Using the Sequence-Space Jacobian to Solve and Estimate Heterogeneous-Agent Models

Adrien Auclert, Bence Bardóczy, Matt Rognlie, Ludwig Straub

July 2019

NBER Summer Institute

This paper

Q: How should we solve heterogeneous-agent (HA) general equilibrium models with aggregate shocks in discrete time?

- Reiter: linearize aggregates, solve **linear state space system**
- Boppart-Krusell-Mitman: then certainty equivalence in aggregates, can get same answer with MIT shocks in **sequence space**

This paper

Q: How should we solve heterogeneous-agent (HA) general equilibrium models with aggregate shocks in discrete time?

- Reiter: linearize aggregates, solve **linear state space system**
- Boppart-Krusell-Mitman: then certainty equivalence in aggregates, can get same answer with MIT shocks in **sequence space**

This paper: write equilibrium as **linear system** in **sequence space**

This paper

Q: How should we solve heterogeneous-agent (HA) general equilibrium models with aggregate shocks in discrete time?

- Reiter: linearize aggregates, solve **linear state space system**
- Boppart-Krusell-Mitman: then certainty equivalence in aggregates, can get same answer with MIT shocks in **sequence space**

This paper: write equilibrium as **linear system** in **sequence space**

1. **Fast**: can solve and estimate HANK models with large state spaces in seconds on laptop
2. **General**: applies to broad class of HA models
3. **Accessible**: key steps automated in publicly available code

This paper

Q: How should we solve heterogeneous-agent (HA) general equilibrium models with aggregate shocks in discrete time?

- Reiter: linearize aggregates, solve **linear state space system**
- Boppart-Krusell-Mitman: then certainty equivalence in aggregates, can get same answer with MIT shocks in **sequence space**

This paper: write equilibrium as **linear system** in **sequence space**

1. **Fast**: can solve and estimate HANK models with large state spaces in seconds on laptop
2. **General**: applies to broad class of HA models
3. **Accessible**: key steps automated in publicly available code

How: new ways to efficiently compute **sequence-space Jacobians**

Key idea: the sequence-space Jacobian is all you need

- Start from model in nonlinear sequence space: for all t

$$H_t(\{\mathbf{U}_s\}, \{\mathbf{Z}_s\}) = 0$$

Key idea: the sequence-space Jacobian is all you need

- Start from model in nonlinear sequence space: for all t

$$H(\mathbf{U}, \mathbf{Z}) = 0$$

$\mathbf{U} \in \mathbb{R}^{n_u T}$ paths of endog vars, $\mathbf{Z} \in \mathbb{R}^{n_z T}$ paths of exog vars

Key idea: the sequence-space Jacobian is all you need

- Start from model in nonlinear sequence space: for all t

$$H(\mathbf{U}, \mathbf{Z}) = 0$$

$\mathbf{U} \in \mathbb{R}^{n_u T}$ paths of endog vars, $\mathbf{Z} \in \mathbb{R}^{n_z T}$ paths of exog vars

- **Linearized dynamics** around steady state

$$d\mathbf{U} = \underbrace{-\mathbf{H}_U^{-1}\mathbf{H}_Z}_{\text{Jacobians}} d\mathbf{Z}$$

Key idea: the sequence-space Jacobian is all you need

- Start from model in nonlinear sequence space: for all t

$$H(\mathbf{U}, \mathbf{Z}) = 0$$

$\mathbf{U} \in \mathbb{R}^{n_u T}$ paths of endog vars, $\mathbf{Z} \in \mathbb{R}^{n_z T}$ paths of exog vars

- **Linearized dynamics** around steady state

$$d\mathbf{U} = \underbrace{-\mathbf{H}_U^{-1}\mathbf{H}_Z}_{\text{Jacobians}} d\mathbf{Z}$$

- **Estimation**: impulse responses = MA(∞) representation

$\rightarrow$ 2nd moments $\rightarrow$ likelihood

- **Local determinacy**: test singularity of $\mathbf{H}_U$
- **Nonlinear MIT shocks**: Newton's method with $\mathbf{H}_U$

Example of computing times for:	Krusell-Smith	two-asset HANK
Jacobians	0.10 s	5.7 s
One impulse response	0.0012 s	0.120 s
All impulse responses	0.0068 s	0.400 s
Bayesian estimation		
single likelihood evaluation	0.00088 s	0.18 s
finding posterior mode	0.12 s	570 s
Determinacy test	252 μs	631 μs
Nonlinear impulse response	0.18 s	27 s
No. of idiosyncratic states	3,500	10,500
Time horizon (T)	300	300
No. of shock parameters in estimation	3	14
No. of model parameters in estimation	0	5

For concreteness: Krusell-Smith model in sequence space

Perfect foresight: only aggregate state is time $t = 0, 1, \ldots$

- Households: ability e, capital k, inelastic labor supply $\equiv 1$

$$V_t(e, k_-) = \max_{c,k} u(c) + \beta \sum_{e'} \mathcal{P}(e, e') \cdot V_{t+1}(e', k)$$

$$\text{s.t. } c + k = (1 + r_t)k_- + w_t el$$

$$k \geq 0$$

For concreteness: Krusell-Smith model in sequence space

Perfect foresight: only aggregate state is time $t = 0, 1, \ldots$

- Households: ability e, capital k, inelastic labor supply $\equiv 1$

$$
\begin{aligned}
V_t(e, k_-) = &\max_{c,k} u(c) + \beta \sum_{e'} \mathcal{P}(e, e') \cdot V_{t+1}(e', k) \\
\text{s.t. } & c + k = (1 + r_t)k_- + w_t el \\
& k \geq 0
\end{aligned}
$$

$\Rightarrow$ policy $k_t^*(e, k_-)$

For concreteness: Krusell-Smith model in sequence space

Perfect foresight: only aggregate state is time $t = 0, 1, \ldots$

- Households: ability e, capital k, inelastic labor supply $\equiv 1$

$$V_t(e, k_-) = \max_{c,k} u(c) + \beta \sum_{e'} \mathcal{P}(e, e') \cdot V_{t+1}(e', k)$$

$$\text{s.t. } c + k = (1 + r_t)k_- + w_t el$$

$$k \geq 0$$

$\Rightarrow$ policy $k_t^*(e, k_-)$
$\Rightarrow$ distribution $D_t(e, K_-) \equiv \Pr(e_t = e, k_{t-1} \in K_-)$, assuming $D_0 = D_{ss}$

For concreteness: Krusell-Smith model in sequence space

Perfect foresight: only aggregate state is time $t = 0, 1, \ldots$

- Households: ability e, capital k, inelastic labor supply $\equiv 1$

$$V_t(e, k_-) = \max_{c,k} u(c) + \beta \sum_{e'} \mathcal{P}(e, e') \cdot V_{t+1}(e', k)$$

$$\text{s.t. } c + k = (1 + r_t)k_- + w_t el$$

$$k \geq 0$$

$\Rightarrow$ policy $k_t^*(e, k_-)$

$\Rightarrow$ distribution $D_t(e, K_-) \equiv \Pr(e_t = e, k_{t-1} \in K_-)$, assuming $D_0 = D_{ss}$

$\Rightarrow$ summarize by **capital function**

$$\mathcal{K}_t(\{r_s, w_s\}_{s \geq 0}) = \int k_t(e, k_-) D_t(e, dk_-)$$

Krusell-Smith model: reduce to $H(\mathbf{K}, \mathbf{Z})$

- Competitive firms, Cobb-Douglas production:
  - $r_t = \alpha Z_t (K_{t-1}/L_t)^{\alpha-1} - \delta$
  - $w_t = (1-\alpha) Z_t (K_{t-1}/L_t)^{\alpha}$
- Market clearing (goods redundant):
  - $L_t = 1$
  - $K_t = \mathcal{K}_t(\{r_s, w_s\})$

Krusell-Smith model: reduce to $H(\mathbf{K}, \mathbf{Z})$

- Competitive firms, Cobb-Douglas production:
  - $r_t = \alpha Z_t (K_{t-1}/L_t)^{\alpha-1} - \delta$
  - $w_t = (1-\alpha) Z_t (K_{t-1}/L_t)^{\alpha}$
- Market clearing (goods redundant):
  - $L_t = 1$
  - $K_t = \mathcal{K}_t(\{r_s, w_s\})$
- Write

$$K_t = \mathcal{K}_t(\{r_s, w_s\})$$

Krusell-Smith model: reduce to $H(\mathbf{K}, \mathbf{Z})$

- Competitive firms, Cobb-Douglas production:
  - $r_t = \alpha Z_t (K_{t-1}/L_t)^{\alpha-1} - \delta$
  - $w_t = (1-\alpha) Z_t (K_{t-1}/L_t)^{\alpha}$
- Market clearing (goods redundant):
  - $L_t = 1$
  - $K_t = \mathcal{K}_t(\{r_s, w_s\})$
- Write

$$K_t = \mathcal{K}_t(\{\alpha Z_s (K_{s-1}/L_s)^{\alpha-1} - \delta, (1-\alpha) Z_s (K_{s-1}/L_s)^{\alpha}\})$$

Krusell-Smith model: reduce to $H(\mathbf{K}, \mathbf{Z})$

- Competitive firms, Cobb-Douglas production:
  - $r_t = \alpha Z_t (K_{t-1}/L_t)^{\alpha-1} - \delta$
  - $w_t = (1-\alpha) Z_t (K_{t-1}/L_t)^{\alpha}$
- Market clearing (goods redundant):
  - $L_t = 1$
  - $K_t = \mathcal{K}_t(\{r_s, w_s\})$
- Write

$$K_t = \mathcal{K}_t(\{\alpha Z_s K_{s-1}^{\alpha-1} - \delta, (1-\alpha) Z_s K_{s-1}^{\alpha}\})$$

Krusell-Smith model: reduce to $H(\mathbf{K}, \mathbf{Z})$

- Competitive firms, Cobb-Douglas production:
  - $r_t = \alpha Z_t (K_{t-1}/L_t)^{\alpha-1} - \delta$
  - $w_t = (1-\alpha) Z_t (K_{t-1}/L_t)^{\alpha}$
- Market clearing (goods redundant):
  - $L_t = 1$
  - $K_t = \mathcal{K}_t(\{r_s, w_s\})$
- Write

$$H_t(\mathbf{K}, \mathbf{Z}) \equiv \mathcal{K}_t(\{\alpha Z_s K_{s-1}^{\alpha-1} - \delta, (1-\alpha) Z_s K_{s-1}^{\alpha}\}) - K_t = 0$$

Krusell-Smith model: reduce to $H(\mathbf{K}, \mathbf{Z})$

- Competitive firms, Cobb-Douglas production:
  - $r_t = \alpha Z_t (K_{t-1}/L_t)^{\alpha-1} - \delta$
  - $w_t = (1-\alpha) Z_t (K_{t-1}/L_t)^{\alpha}$
- Market clearing (goods redundant):
  - $L_t = 1$
  - $K_t = \mathcal{K}_t(\{r_s, w_s\})$
- Write

$$H_t(\mathbf{K}, \mathbf{Z}) \equiv \mathcal{K}_t(\{\alpha Z_s K_{s-1}^{\alpha-1} - \delta, (1-\alpha) Z_s K_{s-1}^{\alpha}\}) - K_t = 0$$

- First-order solution is $d\mathbf{K} = -\mathbf{H}_K^{-1} \mathbf{H}_Z d\mathbf{Z}$ where e.g.

$$\left[\mathbf{H}_K\right]_{t,s} = \frac{\partial \mathcal{K}_t}{\partial r_{s+1}} \frac{\partial r_{s+1}}{\partial K_s} + \frac{\partial \mathcal{K}_t}{\partial w_{s+1}} \frac{\partial w_{s+1}}{\partial K_s} - 1_{\{s=t\}}$$

Krusell-Smith model: reduce to $H(\mathbf{K}, \mathbf{Z})$

- Competitive firms, Cobb-Douglas production:
  - $r_t = \alpha Z_t (K_{t-1}/L_t)^{\alpha-1} - \delta$
  - $w_t = (1-\alpha) Z_t (K_{t-1}/L_t)^{\alpha}$
- Market clearing (goods redundant):
  - $L_t = 1$
  - $K_t = \mathcal{K}_t(\{r_s, w_s\})$
- Write

$$H_t(\mathbf{K}, \mathbf{Z}) \equiv \mathcal{K}_t(\{\alpha Z_s K_{s-1}^{\alpha-1} - \delta, (1-\alpha) Z_s K_{s-1}^{\alpha}\}) - K_t = 0$$

- First-order solution is $d\mathbf{K} = -\mathbf{H}_K^{-1} \mathbf{H}_Z d\mathbf{Z}$ where e.g.

$$\left[\mathbf{H}_K\right]_{t,s} = \frac{\partial \mathcal{K}_t}{\partial r_{s+1}} \frac{\partial r_{s+1}}{\partial K_s} + \frac{\partial \mathcal{K}_t}{\partial w_{s+1}} \frac{\partial w_{s+1}}{\partial K_s} - 1_{\{s=t\}}$$

- Only hard parts are Jacobians $\partial \mathcal{K}/\partial r$ and $\partial \mathcal{K}/\partial w$!

Household Jacobians: sufficient statistics for equilibrium

- What we need are Jacobians of the capital function, up to some truncation horizon T

$$\mathcal{J}_{t,s}^{\mathcal{K},r} = \frac{\partial \mathcal{K}_t}{\partial r_s}, \quad \mathcal{J}_{t,s}^{\mathcal{K},w} = \frac{\partial \mathcal{K}_t}{\partial w_s}$$

Household Jacobians: sufficient statistics for equilibrium

- What we need are Jacobians of the capital function, up to some truncation horizon T

$$\mathcal{J}_{t,s}^{\mathcal{K},r} = \frac{\partial \mathcal{K}_t}{\partial r_s}, \quad \mathcal{J}_{t,s}^{\mathcal{K},w} = \frac{\partial \mathcal{K}_t}{\partial w_s}$$

- Then, can calculate $\mathbf{H}_K^{-1}\mathbf{H}_Z$ once, and get $d\mathbf{K} = -\mathbf{H}_K^{-1}\mathbf{H}_Z d\mathbf{Z}$ for any $d\mathbf{Z}$ almost instantly.

Household Jacobians: sufficient statistics for equilibrium

- What we need are Jacobians of the capital function, up to some truncation horizon T

$$\mathcal{J}^{\mathcal{K},r}_{t,s} = \frac{\partial \mathcal{K}_t}{\partial r_s}, \quad \mathcal{J}^{\mathcal{K},w}_{t,s} = \frac{\partial \mathcal{K}_t}{\partial w_s}$$

- Then, can calculate $\mathbf{H}_K^{-1}\mathbf{H}_Z$ once, and get $d\mathbf{K} = -\mathbf{H}_K^{-1}\mathbf{H}_Z d\mathbf{Z}$ for any $d\mathbf{Z}$ almost instantly.

- Jacobians $\mathcal{J}^{\mathcal{K},r}$ and $\mathcal{J}^{\mathcal{K},w}$ are **sufficient statistics** for household in sequence space equilibrium:
  - Shocks propagate through dynamics of household capital distribution, but we don't need details if we have the Jacobians
  - Column s of $\mathcal{J}^{\mathcal{K},r}$ is impulse response of capital to news shock dr_s, Jacobian gives these for all $s = 0, \ldots, T-1$
  - Rich objects, but fast new algorithm to get them

Approach combines several key innovations

- **Capture heterogeneity using GE sufficient statistics**
  [Auclert and Rognlie 2018, Auclert, Rognlie and Straub 2018, Guren, McKay, Nakamura and Steinsson 2018, Koby and Wolf 2018, Wolf 2019]
  - previously empirical or conceptual, now a computational tool

Approach combines several key innovations

- **Capture heterogeneity using GE sufficient statistics**
  [Auclert and Rognlie 2018, Auclert, Rognlie and Straub 2018, Guren, McKay, Nakamura and Steinsson 2018, Koby and Wolf 2018, Wolf 2019]
    - previously empirical or conceptual, now a computational tool

- **Write equilibrium as linear system in aggregates**
  [Reiter 2009, McKay and Reis 2016, Winberry 2018, Bayer, Luetticke, Pham-Dao and Tjaden 2019, Mongey and Williams 2017, Ahn, Kaplan, Moll, Winberry and Wolf 2018, ...]
    - size of system now independent of underlying HA, no Schur decomposition that's costly for large state space

Approach combines several key innovations

- **Capture heterogeneity using GE sufficient statistics**
  [Auclert and Rognlie 2018, Auclert, Rognlie and Straub 2018, Guren, McKay, Nakamura and Steinsson 2018, Koby and Wolf 2018, Wolf 2019]
  - previously empirical or conceptual, now a computational tool

- **Write equilibrium as linear system in aggregates**
  [Reiter 2009, McKay and Reis 2016, Winberry 2018, Bayer, Luetticke, Pham-Dao and Tjaden 2019, Mongey and Williams 2017, Ahn, Kaplan, Moll, Winberry and Wolf 2018, ...]
  - size of system now independent of underlying HA, no Schur decomposition that's costly for large state space

- **Solve for impulse responses in sequence space**
  [Guerrieri and Lorenzoni 2017, McKay, Nakamura and Steinsson 2016, Kaplan, Moll and Violante 2018, Boppart, Krusell and Mitman 2018, ...]
  - but now compute all in one go, no slowly-converging iteration

Common questions about our approach

Q: Are we somehow ignoring the distribution of agents?

- No, its effect on aggregates is incorporated in Jacobian

Q: When does this approach work vs. Reiter?

- Whenever agents don't care about the distribution per se, they care about aggregates that depend on the distribution

Q: Are we making some approximation (other than linearization)?

- Only truncation; exact on full discretized state space

Q: Can we deal with nonlinearities?

- Only nonlinear MIT shocks, need other methods to study nonlinearities from aggregate risk, etc. [huge literature]

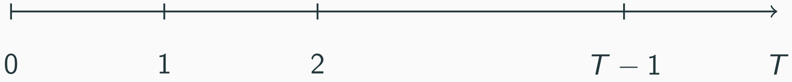
Computing HA Jacobians: the “fake news” algorithm

General formulation of HA blocks

- Start from **discretized** HA decision problem
  - Bellman equation:
$$\mathbf{v}_t = v(\mathbf{v}_{t+1}, X_t) \tag{1}$$
  - Law of motion for distribution of agents:
$$\mathbf{D}_{t+1} = \Lambda(\mathbf{v}_{t+1}, X_t)'\mathbf{D}_t \tag{2}$$
  - Aggregate outcome we need for general equilibrium:
$$Y_t = \mathbf{y}(\mathbf{v}_{t+1}, X_t)'\mathbf{D}_t \tag{3}$$

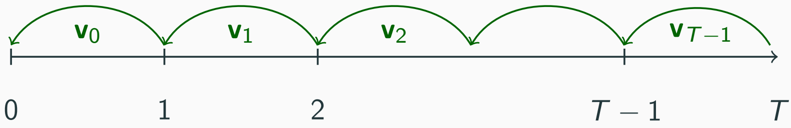
General formulation of HA blocks

- Start from **discretized** HA decision problem
  - Bellman equation:
$$\mathbf{v}_t = v(\mathbf{v}_{t+1}, X_t) \tag{1}$$
  - Law of motion for distribution of agents:
$$\mathbf{D}_{t+1} = \Lambda(\mathbf{v}_{t+1}, X_t)'\mathbf{D}_t \tag{2}$$
  - Aggregate outcome we need for general equilibrium:
$$Y_t = \mathbf{y}(\mathbf{v}_{t+1}, X_t)'\mathbf{D}_t \tag{3}$$
- Standard way to go from $\{X_s\}$ to $\{Y_t\}$:



General formulation of HA blocks

- Start from **discretized** HA decision problem
  - Bellman equation:
$$\mathbf{v}_t = v(\mathbf{v}_{t+1}, X_t) \tag{1}$$
  - Law of motion for distribution of agents:
$$\mathbf{D}_{t+1} = \Lambda(\mathbf{v}_{t+1}, X_t)'\mathbf{D}_t \tag{2}$$
  - Aggregate outcome we need for general equilibrium:
$$Y_t = \mathbf{y}(\mathbf{v}_{t+1}, X_t)'\mathbf{D}_t \tag{3}$$
- Standard way to go from $\{X_s\}$ to $\{Y_t\}$:



General formulation of HA blocks

- Start from **discretized** HA decision problem
  - Bellman equation:

$$\mathbf{v}_t = v(\mathbf{v}_{t+1}, X_t) \tag{1}$$

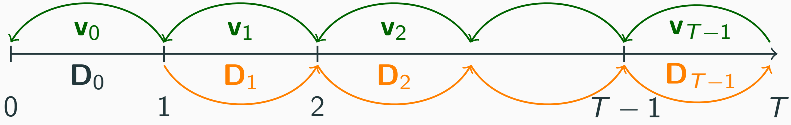
  - Law of motion for distribution of agents:

$$\mathbf{D}_{t+1} = \Lambda(\mathbf{v}_{t+1}, X_t)'\mathbf{D}_t \tag{2}$$

  - Aggregate outcome we need for general equilibrium:

$$Y_t = \mathbf{y}(\mathbf{v}_{t+1}, X_t)'\mathbf{D}_t \tag{3}$$

- Standard way to go from $\{X_s\}$ to $\{Y_t\}$:



General formulation of HA blocks

- Start from **discretized** HA decision problem
  - Bellman equation:

$$\mathbf{v}_t = v(\mathbf{v}_{t+1}, X_t) \tag{1}$$

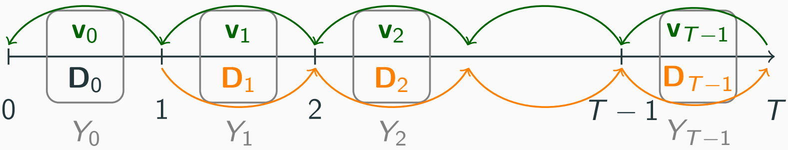
  - Law of motion for distribution of agents:

$$\mathbf{D}_{t+1} = \Lambda(\mathbf{v}_{t+1}, X_t)'\mathbf{D}_t \tag{2}$$

  - Aggregate outcome we need for general equilibrium:

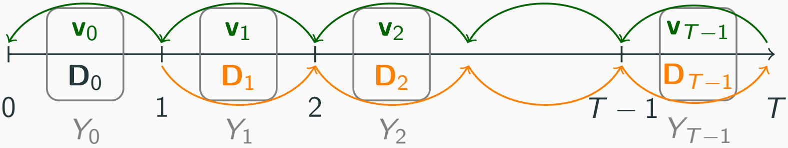
$$Y_t = \mathbf{y}(\mathbf{v}_{t+1}, X_t)'\mathbf{D}_t \tag{3}$$

- Standard way to go from $\{X_s\}$ to $\{Y_t\}$:



General formulation of HA blocks

- Start from **discretized** HA decision problem
  - Bellman equation:
$$\mathbf{v}_t = v(\mathbf{v}_{t+1}, X_t) \tag{1}$$
  - Law of motion for distribution of agents:
$$\mathbf{D}_{t+1} = \Lambda(\mathbf{v}_{t+1}, X_t)'\mathbf{D}_t \tag{2}$$
  - Aggregate outcome we need for general equilibrium:
$$Y_t = \mathbf{y}(\mathbf{v}_{t+1}, X_t)'\mathbf{D}_t \tag{3}$$
- Standard way to go from $\{X_s\}$ to $\{Y_t\}$:



- **Direct algorithm** for Jacobian $[\partial Y_t / \partial X_s]$: repeat this for shocks dX_s for each $s = 0, \ldots, T-1$

Calculating the Jacobian: direct vs. new algorithm

- **Direct algorithm** requires a separate backward and forward iteration for each $s = 0, \ldots, T-1$. $\Longrightarrow$ **costly**.
- **Our method** avoids redundancies in direct algorithm, uses **single** backward & forward iteration to get all T columns
  - improves speed by factor of roughly T with no loss in accuracy

Calculating the Jacobian: direct vs. new algorithm

- **Direct algorithm** requires a separate backward and forward iteration for each $s = 0, \ldots, T-1$. $\implies$ **costly**.
- **Our method** avoids redundancies in direct algorithm, uses **single** backward & forward iteration to get all T columns
  - improves speed by factor of roughly T with no loss in accuracy
- Key is to exploit time symmetries in linearized system

$$
\begin{aligned}
dY_t &= d\mathbf{y}_t' \mathbf{D}_{ss} + \mathbf{y}_{ss}' d\mathbf{D}_t \\
d\mathbf{D}_{t+1} &= d\Lambda_t' \mathbf{D}_{ss} + \Lambda_{ss}' d\mathbf{D}_t \\
\mathbf{D}_0 &= \mathbf{D}_{ss}
\end{aligned}
$$

Calculating the Jacobian: direct vs. new algorithm

- **Direct algorithm** requires a separate backward and forward iteration for each $s = 0, \ldots, T-1$. $\implies$ **costly**.
- **Our method** avoids redundancies in direct algorithm, uses **single** backward & forward iteration to get all T columns
  - improves speed by factor of roughly T with no loss in accuracy
- Key is to exploit time symmetries in linearized system

$$\begin{aligned} dY_t &= d\mathbf{y}_t'\mathbf{D}_{ss} + \mathbf{y}_{ss}' d\mathbf{D}_t \\ d\mathbf{D}_{t+1} &= d\Lambda_t'\mathbf{D}_{ss} + \Lambda_{ss}' d\mathbf{D}_t \\ \mathbf{D}_0 &= \mathbf{D}_{ss} \end{aligned}$$

- Since everything is linear, look at terms from backward and forward iterations separately, then combine

Terms from backward iteration

Q: Holding $\mathbf{D}_t = \mathbf{D}_{ss}$ fixed, what is the effect of dX_s?

$$dY_t = d\mathbf{y}_t' \mathbf{D}_{ss} + \cancel{\mathbf{y}_{ss}' d\mathbf{D}_t}$$

$$d\mathbf{D}_{t+1} = d\Lambda_t' \mathbf{D}_{ss} + \cancel{\Lambda_{ss}' d\mathbf{D}_t}$$

Terms from backward iteration

Q: Holding $\mathbf{D}_t = \mathbf{D}_{ss}$ fixed, what is the effect of dX_s?

$$dY_t = d\mathbf{y}_t'\mathbf{D}_{ss} + \cancel{\mathbf{y}_{ss}' d\mathbf{D}_t}$$
$$d\mathbf{D}_{t+1} = d\Lambda_t'\mathbf{D}_{ss} + \cancel{\Lambda_{ss}' d\mathbf{D}_t}$$

- dy_t and dΛ_t are purely forward-looking from the Bellman eq.
- They depend only on the time until shock, $s - t$.

Terms from backward iteration

Q: Holding $\mathbf{D}_t = \mathbf{D}_{ss}$ fixed, what is the effect of dX_s?

$$dY_t = d\mathbf{y}_t' \mathbf{D}_{ss} + \cancel{\mathbf{y}_{ss}' d\mathbf{D}_t}$$
$$d\mathbf{D}_{t+1} = d\Lambda_t' \mathbf{D}_{ss} + \cancel{\Lambda_{ss}' d\mathbf{D}_t}$$

- dy_t and dΛ_t are purely forward-looking from the Bellman eq.
- They depend only on the time until shock, $s - t$.
- Define

$$\mathcal{Y}_{s-t} \equiv d\mathbf{y}_t' \mathbf{D}_{ss} / dX_s$$
$$\mathcal{D}_{s-t} \equiv d\Lambda_t' \mathbf{D}_{ss} / dX_s$$

- Use a **single backward iteration**, with shock dX_s at $s = T - 1$, to obtain $d\mathbf{y}_t$ and $d\Lambda_t'$ for all $t = T - 1, \ldots, 0$

Terms from forward iteration

Q: Holding $\mathbf{y}_t = \mathbf{y}_{ss}$ and $\Lambda_t = \Lambda_{ss}$ fixed, what is the effect of $d\mathbf{D}_s$?

$$dY_t = \cancel{d\mathbf{y}_t' \mathbf{D}_{ss}} + \mathbf{y}_{ss}' d\mathbf{D}_t$$

$$d\mathbf{D}_{t+1} = \cancel{d\Lambda_t' \mathbf{D}_{ss}} + \Lambda_{ss}' d\mathbf{D}_t$$

Terms from forward iteration

Q: Holding $\mathbf{y}_t = \mathbf{y}_{ss}$ and $\Lambda_t = \Lambda_{ss}$ fixed, what is the effect of $d\mathbf{D}_s$?

$$dY_t = \cancel{d\mathbf{y}_t'\mathbf{D}_{ss}} + \mathbf{y}_{ss}' d\mathbf{D}_t$$
$$d\mathbf{D}_{t+1} = \cancel{d\Lambda_t'\mathbf{D}_{ss}} + \Lambda_{ss}' d\mathbf{D}_t$$

- Since everything's linear, some linear functional (row vector) maps $d\mathbf{D}_s$ to dY_t: what is it?

Terms from forward iteration

Q: Holding $\mathbf{y}_t = \mathbf{y}_{ss}$ and $\Lambda_t = \Lambda_{ss}$ fixed, what is the effect of $d\mathbf{D}_s$?

$$dY_t = \cancel{d\mathbf{y}_t'\mathbf{D}_{ss}} + \mathbf{y}_{ss}' d\mathbf{D}_t$$

$$d\mathbf{D}_{t+1} = \cancel{d\Lambda_t'\mathbf{D}_{ss}} + \Lambda_{ss}' d\mathbf{D}_t$$

- Since everything's linear, some linear functional (row vector) maps $d\mathbf{D}_s$ to dY_t: what is it?
- Can write for $s \leq t$ (zero for $s > t$)

$$dY_t = \underbrace{\mathbf{y}_{ss}'(\Lambda_{ss}')^{t-s}}_{\equiv \mathcal{P}_{t-s}'} d\mathbf{D}_s \tag{4}$$

Terms from forward iteration

Q: Holding $\mathbf{y}_t = \mathbf{y}_{ss}$ and $\Lambda_t = \Lambda_{ss}$ fixed, what is the effect of $d\mathbf{D}_s$?

$$dY_t = \cancel{d\mathbf{y}_t'\mathbf{D}_{ss}} + \mathbf{y}_{ss}' d\mathbf{D}_t$$
$$d\mathbf{D}_{t+1} = \cancel{d\Lambda_t'\mathbf{D}_{ss}} + \Lambda_{ss}' d\mathbf{D}_t$$

- Since everything's linear, some linear functional (row vector) maps $d\mathbf{D}_s$ to dY_t: what is it?
- Can write for $s \le t$ (zero for $s > t$)

$$dY_t = \underbrace{\mathbf{y}_{ss}'(\Lambda_{ss}')^{t-s}}_{\equiv \mathcal{P}_{t-s}'} d\mathbf{D}_s \tag{4}$$

- These $\mathcal{P}_u'$ can be calculated recursively as $\mathcal{P}_u = \Lambda_{ss}\mathcal{P}_{u-1}$
  - Λ_{ss} is ss transition, transpose of Λ_{ss}' used in forward iteration

Terms from forward iteration

Q: Holding $\mathbf{y}_t = \mathbf{y}_{ss}$ and $\Lambda_t = \Lambda_{ss}$ fixed, what is the effect of $d\mathbf{D}_s$?

$$dY_t = \cancel{d\mathbf{y}_t'\mathbf{D}_{ss}} + \mathbf{y}_{ss}' d\mathbf{D}_t$$
$$d\mathbf{D}_{t+1} = \cancel{d\Lambda_t'\mathbf{D}_{ss}} + \Lambda_{ss}' d\mathbf{D}_t$$

- Since everything's linear, some linear functional (row vector) maps $d\mathbf{D}_s$ to dY_t: what is it?
- Can write for $s \leq t$ (zero for $s > t$)

$$dY_t = \underbrace{\mathbf{y}_{ss}'(\Lambda_{ss}')^{t-s}}_{\equiv \mathcal{P}_{t-s}'} d\mathbf{D}_s \tag{4}$$

- These $\mathcal{P}_u'$ can be calculated recursively as $\mathcal{P}_u = \Lambda_{ss}\mathcal{P}_{u-1}$
  - Λ_{ss} is ss transition, transpose of Λ_{ss}' used in forward iteration

$\implies$ A **single "transpose" forward iteration** get all $\mathcal{P}_u'$.

Putting it together: the Jacobian

- For shock dX_s, want

$$dY_t = d\mathbf{y}_t'\mathbf{D}_{ss} + \mathbf{y}_{ss}' d\mathbf{D}_t$$

Putting it together: the Jacobian

- For shock dX_s, want

$$dY_t = d\mathbf{y}_t'\mathbf{D}_{ss} + \mathbf{y}_{ss}' d\mathbf{D}_t$$

- Change in policy, holding distribution constant:

$$d\mathbf{y}_t'\mathbf{D}_{ss} = \mathcal{Y}_{s-t} dX_s$$

Putting it together: the Jacobian

- For shock dX_s, want

$$dY_t = d\mathbf{y}_t'\mathbf{D}_{ss} + \mathbf{y}_{ss}' d\mathbf{D}_t$$

- Change in policy, holding distribution constant:

$$d\mathbf{y}_t'\mathbf{D}_{ss} = \mathcal{Y}_{s-t} dX_s$$

- Change in distribution, holding policy constant:

$$\mathbf{y}_{ss}' d\mathbf{D}_t = (\mathcal{P}_0'\mathcal{D}_{s-t+1} + \ldots + \mathcal{P}_{t-1}'\mathcal{D}_s) dX_s$$

  sum of shocks $\mathcal{D}$ to distribution from anticipating dX_s before t, propagated forward to effect on dY_t with $\mathcal{P}$

Putting it together: the Jacobian

- For shock dX_s, want

$$dY_t = d\mathbf{y}_t'\mathbf{D}_{ss} + \mathbf{y}_{ss}'d\mathbf{D}_t$$

- Change in policy, holding distribution constant:

$$d\mathbf{y}_t'\mathbf{D}_{ss} = \mathcal{Y}_{s-t}dX_s$$

- Change in distribution, holding policy constant:

$$\mathbf{y}_{ss}'d\mathbf{D}_t = (\mathcal{P}_0'\mathcal{D}_{s-t+1} + \ldots + \mathcal{P}_{t-1}'\mathcal{D}_s)dX_s$$

  sum of shocks $\mathcal{D}$ to distribution from anticipating dX_s before t, propagated forward to effect on dY_t with $\mathcal{P}$

- From $\mathcal{Y}$s, $\mathcal{D}$s, $\mathcal{P}$s, **get entire Jacobian**:

$$\mathcal{J}_{t,s} \equiv \frac{\partial Y_t}{\partial X_s} = \mathcal{Y}_{t-s} + \mathcal{P}_0'\mathcal{D}_{s-t+1} + \ldots + \mathcal{P}_{t-1}'\mathcal{D}_s$$

“Fake news” matrix

- Define the **fake news matrix** as

$$\mathcal{F}_{t,s} \equiv \begin{cases} \mathcal{Y}_s & \text{for } t = 0 \\ \mathcal{P}'_{t-1}\mathcal{D}_s & \text{for } t > 0 \end{cases} \tag{5}$$

- Interpretation
  - $t = 0$: agents get news of shock at date $s \geq 0$
  - $t \geq 1$: news of unrealized shock disappears (“fake news”)

"Fake news" matrix

- Define the **fake news matrix** as

$$\mathcal{F}_{t,s} \equiv \begin{cases} \mathcal{Y}_s & \text{for } t = 0 \\ \mathcal{P}'_{t-1}\mathcal{D}_s & \text{for } t > 0 \end{cases} \tag{5}$$

- Interpretation
  - $t = 0$: agents get news of shock at date $s \geq 0$
  - $t \geq 1$: news of unrealized shock disappears ("fake news")

- Build Jacobian (response to news shocks) recursively from fake news matrix (response to fake news shocks) as

$$\mathcal{J}_{t,s} \equiv \begin{cases} \mathcal{F}_{t,s} & \text{if } t = 0 \text{ or } s = 0 \\ \mathcal{F}_{t,s} + \mathcal{J}_{t-1,s-1} & \text{otherwise} \end{cases}$$

Overview of fake news algorithm (general case)

To obtain Jacobians $\mathcal{J}^{o,i}$ for many inputs dX^i and outputs dY^o:

1. For each input i, perform **backward iteration** with T steps to get all $\mathcal{Y}_u^{o,i}$ and $\mathcal{D}_u^i$.
2. For each output o, perform **transpose forward iteration** with $T-1$ steps to get all $\mathcal{P}_u^o$.
3. For each pair (o,i), construct **fake news matrix** $\mathcal{F}^{o,i}$ from $\mathcal{Y}^{o,i}$ in first row and product $(\mathcal{P}^o)'\mathcal{D}^i$ for other rows.
4. For each pair (o,i), recurse to get $\mathcal{J}^{o,i}$ from $\mathcal{F}^{o,i}$.

Overview of fake news algorithm (general case)

To obtain Jacobians $\mathcal{J}^{o,i}$ for many inputs dX^i and outputs dY^o:

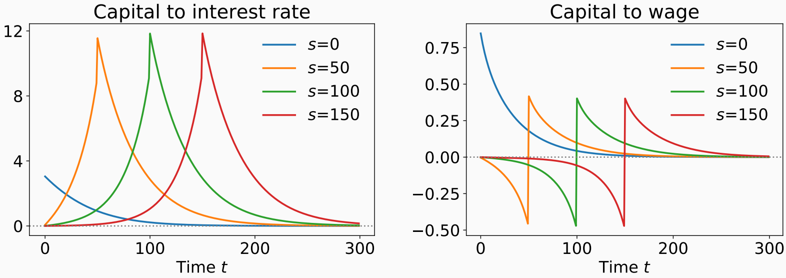
1. For each input i, perform **backward iteration** with T steps to get all $\mathcal{Y}_u^{o,i}$ and $\mathcal{D}_u^i$.
2. For each output o, perform **transpose forward iteration** with $T-1$ steps to get all $\mathcal{P}_u^o$.
3. For each pair (o,i), construct **fake news matrix** $\mathcal{F}^{o,i}$ from $\mathcal{Y}^{o,i}$ in first row and product $(\mathcal{P}^o)'\mathcal{D}^i$ for other rows.
4. For each pair (o,i), recurse to get $\mathcal{J}^{o,i}$ from $\mathcal{F}^{o,i}$.

Step 1 almost always bottleneck

Algorithm accuracy

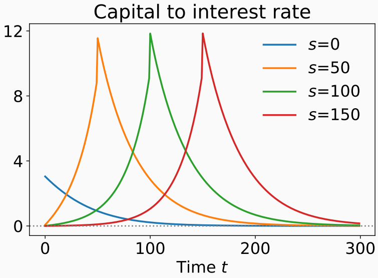
HA Jacobians in Krusell-Smith model

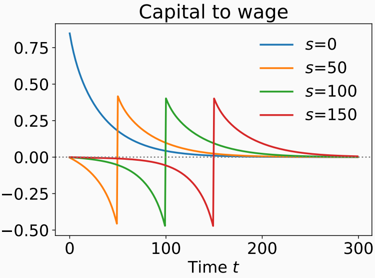
- Inputs $\{r_s, w_s\}$ and outputs $\{\mathcal{K}_t, \mathcal{C}_t\} \implies$ 4 Jacobians.
- For $T = 300$ and $n_{grid} = 3500$, get all $\mathcal{J}$s in **100 ms** on a laptop, for $n_{grid} = 250000$, still just **8 s**.



HA Jacobians in Krusell-Smith model

- Inputs $\{r_s, w_s\}$ and outputs $\{\mathcal{K}_t, \mathcal{C}_t\} \implies$ 4 Jacobians.
- For $T = 300$ and $n_{grid} = 3500$, get all $\mathcal{J}$s in **100 ms** on a laptop, for $n_{grid} = 250000$, still just **8 s**.





- "Asymptotically time invariant" structure: can prove with fake news matrix!

Taking stock

- Dynamic general equilibrium models in sequence space are just a system of nonlinear equations

$$H(\mathbf{U}, \mathbf{Z}) = 0, \qquad \mathbf{U} \in \mathbb{R}^{n_u \times T}, \quad \mathbf{Z} \in \mathbb{R}^{n_z \times T}.$$

  - get linearized solution in one step as $d\mathbf{U} = -\mathbf{H}_U^{-1}\mathbf{H}_Z \, d\mathbf{Z}$
  - efficient algorithm for computing the Jacobians of HA blocks
  - **Jacobians = all that matters from micro heterogeneity**

Taking stock

- Dynamic general equilibrium models in sequence space are just a system of nonlinear equations

$$H(\mathbf{U},\mathbf{Z}) = 0, \qquad \mathbf{U} \in \mathbb{R}^{n_u \times T}, \quad \mathbf{Z} \in \mathbb{R}^{n_z \times T}.$$

  - get linearized solution in one step as $d\mathbf{U} = -\mathbf{H}_U^{-1}\mathbf{H}_Z \, d\mathbf{Z}$
  - efficient algorithm for computing the Jacobians of HA blocks
  - **Jacobians = all that matters from micro heterogeneity**
- Where does that leave us?
  - quantitative DSGE models easily have $n_u \approx 20$ endog vars
  - typical application requires at least $T \approx 200$
  - for models that are complex on **macro** side, getting $\mathbf{H}_U, \mathbf{H}_Z$ and solving $\mathbf{H}_U^{-1}\mathbf{H}_Z$ can become difficult

Taking stock

- Dynamic general equilibrium models in sequence space are just a system of nonlinear equations

$$H(\mathbf{U}, \mathbf{Z}) = 0, \qquad \mathbf{U} \in \mathbb{R}^{n_u \times T}, \quad \mathbf{Z} \in \mathbb{R}^{n_z \times T}.$$

  - get linearized solution in one step as $d\mathbf{U} = -\mathbf{H}_U^{-1}\mathbf{H}_Z\, d\mathbf{Z}$
  - efficient algorithm for computing the Jacobians of HA blocks
  - **Jacobians = all that matters from micro heterogeneity**
- Where does that leave us?
  - quantitative DSGE models easily have $n_u \approx 20$ endog vars
  - typical application requires at least $T \approx 200$
  - for models that are complex on **macro** side, getting $\mathbf{H}_U, \mathbf{H}_Z$ and solving $\mathbf{H}_U^{-1}\mathbf{H}_Z$ can become difficult
- **Next**: intuitive directed graph representation to address this

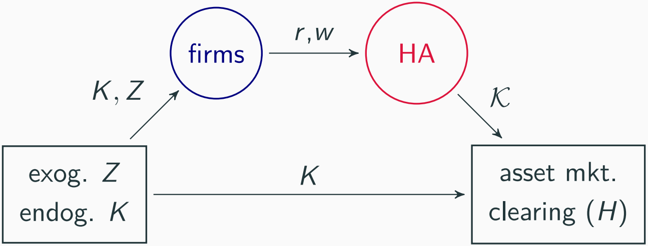
Equilibrium as a Directed Graph

Krusell-Smith model as a directed acyclic graph (DAG)

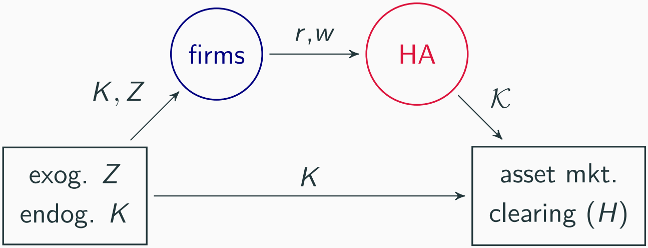
- Recall that we wrote equilibrium in KS model as system:

$$H_t(\mathbf{K},\mathbf{Z}) \equiv \mathcal{K}_t\left(\left\{\underbrace{\alpha Z_s K_{s-1}^{\alpha-1}-\delta}_{r_s},\underbrace{(1-\alpha)Z_s K_{s-1}^{\alpha}}_{w_s}\right\}\right)-K_t=0$$

- This corresponds to a simple graph:

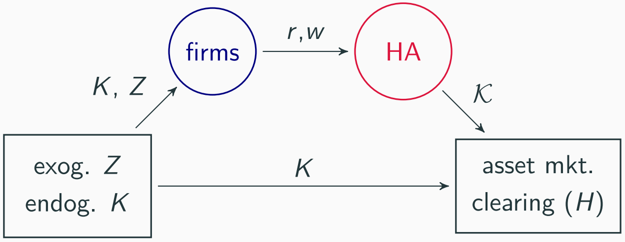


SHADE models: DAGs combining simple and HA blocks



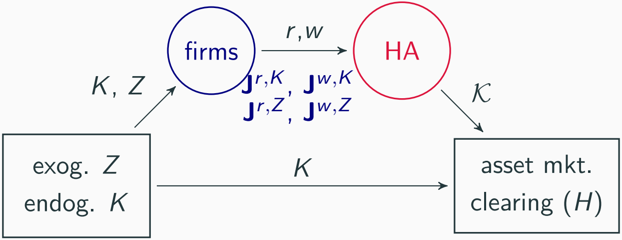
- Each node ("block") takes sequences as inputs & outputs.
  - inputs of later blocks are outputs of earlier blocks
  - # of endogenous variables ("unknowns") equals # of equations in H ("targets"), solve to get equilibrium
- Two kinds of blocks in SHADE models:
  - Simple blocks: analytical equations directly in aggregates, e.g. $r_t = \alpha Z_t K_{t-1}^{\alpha-1} - \delta$, Jacobians sparse and easy to obtain
  - HA blocks: as described previously

Solving the model with DAG



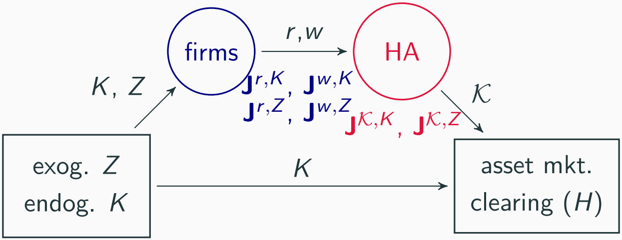
- Define $\mathbf{J}^{o,i}$ to be the **total derivative** o along DAG with respect to exog or endog i (distinct from block Jacobian $\mathcal{J}^{o,i}$)

Solving the model with DAG



- Define $\mathbf{J}^{o,i}$ to be the **total derivative** o along DAG with respect to exog or endog i (distinct from block Jacobian $\mathcal{J}^{o,i}$)
- Here, $\mathbf{J}^{r,K}$ equals $\mathcal{J}^{r,K}$, etc: direct effect is only effect

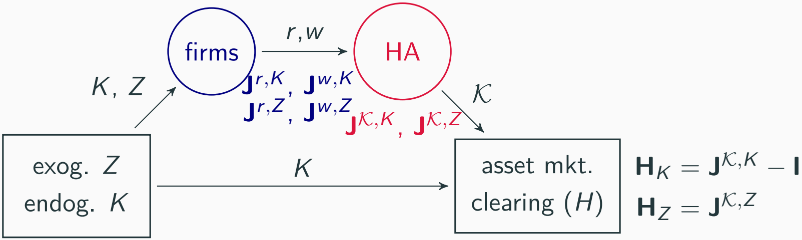
Solving the model with DAG



- Define $\mathbf{J}^{o,i}$ to be the **total derivative** o along DAG with respect to exog or endog i (distinct from block Jacobian $\mathcal{J}^{o,i}$)
- Here, $\mathbf{J}^{r,K}$ equals $\mathcal{J}^{r,K}$, etc: direct effect is only effect
- Then we need chain rule, for instance:

$$\mathbf{J}^{\mathcal{K},K} = \mathcal{J}^{\mathcal{K},r}\mathbf{J}^{r,K} + \mathcal{J}^{\mathcal{K},w}\mathbf{J}^{w,K}$$

Solving the model with DAG



- Define $\mathbf{J}^{o,i}$ to be the **total derivative** o along DAG with respect to exog or endog i (distinct from block Jacobian $\mathcal{J}^{o,i}$)
- Here, $\mathbf{J}^{r,K}$ equals $\mathcal{J}^{r,K}$, etc: direct effect is only effect
- Then we need chain rule, for instance:

$$\mathbf{J}^{\mathcal{K},K} = \mathcal{J}^{\mathcal{K},r}\mathbf{J}^{r,K} + \mathcal{J}^{\mathcal{K},w}\mathbf{J}^{w,K}$$

- Finally get $\mathbf{H}_K$ and $\mathbf{H}_Z$: $\mathbf{G}^{K,Z} \equiv -\mathbf{H}_K^{-1}\mathbf{H}_Z$ is general equilibrium map from Z to K, get all $\mathbf{G}^{o,Z}$ from it

General case: solving with forward accumulation

- Initialize $\mathbf{J}^{i,i}$ as identity for all exogenous or unknown i
- Evaluate chain rule following a topological sort

$$\mathbf{J}^{o,i} = \sum_{m \in \mathcal{I}_b} \mathcal{J}^{o,m} \mathbf{J}^{m,i}$$

  on blocks b to get $\mathbf{H}_U = \mathbf{J}^{H,U}$ and $\mathbf{H}_Z = \mathbf{J}^{H,Z}$
  - Called "forward accumulation" in algorithmic differentiation lit

General case: solving with forward accumulation

- Initialize $\mathbf{J}^{i,i}$ as identity for all exogenous or unknown i
- Evaluate chain rule following a topological sort

$$\mathbf{J}^{o,i} = \sum_{m \in \mathcal{I}_b} \mathcal{J}^{o,m} \mathbf{J}^{m,i}$$

  on blocks b to get $\mathbf{H}_U = \mathbf{J}^{H,U}$ and $\mathbf{H}_Z = \mathbf{J}^{H,Z}$
  - Called "forward accumulation" in algorithmic differentiation lit
- Compute $\mathbf{G}^{U,Z} = -\mathbf{H}_U^{-1} \mathbf{H}_Z$

General case: solving with forward accumulation

- Initialize $\mathbf{J}^{i,i}$ as identity for all exogenous or unknown i
- Evaluate chain rule following a topological sort

$$\mathbf{J}^{o,i} = \sum_{m \in \mathcal{I}_b} \mathcal{J}^{o,m} \mathbf{J}^{m,i}$$

  on blocks b to get $\mathbf{H}_U = \mathbf{J}^{H,U}$ and $\mathbf{H}_Z = \mathbf{J}^{H,Z}$

  - Called "forward accumulation" in algorithmic differentiation lit

- Compute $\mathbf{G}^{U,Z} = -\mathbf{H}_U^{-1}\mathbf{H}_Z$
- Then do forward accumulation

$$\mathbf{G}^{o,Z} = \sum_{m \in \mathcal{I}_b} \mathcal{J}^{o,m} \mathbf{G}^{m,Z}$$

  to get general equilibrium mappings $\mathbf{G}^{o,Z}$ from shocks $d\mathbf{Z}$ to all variables o of interest

General case: solving with forward accumulation

- Initialize $\mathbf{J}^{i,i}$ as identity for all exogenous or unknown i
- Evaluate chain rule following a topological sort

$$\mathbf{J}^{o,i} = \sum_{m \in \mathcal{I}_b} \mathcal{J}^{o,m} \mathbf{J}^{m,i}$$

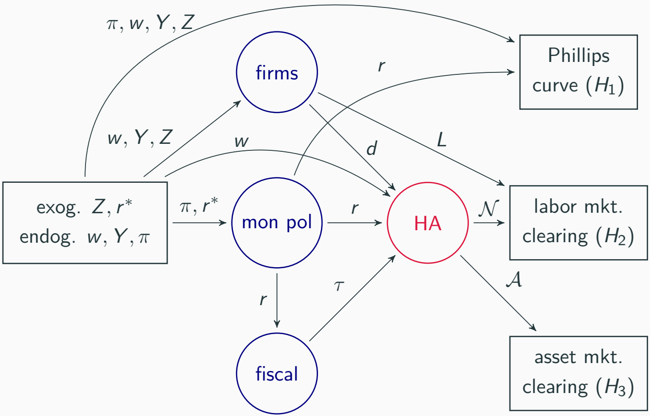
  on blocks b to get $\mathbf{H}_U = \mathbf{J}^{H,U}$ and $\mathbf{H}_Z = \mathbf{J}^{H,Z}$
  - Called "forward accumulation" in algorithmic differentiation lit
- Compute $\mathbf{G}^{U,Z} = -\mathbf{H}_U^{-1}\mathbf{H}_Z$
- Then do forward accumulation

$$\mathbf{G}^{o,Z} = \sum_{m \in \mathcal{I}_b} \mathcal{J}^{o,m} \mathbf{G}^{m,Z}$$

  to get general equilibrium mappings $\mathbf{G}^{o,Z}$ from shocks $d\mathbf{Z}$ to all variables o of interest
- This gives **impulse response to every shock simultaneously**

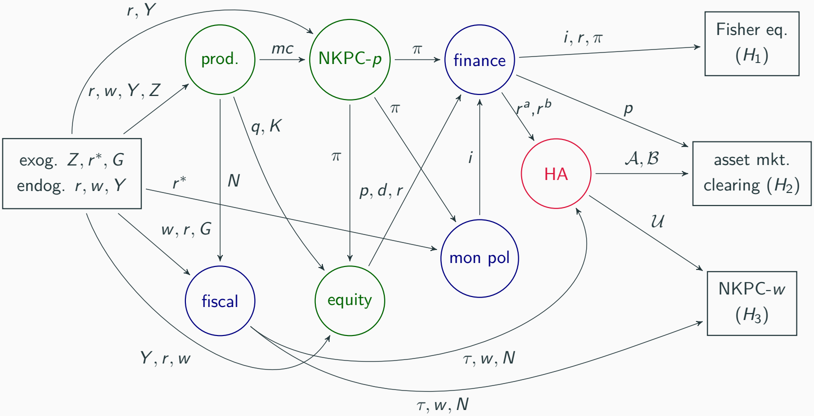
One-asset HANK model with endogenous labor as DAG

- 8 endog vars → 3 unknowns in DAG



Two-asset HANK model with capital and sticky prices/wages

- 22 endog vars → 3 unknowns in DAG



Questions about DAG and equilibrium

Q: Are the DAG and forward accumulation necessary?

- No, we could always write as a giant system of equations
- But solving could be prohibitively costly for models with too many endogenous aggregates...
- For models in paper, DAG saves 2-20x in calculation time

Questions about DAG and equilibrium

Q: Are the DAG and forward accumulation necessary?

- No, we could always write as a giant system of equations
- But solving could be prohibitively costly for models with too many endogenous aggregates...
- For models in paper, DAG saves 2-20x in calculation time

Q: What kinds of models are SHADE models?

- Anything we can stitch together from simple and HA blocks
- Could replace representative-firm prod block with HA block

Questions about DAG and equilibrium

Q: Are the DAG and forward accumulation necessary?

- No, we could always write as a giant system of equations
- But solving could be prohibitively costly for models with too many endogenous aggregates...
- For models in paper, DAG saves 2-20x in calculation time

Q: What kinds of models are SHADE models?

- Anything we can stitch together from simple and HA blocks
- Could replace representative-firm prod block with HA block

Q: Is there any loss in accuracy?

- No, rewriting as DAG shrinks macro system $H(\mathbf{U}, \mathbf{Z}) = 0$ with no additional approximation
- Truncation still only error, minimal for reasonable T

Measuring truncation error

Second moments and estimation

Second moments in a stochastic model

- Assume $\{d\mathbf{Z}_t\}$ can be written as $MA(\infty)$ in iid structural innovation vectors $\{\boldsymbol{\epsilon}_t\}$:

$$d\mathbf{Z}_t = \sum_{s=0}^{\infty} \mathbf{M}_s^Z \boldsymbol{\epsilon}_{t-s}$$

Second moments in a stochastic model

- Assume $\{d\mathbf{Z}_t\}$ can be written as $MA(\infty)$ in iid structural innovation vectors $\{\boldsymbol{\epsilon}_t\}$:

$$d\mathbf{Z}_t = \sum_{s=0}^{\infty} \mathbf{M}_s^Z \boldsymbol{\epsilon}_{t-s}$$

- Thanks to **certainty equivalence**, any $\{dX_t\}$ is also $MA(\infty)$:

$$dX_t = \sum_{s=0}^{\infty} \mathbf{M}_s^X \boldsymbol{\epsilon}_{t-s}$$

where (stacked) $\mathbf{M}^X = \mathbf{G}^{X,Z}\mathbf{M}^Z$

Second moments in a stochastic model

- Assume $\{d\mathbf{Z}_t\}$ can be written as $MA(\infty)$ in iid structural innovation vectors $\{\boldsymbol{\epsilon}_t\}$:

$$d\mathbf{Z}_t = \sum_{s=0}^{\infty} \mathbf{M}_s^Z \boldsymbol{\epsilon}_{t-s}$$

- Thanks to **certainty equivalence**, any $\{dX_t\}$ is also $MA(\infty)$:

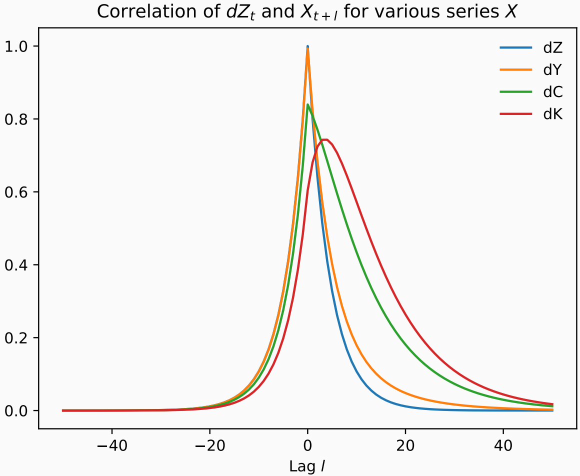
$$dX_t = \sum_{s=0}^{\infty} \mathbf{M}_s^X \boldsymbol{\epsilon}_{t-s}$$

  where (stacked) $\mathbf{M}^X = \mathbf{G}^{X,Z}\mathbf{M}^Z$

- Covariances at any lag given by standard expression

$$\text{Cov}(dX_t, dY_{t'}) = \sum_{s=0}^{\infty} (\mathbf{M}_s^X)(\mathbf{M}_{s+t'-t}^Y)'$$

Krusell-Smith with AR(1): $M_s^Z = 0.9^s$



Given **G**, can use FFT to simultaneously calculate all 2nd moments in at most couple milliseconds, in any of our models—no simulation needed!

From second moments to log-likelihood

- Stacking these covariances at all lags for observed data $\mathbf{Y}$ in $\mathbf{V}(\theta)$, where θ are parameters, can calculate log-likelihood of θ and $\mathbf{Y}$, assuming Gaussian innovations, directly as:

$$\mathcal{L}(\mathbf{Y};\theta) = -\frac{1}{2}\log\det\mathbf{V}(\theta) - \frac{1}{2}\mathbf{Y}'\mathbf{V}(\theta)^{-1}\mathbf{Y}$$

From second moments to log-likelihood

- Stacking these covariances at all lags for observed data $\mathbf{Y}$ in $\mathbf{V}(\theta)$, where θ are parameters, can calculate log-likelihood of θ and $\mathbf{Y}$, assuming Gaussian innovations, directly as:

$$\mathcal{L}(\mathbf{Y};\theta) = -\frac{1}{2}\log\det\mathbf{V}(\theta) - \frac{1}{2}\mathbf{Y}'\mathbf{V}(\theta)^{-1}\mathbf{Y}$$

- No Kalman filter! Old estimation strategy in time series.
    - several recent revivals in DSGE (e.g. Mankiw and Reis 2007)
    - use Cholesky or Levinson on $\mathbf{V}$, or Whittle approx when T large
    - first application to het agents, perfectly suited for sequence-space methods!

From second moments to log-likelihood

- Stacking these covariances at all lags for observed data $\mathbf{Y}$ in $\mathbf{V}(\theta)$, where θ are parameters, can calculate log-likelihood of θ and $\mathbf{Y}$, assuming Gaussian innovations, directly as:

$$\mathcal{L}(\mathbf{Y};\theta) = -\frac{1}{2}\log\det\mathbf{V}(\theta) - \frac{1}{2}\mathbf{Y}'\mathbf{V}(\theta)^{-1}\mathbf{Y}$$

- No Kalman filter! Old estimation strategy in time series.
  - several recent revivals in DSGE (e.g. Mankiw and Reis 2007)
  - use Cholesky or Levinson on $\mathbf{V}$, or Whittle approx when T large
  - first application to het agents, perfectly suited for sequence-space methods!
- Estimating shock processes practically free: calculate $\mathbf{G}^{Y,Z}$ once, reuse in $\mathbf{M}^Y = \mathbf{G}^{Y,Z}\mathbf{M}^Z$ over and over

From second moments to log-likelihood

- Stacking these covariances at all lags for observed data $\mathbf{Y}$ in $\mathbf{V}(\theta)$, where θ are parameters, can calculate log-likelihood of θ and $\mathbf{Y}$, assuming Gaussian innovations, directly as:

$$\mathcal{L}(\mathbf{Y};\theta) = -\frac{1}{2}\log\det\mathbf{V}(\theta) - \frac{1}{2}\mathbf{Y}'\mathbf{V}(\theta)^{-1}\mathbf{Y}$$

- No Kalman filter! Old estimation strategy in time series.
  - several recent revivals in DSGE (e.g. Mankiw and Reis 2007)
  - use Cholesky or Levinson on $\mathbf{V}$, or Whittle approx when T large
  - first application to het agents, perfectly suited for sequence-space methods!
- Estimating shock processes practically free: calculate $\mathbf{G}^{Y,Z}$ once, reuse in $\mathbf{M}^Y = \mathbf{G}^{Y,Z}\mathbf{M}^Z$ over and over
- Other estimation still cheap *as long as we don't need to recalculate HA steady state*

Proof-of-concept estimation exercises in paper

- All three models (KS, 1-asset HANK, 2-asset HANK)
    - 3–19 params
- Find posterior modes, Smets and Wouters (2007) data

Proof-of-concept estimation exercises in paper

- All three models (KS, 1-asset HANK, 2-asset HANK)
  - 3–19 params
- Find posterior modes, Smets and Wouters (2007) data
- **Shock process estimation** times (laptop)
  - 1 to 10 ms for single likelihood draw
  - 100 ms to 20 s to get posterior mode

Proof-of-concept estimation exercises in paper

- All three models (KS, 1-asset HANK, 2-asset HANK)
  - 3–19 params
- Find posterior modes, Smets and Wouters (2007) data
- **Shock process estimation** times (laptop)
  - 1 to 10 ms for single likelihood draw
  - 100 ms to 20 s to get posterior mode
- **Shock process & model estimation** times (laptop)
  - 50 to 200 ms for single likelihood draw
  - 10 s to 10 m to get posterior mode
  - (efficiency from reusing some info across draws)

Parameter / shock		Prior distribution	Posterior Mode	Posterior std. dev
TFP shock	s.d.	Invgamma(0.4, 4)	0.223	(0.013)
	AR-1	Beta(0.5, 0.2)	0.134	(0.063)
G shock	s.d.	Invgamma(0.4, 4)	1.357	(0.218)
	AR-1	Beta(0.5, 0.2)	0.830	(0.012)
β shock	s.d.	Invgamma(0.4, 4)	1.077	(0.060)
	AR-1	Beta(0.5, 0.2)	0.944	(0.007)
(+ 4 other shocks...)				
ϕ		Gamma(1.5, 0.25)	1.407	(0.110)
ϕ_y		Gamma(0.5, 0.25)	1.378	(0.257)
κ^p		Gamma(0.1, 0.1)	0.075	(0.043)
κ^w		Gamma(0.1, 0.1)	0.125	(0.035)
ϵ_I		Gamma(4, 2)	2.998	(1.731)

Local determinacy

New criterion exploiting asymptotic time invariance

- In state space, have e.g. Blanchard-Kahn: count stable roots
  - What analogue in sequence space?
  - Could test singularity of $\mathbf{H}_U$: works, but slow and imprecise

New criterion exploiting asymptotic time invariance

- In state space, have e.g. Blanchard-Kahn: count stable roots
  - What analogue in sequence space?
  - Could test singularity of $\mathbf{H}_U$: works, but slow and imprecise
- Asymptotic time invariance for HA blocks extends to SHADE:

$$[\mathbf{H}_U]_{t,s} \to A_{t-s} \quad \text{as } t, s \to \infty$$

New criterion exploiting asymptotic time invariance

- In state space, have e.g. Blanchard-Kahn: count stable roots
  - What analogue in sequence space?
  - Could test singularity of $\mathbf{H}_U$: works, but slow and imprecise

- Asymptotic time invariance for HA blocks extends to SHADE:

$$[\mathbf{H}_U]_{t,s} \to A_{t-s} \quad \text{as } t,s \to \infty$$

- **Winding number criterion**: precise and fast
- **Local determinacy** for generic model if winding number of

$$\det A(\lambda) \equiv \det \sum A_j e^{ij\lambda}; \qquad \lambda \in [0, 2\pi]$$

around the origin is zero

New criterion exploiting asymptotic time invariance

- In state space, have e.g. Blanchard-Kahn: count stable roots
    - What analogue in sequence space?
    - Could test singularity of $\mathbf{H}_U$: works, but slow and imprecise
- Asymptotic time invariance for HA blocks extends to SHADE:

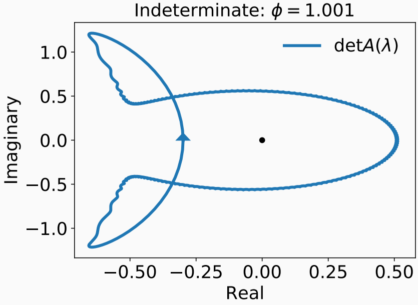
$$[\mathbf{H}_U]_{t,s} \to A_{t-s} \quad \text{as } t, s \to \infty$$

- **Winding number criterion**: precise and fast
- **Local determinacy** for generic model if winding number of

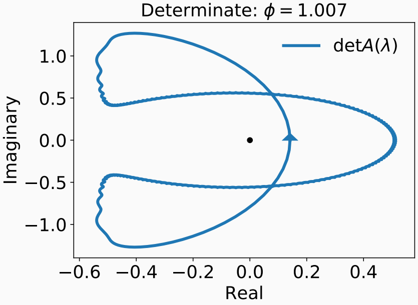
$$\det A(\lambda) \equiv \det \sum A_j e^{ij\lambda}; \qquad \lambda \in [0, 2\pi]$$

  around the origin is zero
    - Generalizes Onatski (2006)
    - Given As, sample many λ and test in less than 1 ms using FFT

Example: determinacy in our one-asset HANK



(Winding number $= -1$)



(Winding number $= 0$)

Nonlinear perfect foresight transitions

Computing nonlinear perfect foresight transitions

- Given Jacobian $\mathbf{H}_U$, can compute full nonlinear solution to

$$H(\mathbf{U}, \mathbf{Z}) = 0$$

Computing nonlinear perfect foresight transitions

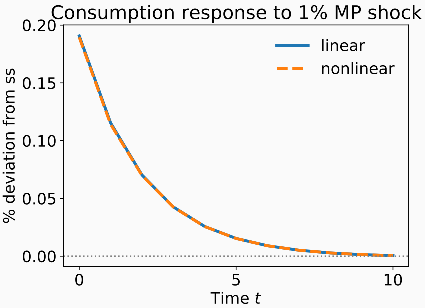
- Given Jacobian $\mathbf{H}_U$, can compute full nonlinear solution to

$$H(\mathbf{U}, \mathbf{Z}) = 0$$

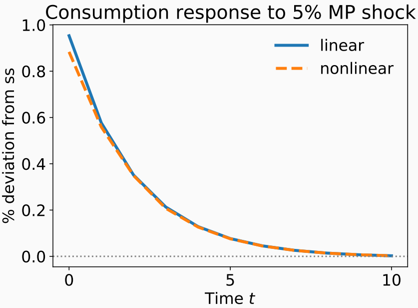
- Idea: use (quasi)-**Newton method**
- Start from $\mathbf{U}^{(0)} = \mathbf{U}_{ss}$ and iterate using

$$\mathbf{U}^{(n)} = \mathbf{U}^{(n-1)} - [\mathbf{H}_U]^{-1} H\left(\mathbf{U}^{(n-1)}, \mathbf{Z}\right)$$

Nonlinear perfect foresight transitions: example



(5 iterations)



(8 iterations)

Recap

Recap

- How to **get sequence-space Jacobians**?
  - Fake news algorithm for HA
  - Forward accumulation on DAG

Recap

- How to **get sequence-space Jacobians**?
  - Fake news algorithm for HA
  - Forward accumulation on DAG

- What can we **do with them**?
  - Get all impulse responses
  - Compute second moments
  - Estimate via likelihood
  - Test local determinacy
  - Compute nonlinear MIT shocks

- **Fast**, **general**, and **accessible!**

Accessible: see notebooks and code!

3.1 Simple blocks

To build intuition, let's start with the firm block. In our code, simple blocks are specified as regular Python functions endowed with the decorator `@simple`. In the body of the function, we directly implement the corresponding equilibrium conditions. The decorator turns the function into an instance of `SimpleBlock`, a simple class with methods to evaluate itself in steady state and along a transition path. Notice the use of K(-1) to denote 1-period lag, similarly to Dynare. In general, one can write (-s) and (+s) to denote s-period lags and leads.

The DAG above has 6 simple nodes. But it makes sense to consolidate all market clearing conditions in a single block. This leaves us with the following five blocks.

In [6]:
```
@simple
def firm(Y, w, Z, pi, mu, kappa):
    L = Y / Z
    Div = Y - w * L - mu/(mu-1)/(2*kappa) * np.log(1+pi)**2 * Y
    return L, Div

@simple
def monetary(pi, rstar, phi):
    r = (1 + rstar(-1) + phi * pi(-1)) / (1 + pi) - 1
    return r

@simple
def fiscal(r, B):
    Tax = r * B
    return Tax

@simple
def nkpc(pi, w, Z, Y, r, mu, kappa):
    nkpc_res = kappa * (w / Z - 1 / mu) + Y(+1) / Y * np.log(1 + pi(+1)) / (1 + r(+1)) - np.log(
    return nkpc_res
```

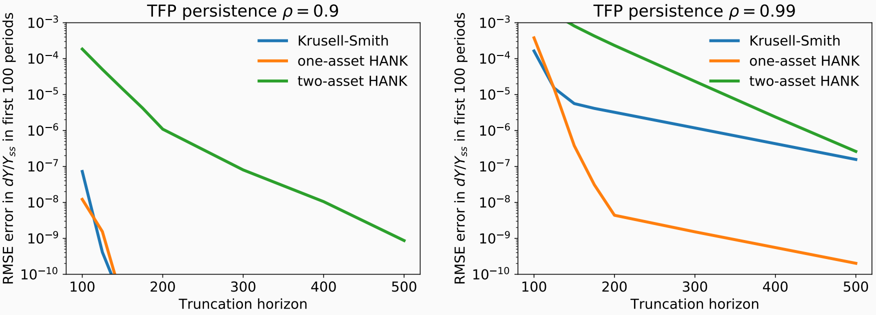
github.com/shade-econ/sequence-jacobian

Thank you!

Measuring truncation error

▸ Back

Use very long $T = 1000$ horizon as benchmark for exact solution (no further convergence apparent after this):



Accuracy of fake news algorithm: all three models

Back