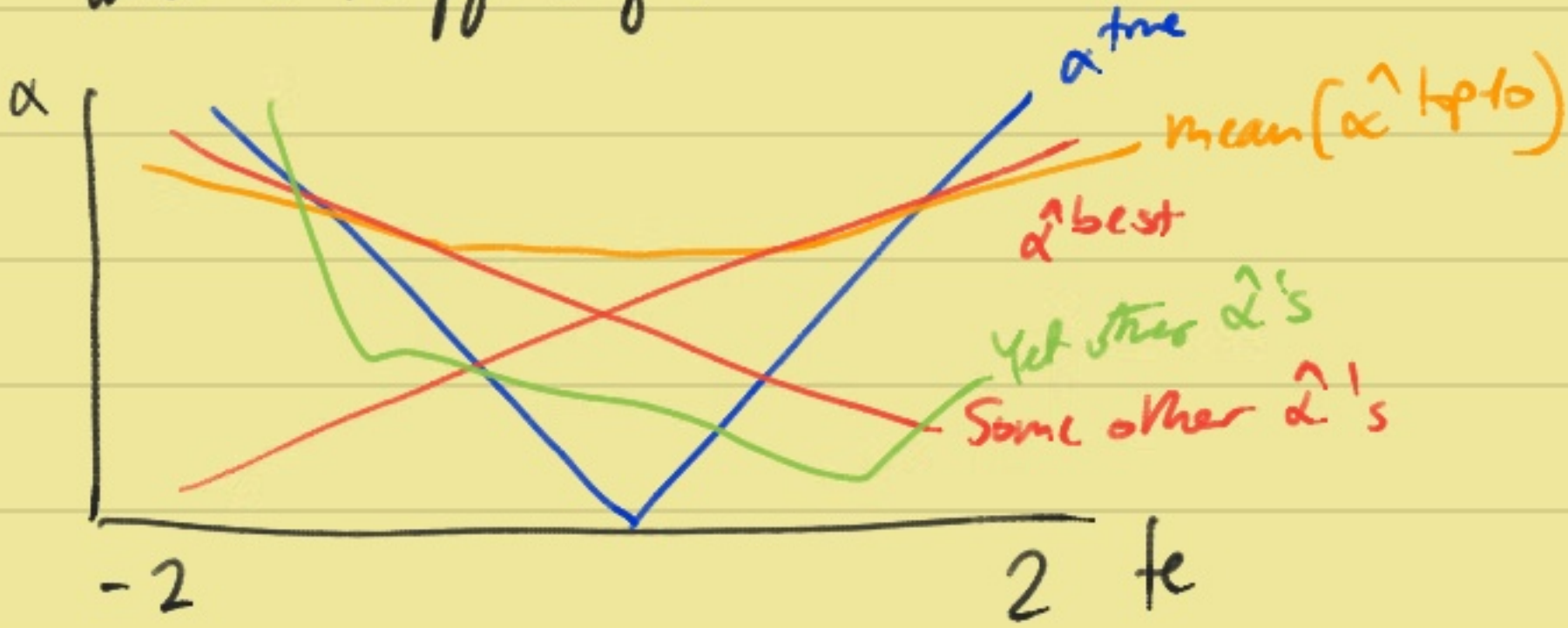
- Wdiffs1 = 0, Wdiffs2 = 1000, $T = 2 \cdot T^{dah}$
  → again, this flattens $\hat{\alpha}$

The problem is also that when imposing the convexity moment, it really has to work against its instincts: it fares worse w/ the ACF-moments so it has to trade off fitting those and fitting the convexity moment

- Wdiffs1 = 0, Wdiffs2 = 10 000
  → ok, almost convex, but again ACF-moments deteriorate
- Wdiffs1 = 0, Wdiffs2 = 100 000
  → actually looks a lot better, but again ACF-moments deteriorate further
- Wdiffs1 = 0, Wdiffs2 = 1000 000
  Again more convex, but ACF still close to initial.

- Wdiffs2 = 10 000 000

  Ok, on average this is convex. But I see what is happening:



This is a disaster. It's not the case that there are some local min the solver gets stuck on. It's really that the ACF-moments are best matched w/ a concave $\hat{\alpha}$.

Actually, now that I'm comparing the ACFs, they're not that much worse for the convexity moments.

What my analysis of IRFs(α) shows is that the same α on a diff fe-space can yield diff dynamics. This is why diff shock sizes give such different estimates, even for the same fe-support b/c they yield different parts of fe-space visited.

↳ In the data, I need to have a sense of the fe-space.

- Try Wdiff52 = 100'000'000 (rng(2)) (N=20 still)

  Looks quite good, still not great but ok.

  (Only 5 converged though!)

The bigger Wdiff52, the less convergence we have.

- Wdiff52 = 100'000'000 (rng(1)) (N=20 still)

I guess better but only 3 converged now, so for N=20, this is the most I can crank up Wdiff52. But this does make me wonder whether

an avg est w/ N=1000 would work...
→ need to get the server working!

Let's try wdiffs2 = 0 again and set Wprior = 1000.
→ doesn't make diff compared to Wprior = 0.
Let's try Wprior = 100 000.
→ no effect.
⇒ no evidence of the FEM-problem.

→ Now I completed a smart addpath that understands when it is: local, or server and gets your paths added & cd'ed right.
the only problem is that an older version of Matlab (2017a) runs on the server, which has the drawback that some things are diff
- diff error messages for lsqnonlin
- mink does not exist yet (introduced in 2017b)

• potentially even the rand seed may be diff

Ok so I checked and randi w/ rng(0) 9 July 2020 gives the same, so the seed at least seems the same.

New idea: what if I simply "unzip" an add-on for all the functions missing in 2017a?
⇒ addons 2017a folder both on local and on server, and the first file to load it is mine. in which does what Mathab's unzip does, except for vectors only. It works!

A problem: even w/o there being seed differences, the results are different:

	local	server
rSnormpn	968	2134

	local	server
mean($\hat{\alpha}$)	0.03	0.0419
	0.0496	0.0341
	0.0456	0.092
	0.0228	0.025
	0.0372	0.0497

B/c in lsqnonlin, the dim mismatch error occurs much more frequently than on local.

If I select on local only the rshooms that converged on the server, and I take the top10 of that, and mean of that, I get rshoom = 2129, close to what I get on the server. So the hope would be that the two converge as $N \to \infty$.

For N=100 they do seem to converge to each other, but they don't find all the same optima.

There's also a sense in which if I don't first take the mean of the top10, I might converge to the truth faster.

Peter meeting — 9 July 2020

- sample length of λT (length of real data) (Fernandez et al 2016 P. 155 Mac)
- fix seed (p. 156 in Mac) so objective is smooth
  → same ϵ for every evaluation of objective given α

$m(\text{data})$ vs $m(\beta)$

↑ moments ↑ parameters

↑ moments on simulated data given β

- Main contrib of Smith: jitter macro to sim that had been used prev. to estimate by micro

- $\uparrow T$
  the bigger the sample of artificial data, the better the estim
  In a stationary environment, $\uparrow J$ vs $\uparrow T$ doesn't matter, in my model it does b/c the ts generated by the model isn't stationary
  ↳ should move it as a footnote in the paper so that it's SMM-style but can't invoke the theorem in Smith's paper b/c the DGP isn't stationary, where N_1 (not T) $\to \infty$

- Yellow line from C in obj of a
  → if loss is smaller than $\hat{\alpha}$ in a)
  then the solver is just stuck at a local min

vs. the fe-story would suggest an ID issue.

Need an App that describes step-by-step exactly how I did the claim.

Work after

Ha! $obj\left(\hat{\alpha}^{Wdiff2=10M}\right)^{Wdiff2=10M} = obj\left(\hat{\alpha}^{Wdiff2=10M}\right)^{Wdiff2=0}$

→ The $\hat{\alpha}$ involves the same loss whether Wdiff2 is turned on or off: most likely b/c this set is convex and so the convexity moment is not adding a penalty.

For me that seems like it's an ID issue!

Let's now do N=1000 both locally and on the server. Locally it should take more than 5 hours, on the server it should take more than 10 hrs.

Started local 4:30 → ETA: 9:30 pm

Server began exec 5:06 pm → ETA: ~3:00 am.

→ Took 4.76 hours

→ Took 3:36 hrs:min

And they do converge, both towards the truth and towards each other!

This was N = 1000.

Let's try N=2000, server only.

Expect it to take 7 hrs, so if start at 12 am, should be ready at around 7 am.

→ It took almost 8 hrs. 10 July 2022

It looks like it's converging very slowly.

Need to try $N = 10\,000$, or even $N = 100\,000$.

↳ 5.8 hrs = 40 hrs, almost 2 days

$N = 100\,000 \rightarrow 400$ hrs $\approx 16.\overline{66}$ days

$N = 50\,000 \rightarrow 200$ hrs ≈ 8.3 days

↳ $N = 10\,000$ on server

↳ started at: 9:56 am 10 July 2020

ETA: 2:00 am 12 July 2020 (Sunday)

finished at: 12:55 am Sunday, 12 July

Comment: It hasn't changed much vis-à-vis $N = 1000$, or $N = 2000$. Good & bad: $N = 1000$ seems to converge, bad: it's not consistent. Need more moments or is $f_R \in (1,1)$ just not occurring much?

Let's try to do 2 things locally:

1) speed up code

2) try estimation w/ real data

+1) should also see if estim w/o convexity moved converges to truth or not.

#1) Local: Wdiffs2 = 0, N = 100

Doesn't look like converging here.

Local: Wdiffs2 = 0, N = 1000 (2 hrs)

ETA: 12:10 pm

Nope, no sign of converging.

2) Real data, Wdiffs2 = 10M, N = 1000.

ETA: 3:30 pm.

Instead of 2 hrs, it took almost 5, but it gives a nice convex shape.

Makes me wonder if

1) we can reduce Wdiffs2

2) we can estimate more parameters (nfe = 6 instead of 4)

1) Speed up code: go back to sim data 11 July 2020

N = 10. Should take 3 min.

Take that as a benchmark. 316 sec (5.28 min)

- Try to swap parallel within the lsqnonlin to parfor outside: 123.9 sec! Huge improvement!
- Take out reference to Wdiffs1

took 129.78 sec, no diffs there.

So I put 'em back in.

Ok. So this means we can run N=700 for sim data.

Previously, it should take 3160 sec = 52 min

Now, it should take 1239 sec = 20 min

↳ it took 1184 sec.

N=1000 previously took 5 hrs = 18000 sec

now it should take 12390 sec = 3.44 hrs.

Improvements on the server should be bigger b/c more cores.

So run N=100 for sim data. → 1184 sec.

Run N=700, Wdiffs2=100 000. ⇒ get same on avg!

Run N=100, Wdiff2 = 1000.

Can it be: elapsed time was only 367 sec!?

→ Not sufficient for convergence!

Run N=100, Wdiff2 = 10 000

→ also not enough. So Wdiff2 = 100 000

is the main weight I need to use.

So let's turn then to real data,

Real data, N=100, Wdiff2 = 100 000.

→ I get nearly the same as w/ N=700,

Wdiff2 = 10M !

and it took 488 sec !

⇒ N=1000 can then take 4880 ≃ 5000 sec

= 83 min only! Wow!

Let's do it: N = 1000, Wdiff2 = 100 000, real data:

It took 4825 sec.

Almost the same, a little more symmetric.

Should do it w/ i) more N, ii) more nfe
iii) on a bigger fe-support.
iv) Think about why for sim data, the $f_E \in (1,1)$ region is not well ID-ed.

I think that my conclusion so far is that a) LLN holds more or less in that estimates do converge to a true

13 July 2020

b) LLN partly fails b/c the $f_E \in (-1,1)$ range doesn't seem to converge

c) asymptotics kicks in already at N = 1000 afterwards there are only small refinements

→ Today's task then is to see whether we can circumvent problem (b) by choosing the nfe = 6 further.

$m_{fc} = 6$, $f_c \in (-3.5, 3.5)$ dm

$N = 10$ 112 sec

$N = 100$ 370 sec

$N = 1000$ 3590 sec = 59 min. Does conv but a) I'm not sure if the asymptotics has kicked in yet b) this $f_c \in (-2, 2)$ region feels a little un-1D-ed.

$N = 10\,000$ should take 35 900 sec = 10 hrs. The server is down, so wait w/ it, but speed gains there should be bigger.

$N = 100\,000$ should then take $\approx$ 100 hrs = 4 days.

$N = 10\,000$ ETA: 22:30

Took 35818 sec.

Peter meeting

- Good: $\uparrow N$ further doesn't change the result
- Impose the requirement that for $f_c = 0$, $k_1^{-1} = 0$.

- some mistake in code?
  ⇒ deliberately choose the data (i.e. non-randomly) w/ f_e close to zero.
  ⇒ Run est. procedure and see if it's able

- while f_e are in memory, store them later on for $N = 10\,000$ histories. → Is there anything unexpected going on?

- discrepancy btwn blue & yellow is only a problem at $f_e = 0$.

- You have to have a knot at zero
  → just think of approx abs value fcn w/ a pw linear

- Finer grid closer to the origin?

- A possibility about autocov being closer for small N.

- fix α
- sim model
- calc autocor
- Change α's in various ways, all together or one at a time ⇒ compare autocovariances!
  ⇒ should see: are there moments that are sensitive, and moments that aren't?
  ⇒ the info in the latter won't gonna help you in the estim.

Priority: finer grid; knock at zero, impose "0 at 0"

- Try: fix all but 1 α at the true and see if you can find the true α_i^{true} for one i.
  ⇒ can plot loss for $N \to \infty$
  should approx the global min
  ⇒ demonstration of the consistency of the estimator.

Note that w/ 7 breakpoints, N=200 15 July 2020
takes 626 sec instead of 370 ;;

Right now I'm trying to look into the matrix dim error that sometimes occurs in lsqnonlin.
↳ I can't, it's too internal : C

After that, try to impose $\alpha(\text{fc}=0) = 0$
What I don't know is if I should just set the middle $\alpha = 0$, or if I should simply penalize deviations as $\hat{\alpha}_{middle}^2$?
The latter is easier to implement but a little less strong. But let's do that

$W_{mid} \cdot \hat{\alpha}_{mid}$ ~~^{2}~~

Let's actually do $\hat{\alpha}_{mid}$ b/c since $\hat{\alpha} < 1$, ^{2} just decreases the penalty.
$W_{mid} = 100$ ($w_{fc}=5$). Nothing.

Wmid = 1k → works well.

Wmid = 10 K. → Works perfectly

! Wait a sec: about the convexity moment:
I'm currently imposing 2nd diffs, not 2nd derivs.

$$\frac{\partial y}{\partial x} = y' = \frac{\Delta y}{\Delta x} = \frac{\Delta \alpha}{\Delta \text{fegrid}}$$

$$\frac{\partial^2 y}{\partial x^2} = \frac{\partial y'}{\partial x} = y'' = \frac{\Delta y'}{\Delta x} = \frac{\Delta \alpha'}{\Delta \text{fegrid}}$$

⇒ Redo: Wmid = 1k, new convex moment
Toole 1942 sec (≃ 19 min)! Na király, csak
feldobníztam az időt! ≃
It's the uneven + new convexity that does it.
But now the fit is pretty damn good!
And it also had no error messages!

Need to redo things w/ new convexity moment!
I expect that you no longer need 100K as a
lowright.

# gridpoints	spacing	Wmid	Wdiffs2	
nfe = 5	uniform	0	1000	→ not convex
nfe = 5	uniform	0	10 000	→ not convex
nfe = 5	uniform	0	100 000	→ convex :)
nfe = 5	uneven	0	100k	✓
nfe = 5	uneven	1000	100k	✓

nfe = 5 uneven 1000 1000 → not convex
but 339 sec → so it mainly is the
convexity weight that makes it slower,
and it does so b/c much fewer converge
then → decreasing MaxFunEvals
to 700!

# gridpoints	spacing	Wmid	Wdiffs2	
nfe = 5	uneven	1000	10000	→ ✓, 560 sec.
nfe = 7	uneven	1000	10k	→ ✓, 695 sec

Data

nfe = 5 uneven 1000 10K → ✓, 711 sec

but model explodes at $\hat{a}$!!

Back to synthetic data:

If N=100 is 560 sec, then N=1000 ⇒ 5600 sec,

93 min

and N=2000 → 11200 sec ⇒ 187 min.

What I just realized is that my data autocovariograms should also be averaged: for each $\hat{\alpha}$, should compute Ω and then take an avg.

→ Now doing that for real data, & need to redo others!

Tomorrow: redo the odd & even no. of gridpoints using the new convex moment, but on evenly spaced grid and w/ Wmid = 0.

My concern now is that if I impose convexity strongly, and I impose the 0 at 0 condition too, then I get a V-shape, not b/c that's the truth, but b/c I'm forcing it to look that way.

16 July 2020

9 knots: 20 min, fits well, many don't converge

Saved N=1000, real data as estim_Lomgam_outputs_univariate_16_Jul_2020_15_25_10.

Tomorrow: do PEA & UFI w/ this!