

## 4. Transition Path

Adv. Macro: Heterogenous Agent Models

---

Nicolai Waldstrøm

2024



# Introduction

---

- **Last time:** *Stationary equilibrium (steady states)*

# Introduction

- **Last time:** *Stationary equilibrium (steady states)*
- **Today:** *Transition path (dynamic responses away from steady state)*

# Introduction

- **Last time:** *Stationary equilibrium (steady states)*
- **Today:** *Transition path (dynamic responses away from steady state)*
- **Model:** Heterogeneous Agent Neo-Classical (HANC) model

# Introduction

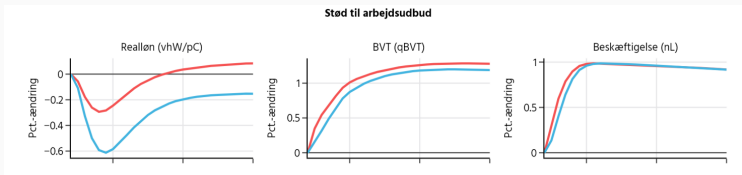
- **Last time:** *Stationary equilibrium (steady states)*
- **Today:** *Transition path (dynamic responses away from steady state)*
- **Model:** Heterogeneous Agent Neo-Classical (HANC) model
- **Code:**
  1. Based on the **GEModelTools** package
  2. Example from **GEModelToolsNotebooks/HANC**  
(except stuff on *linearized solution* and *simulation*)

# Introduction

- **Last time:** *Stationary equilibrium (steady states)*
- **Today:** *Transition path (dynamic responses away from steady state)*
- **Model:** Heterogeneous Agent Neo-Classical (HANC) model
- **Code:**
  1. Based on the **GEModelTools** package
  2. Example from **GEModelToolsNotebooks/HANC**  
(except stuff on *linearized solution* and *simulation*)
- **Literature:**
  1. Auclert et. al. (2021), »Using the Sequence-Space Jacobian to Solve and Estimate Heterogeneous-Agent Models«
  2. Documentation for GEModelTools  
(except stuff on *linearized solution* and *simulation*)
  3. Kirkby (2017)

# Example I

- What do we mean by transition path?
- Permanent shock to labor supply (think increase in retirement age) in the macroeconomic model of the Ministry of Finance:

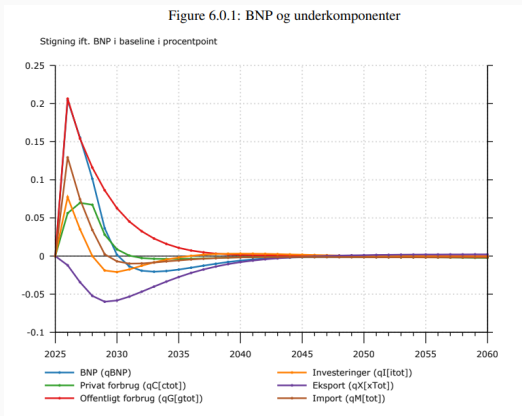


- Note: Permanent shock, so transition path *between* two different steady states



## Example II

- Temporary shock to public spending (i.e. fiscal stimulus during recessions)



- Note: Temporary shock, so model returns to the *same steady state*

## Standard Ramsey model

---

# Ramsey: Summary

- **Simplified form:**

$$\begin{aligned}u'(C_t^{hh}) &= \beta(1 + F_K(K_t, 1) - \delta)u'(C_{t+1}^{hh}) \\K_t &= (1 - \delta)K_{t-1} + F(K_{t-1}, 1) - C_t^{hh}\end{aligned}$$

- **Production function:**  $\Gamma_t K_t^\alpha L_t^{1-\alpha}$
- **Utility function:**  $\frac{(C_t^{hh})^{1-\sigma}}{1-\sigma}$
- **Steady state:**

$$\begin{aligned}K_{ss} &= \left( \frac{\left( \frac{1}{\beta} - 1 + \delta \right)}{\Gamma_{ss}^\alpha} \right)^{\frac{1}{\alpha-1}} \\C_{ss}^{hh} &= (1 - \delta)K_{ss} + \Gamma_{ss} K_{ss}^\alpha - K_{ss}\end{aligned}$$

## Ramsey: As an equation system

$$\begin{bmatrix} r_t^K - \alpha \Gamma_t K_t^{\alpha-1} L_t^{1-\alpha} \\ w_t - (1-\alpha) \Gamma_t K_t^\alpha L_t^{-\alpha} \\ r_t - (r_t^K - \delta) \\ A_t - K_t \\ A_t^{hh} - ((1+r_t)A_{t-1}^{hh} + w_t L_t^{hh} - C_t^{hh}) \\ C_t^{hh,-\sigma} - \beta(1+r_{t+1})C_{t+1}^{hh,-\sigma} \\ A_t - A_t^{hh} \\ L_t - L_t^{hh} \\ \forall t \in \{0, 1, \dots\}, \text{ given } K_{-1} \end{bmatrix} = 0$$

**Remember:** Perfect foresight w.r.t aggregate variables

**Unknowns:**  $\{r_t^K, w_t, L_t, K_t, r_t, A_t, C_t^{hh}, A_t^{hh}\}$  for  $\forall t \in \{0, 1, \dots\}$

## Recap: Newton's method I

- Before solving the dynamic Ramsey model, consider a simpler example

## Recap: Newton's method I

- Before solving the dynamic Ramsey model, consider a simpler example
- Want to solve 1 eq. with 1 unknown ( $x$  is a scalar):

$$f(x) = 0$$

## Recap: Newton's method I

- Before solving the dynamic Ramsey model, consider a simpler example
- Want to solve 1 eq. with 1 unknown ( $x$  is a scalar):

$$f(x) = 0$$

- **How to find  $x$ ?** First-order Taylor approximation around current guess  $x^i$ :

$$f(x) \approx f(x^i) + f'(x^i)(x - x^i)$$

## Recap: Newton's method I

- Before solving the dynamic Ramsey model, consider a simpler example
- Want to solve 1 eq. with 1 unknown ( $x$  is a scalar):

$$f(x) = 0$$

- **How to find  $x$ ?** First-order Taylor approximation around current guess  $x^i$ :

$$f(x) \approx f(x^i) + f'(x^i)(x - x^i)$$

- Set  $f(x) = 0$  and solve for  $x$  to get:

$$x = x^i - \frac{f(x^i)}{f'(x^i)}$$



## Recap: Newton's method II

- Newton's method: Given initial guess  $x_0$  update guess for  $x$  from  $i$  to  $i + 1$  as:

$$x^{i+1} = x^i - \frac{f(x^i)}{f'(x^i)}$$

- until  $|f(x^i)| < \epsilon$

## Recap: Newton's method II

- Newton's method: Given initial guess  $x_0$  update guess for  $x$  from  $i$  to  $i + 1$  as:

$$x^{i+1} = x^i - \frac{f(x^i)}{f'(x^i)}$$

- until  $|f(x^i)| < \epsilon$
- Can always get  $f(x^i)$  by simply evaluating the function at current estimate. What about derivative  $f'(x^i)$ ?

## Recap: Newton's method II

- Newton's method: Given initial guess  $x_0$  update guess for  $x$  from  $i$  to  $i + 1$  as:

$$x^{i+1} = x^i - \frac{f(x^i)}{f'(x^i)}$$

- until  $|f(x^i)| < \epsilon$
- Can always get  $f(x^i)$  by simply evaluating the function at current estimate. What about derivative  $f'(x^i)$ ?
- Use numerical approximation:

$$f'(x^i) \approx \frac{f(x^i + h) - f(x^i)}{h}$$

- For small  $h$ .

## Recap: Newton's method II

- Newton's method: Given initial guess  $x_0$  update guess for  $x$  from  $i$  to  $i + 1$  as:

$$x^{i+1} = x^i - \frac{f(x^i)}{f'(x^i)}$$

- until  $|f(x^i)| < \epsilon$
- Can always get  $f(x^i)$  by simply evaluating the function at current estimate. What about derivative  $f'(x^i)$ ?
- Use numerical approximation:

$$f'(x^i) \approx \frac{f(x^i + h) - f(x^i)}{h}$$

- For small  $h$ .
- How well does it work?
  - If  $f(x)$  is linear this update solves  $f(x) = 0$  in **1 iteration**
  - If  $f(x)$  is non-linear we typically need more iterations, but works well if initial guess is within basin of attraction

## Recap: Multivariate Newton's method

- Generalize to vector-valued, multivariate functions  $[f_1(x_1, x_2), f_2(x_1, x_2)]' = \mathbf{f}(\mathbf{x})$  with  $\mathbf{x} = (x_1, x_2)'$  :

$$\mathbf{x}^{i+1} = \mathbf{x}^i - \mathbf{J}(\mathbf{x}^i)^{-1} \mathbf{f}(\mathbf{x}^i)$$

## Recap: Multivariate Newton's method

- Generalize to vector-valued, multivariate functions  $[f_1(x_1, x_2), f_2(x_1, x_2)]' = \mathbf{f}(\mathbf{x})$  with  $\mathbf{x} = (x_1, x_2)'$  :

$$\mathbf{x}^{i+1} = \mathbf{x}^i - \mathbf{J}(\mathbf{x}^i)^{-1} \mathbf{f}(\mathbf{x}^i)$$

- Where  $\mathbf{J}(\mathbf{x}^i)$  is the *Jacobian* of  $\mathbf{f}(\mathbf{x})$  w.r.t  $\mathbf{x}^i$ :

$$\mathbf{J}(\mathbf{x}_i) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1^i} & \frac{\partial f_1}{\partial x_2^i} \\ \frac{\partial f_2}{\partial x_1^i} & \frac{\partial f_2}{\partial x_2^i} \end{bmatrix}$$

## Recap: Multivariate Newton's method

- Generalize to vector-valued, multivariate functions  $[f_1(x_1, x_2), f_2(x_1, x_2)]' = \mathbf{f}(\mathbf{x})$  with  $\mathbf{x} = (x_1, x_2)'$  :

$$\mathbf{x}^{i+1} = \mathbf{x}^i - \mathbf{J}(\mathbf{x}^i)^{-1} \mathbf{f}(\mathbf{x}^i)$$

- Where  $\mathbf{J}(\mathbf{x}^i)$  is the *Jacobian* of  $\mathbf{f}(\mathbf{x})$  w.r.t  $\mathbf{x}^i$ :

$$\mathbf{J}(\mathbf{x}_i) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1^i} & \frac{\partial f_1}{\partial x_2^i} \\ \frac{\partial f_2}{\partial x_1^i} & \frac{\partial f_2}{\partial x_2^i} \end{bmatrix}$$

- Can calculate this jacobian in the same way as  $f'(x)$  in previous example, but need to do so for every element in  $\mathbf{x}$

## Recap: Multivariate Newton's method

- Generalize to vector-valued, multivariate functions  $[f_1(x_1, x_2), f_2(x_1, x_2)]' = \mathbf{f}(\mathbf{x})$  with  $\mathbf{x} = (x_1, x_2)'$ :

$$\mathbf{x}^{i+1} = \mathbf{x}^i - \mathbf{J}(\mathbf{x}^i)^{-1} \mathbf{f}(\mathbf{x}^i)$$

- Where  $\mathbf{J}(\mathbf{x}^i)$  is the *Jacobian* of  $\mathbf{f}(\mathbf{x})$  w.r.t  $\mathbf{x}^i$ :

$$\mathbf{J}(\mathbf{x}_i) = \begin{bmatrix} \frac{\partial f_1}{\partial x_1^i} & \frac{\partial f_1}{\partial x_2^i} \\ \frac{\partial f_2}{\partial x_1^i} & \frac{\partial f_2}{\partial x_2^i} \end{bmatrix}$$

- Can calculate this jacobian in the same way as  $f'(x)$  in previous example, but need to do so for every element in  $\mathbf{x}$
- Go through code**



## Recap: Broyden's method I

- Newton's method updates Jacobian  $J$  in **every iteration**

## Recap: Broyden's method I

- Newton's method updates Jacobian  $J$  in **every iteration**
- If  $J$  is expensive to calculate, this is a serious bottleneck

## Recap: Broyden's method I

- Newton's method updates Jacobian  $J$  in **every iteration**
- If  $J$  is expensive to calculate, this is a serious bottleneck
- Broyden's method solves this issue by only calculating  $J$  around some initial point.

## Recap: Broyden's method I

- Newton's method updates Jacobian  $J$  in **every iteration**
- If  $J$  is expensive to calculate, this is a serious bottleneck
- Broyden's method solves this issue by only calculating  $J$  around some initial point.
- Then apply following (linear) update of  $f'(x^{i+1})$  at every iteration  $i$ :

$$f'(x^{i+1}) = f'(x^i) + \frac{[f(x^{i+1}) - f(x^i)] - f'(x^i)(x^{i+1} - x^i)}{x^{i+1} - x^i}$$

## Recap: Broyden's method II

1. Guess  $\mathbf{x}^0$  and set  $i = 0$
2. Calculate the Jacobian around initial point  $\mathbf{J}_0$
3. Calculate  $\mathbf{f}^i = \mathbf{f}(\mathbf{x}^i)$ .
4. Stop if  $\|\mathbf{f}^i\|$  below tolerance  $\epsilon$
5. Calculate Jacobian by

$$\mathbf{J}^i = \begin{cases} \mathbf{J}_0 & \text{if } i = 0 \\ \mathbf{J}^{i-1} + \frac{(\mathbf{f}^i - \mathbf{f}^{i-1}) - \mathbf{H}_K^{i-1}(\mathbf{K}^i - \mathbf{K}^{i-1})}{\|\mathbf{x}^i - \mathbf{x}^{i-1}\|_2} (\mathbf{x}^i - \mathbf{x}^{i-1})' & \text{if } i > 0 \end{cases}$$

6. Update guess by  $\mathbf{x}^{i+1} = \mathbf{x}^i - (\mathbf{J}^i)^{-1} \mathbf{f}^i$
7. Increment  $i$  and return to step 3

- Go through code

## Back to Ramsey

$$\begin{bmatrix} r_t^K - \alpha \Gamma_t K_t^{\alpha-1} L_t^{1-\alpha} \\ w_t - (1-\alpha) \Gamma_t K_t^\alpha L_t^{-\alpha} \\ r_t - (r_t^K - \delta) \\ A_t - K_t \\ A_t^{hh} - ((1+r_t)A_{t-1}^{hh} + w_t L_t^{hh} - C_t^{hh}) \\ C_t^{hh,-\sigma} - \beta(1+r_{t+1})C_{t+1}^{hh,-\sigma} \\ A_t - A_t^{hh} \\ L_t - L_t^{hh} \\ \forall t \in \{0, 1, \dots\}, \text{ given } K_{-1} \end{bmatrix} = 0$$

- 2 issues:

## Back to Ramsey

$$\begin{bmatrix} r_t^K - \alpha \Gamma_t K_t^{\alpha-1} L_t^{1-\alpha} \\ w_t - (1-\alpha) \Gamma_t K_t^\alpha L_t^{-\alpha} \\ r_t - (r_t^K - \delta) \\ A_t - K_t \\ A_t^{hh} - ((1+r_t)A_{t-1}^{hh} + w_t L_t^{hh} - C_t^{hh}) \\ C_t^{hh,-\sigma} - \beta(1+r_{t+1})C_{t+1}^{hh,-\sigma} \\ A_t - A_t^{hh} \\ L_t - L_t^{hh} \\ \forall t \in \{0, 1, \dots\}, \text{ given } K_{-1} \end{bmatrix} = 0$$

- 2 issues:
  - Many unknowns (8 eqs per period)

## Back to Ramsey

$$\begin{bmatrix} r_t^K - \alpha \Gamma_t K_t^{\alpha-1} L_t^{1-\alpha} \\ w_t - (1-\alpha) \Gamma_t K_t^\alpha L_t^{-\alpha} \\ r_t - (r_t^K - \delta) \\ A_t - K_t \\ A_t^{hh} - ((1+r_t)A_{t-1}^{hh} + w_t L_t^{hh} - C_t^{hh}) \\ C_t^{hh,-\sigma} - \beta(1+r_{t+1})C_{t+1}^{hh,-\sigma} \\ A_t - A_t^{hh} \\ L_t - L_t^{hh} \\ \forall t \in \{0, 1, \dots\}, \text{ given } K_{-1} \end{bmatrix} = 0$$

- 2 issues:
  - Many unknowns (8 eqs per period)
  - In fact, infinitely many since time is infinite,  $T \rightarrow \infty$



## Truncated Ramsey, reduced vector form

$$H(K, L, \Gamma, K_{-1}) = \begin{bmatrix} A_t - A_t^{hh} \\ L_t - L_t^{hh} \\ \forall t \in \{0, 1, \dots, T-1\} \end{bmatrix} = 0$$

where  $\mathbf{X} = (X_0, X_1, \dots, X_{T-1})$ ,  $A_{-1}^{hh} = K_{-1}$  and

$$r_t^K = \alpha \Gamma_t (K_{t-1}/L_t)^{\alpha-1}$$

$$w_t = (1 - \alpha) \Gamma_t (K_{t-1}/L_t)^\alpha$$

$$A_t = K_t$$

$$r_t = r_t^K - \delta$$

$$C_t^{hh} = (\beta(1 + r_{t+1}))^{-\sigma} C_{t+1}^{hh} \text{ (backwards)}$$

$$L_t^{hh} = 1$$

$$A_t^{hh} = (1 + r_t)A_{t-1}^{hh} + w_t L_t^{hh} - C_t^{hh} \text{ (forwards)}$$

**Truncation:**  $T < \infty$  fine when  $\Gamma_t = \Gamma_{ss}$  for all  $t > \underline{t}$  with  $\underline{t} \ll T$

## Further reduced

$$H(K, \Gamma, K_{-1}) = [\mathbf{A} - \mathbf{A}^{hh}] = \mathbf{0}$$

where  $\mathbf{X} = (X_0, X_1, \dots, X_{T-1})$ ,  $A_{-1}^{hh} = K_{-1}$  and

$$L_t = L_t^{hh} = 1$$

$$r_t^K = \alpha \Gamma_t(K_{t-1}/L_t)^{\alpha-1}$$

$$w_t = (1 - \alpha) \Gamma_t(K_{t-1}/L_t)^\alpha$$

$$A_t = K_t$$

$$r_t = r_t^K - \delta$$

$$C_t^{hh} = (\beta(1 + r_{t+1}))^{-\sigma} C_{t+1}^{hh} \text{ (backwards)}$$

$$A_t^{hh} = (1 + r_t)A_{t-1}^{hh} + w_t L_t^{hh} - C_t^{hh} \text{ (forwards)}$$

for  $\forall t \in \{0, 1, \dots, T-1\}$

# Sequence space

- Note: We have now written the model in *sequence space*
  - Representing an entire timepath/*sequence* of variables as a function of timepath/*sequence* of other variables

# Sequence space

- Note: We have now written the model in *sequence space*
  - Representing an entire timepath/*sequence* of variables as a function of timepath/*sequence* of other variables
- Example: Keynesian consumption function  $C_t = a + bY_t$ :

$$\begin{bmatrix} C_0 & C_1 & C_2 & \dots \end{bmatrix}' = a + b \begin{bmatrix} Y_0 & Y_1 & Y_2 & \dots \end{bmatrix}'$$

$$\Leftrightarrow \mathbf{C} = a + b\mathbf{Y}$$

$$\Leftrightarrow \mathbf{C} = f(\mathbf{Y})$$

# Sequence space

- Note: We have now written the model in *sequence space*
  - Representing an entire timepath/*sequence* of variables as a function of timepath/*sequence* of other variables
- Example: Keynesian consumption function  $C_t = a + bY_t$ :

$$\begin{bmatrix} C_0 & C_1 & C_2 & \dots \end{bmatrix}' = a + b \begin{bmatrix} Y_0 & Y_1 & Y_2 & \dots \end{bmatrix}'$$

$$\Leftrightarrow \mathbf{C} = a + b\mathbf{Y}$$

$$\Leftrightarrow \mathbf{C} = f(\mathbf{Y})$$

- Powerfull since it also applies *non-linear*, forward-looking and backwards-looking eqs:

$$C_t = a + b_0 Y_t + b_1 \log Y_{t-4} + b_2 Y_{t+4}^2$$

$$\Leftrightarrow \mathbf{C} = g(\mathbf{Y})$$

# Sequence space

- Note: We have now written the model in *sequence space*
  - Representing an entire timepath/*sequence* of variables as a function of timepath/*sequence* of other variables
- Example: Keynesian consumption function  $C_t = a + bY_t$ :

$$\begin{aligned} \begin{bmatrix} C_0 & C_1 & C_2 & \dots \end{bmatrix}' &= a + b \begin{bmatrix} Y_0 & Y_1 & Y_2 & \dots \end{bmatrix}' \\ \Leftrightarrow \mathbf{C} &= a + b\mathbf{Y} \\ \Leftrightarrow \mathbf{C} &= f(\mathbf{Y}) \end{aligned}$$

- Powerfull since it also applies *non-linear*, forward-looking and backwards-looking eqs:

$$\begin{aligned} C_t &= a + b_0 Y_t + b_1 \log Y_{t-4} + b_2 Y_{t+4}^2 \\ \Leftrightarrow \mathbf{C} &= g(\mathbf{Y}) \end{aligned}$$

- As long as we have the sequence  $\mathbf{Y}$  we can calculate  $\mathbf{C}$ 
  - Will leverage this later when working with the HA model

# Solution in sequence space

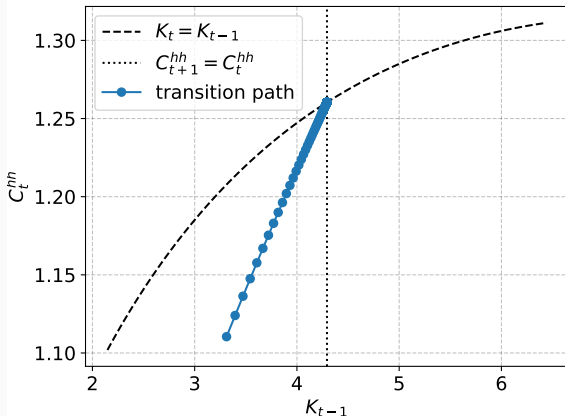
- **Truncation:**  $T = 200$  (transition path should have converged to ss by then)
- **Jacobian:** Find  $\mathbf{H}_K$  by *numerical differentiation*

$$\mathbf{H}_K = \begin{bmatrix} \frac{\partial(A_0 - A_0^{hh})}{\partial K_0} & \frac{\partial(A_0 - A_0^{hh})}{\partial K_1} & \dots \\ \frac{\partial(A_1 - A_1^{hh})}{\partial K_0} & \ddots & \ddots \\ \vdots & \ddots & \ddots \end{bmatrix}$$

- **Transition path:** Given  $\mathbf{\Gamma}$  and  $K_{-1}$  solve  $\mathbf{H}(\mathbf{K}, \mathbf{\Gamma}, K_{-1})$  with non-linear equation system solver (e.g. broyden)
- **Notebook:** *Ramsey.ipynb*

## Example 1: Initially low capital

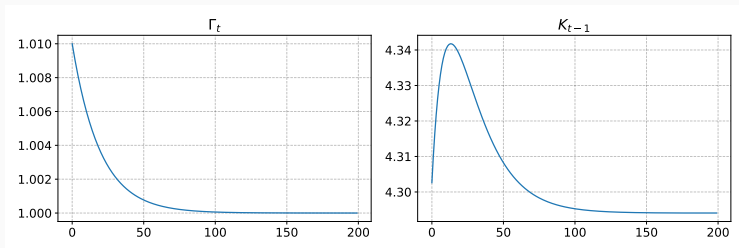
Initially away from steady state:  $K_{-1} = 0.75K_{ss}$





## Example 2: Technology shock

**Technology shock:**  $\Gamma_t = 0.01 \times \Gamma_{ss} \times 0.95^t$  (i.e AR(1) with  $\rho = 0.95$ ) (exogenous, deterministic)



**Terminology:** MIT-shock

## **Transition path with HA**

---

# Equation system

The model can be written as an **equation system**

$$\begin{bmatrix} r_t^K - F_K(K_{t-1}, L_t) \\ w_t - F_L(K_{t-1}, L_t) \\ r_t - (r_t^K - \delta) \\ A_t - K_t \\ \underline{D}_t - \Pi_z \underline{D}_t \\ \underline{D}_{t+1} - \Lambda_t \underline{D}_t \\ A_t - A_t^{hh} \\ L_t - L_t^{hh} \\ \forall t \in \{0, 1, \dots\}, \text{ given } \underline{D}_0 \end{bmatrix} = 0$$

where  $\{\Gamma_t\}_{t \geq 0}$  is a given technology path and  $K_{-1} = \int a_{t-1} d\underline{D}_0$

**Remember:** Policies and choice transitions depend on prices

1. Policy function:  $x_t^* = x^* \left( \{r_\tau, w_\tau\}_{\tau \geq t} \right)$  and  $X_t^{hh} = \sum_i x_{it}^* D_{it} = \mathbf{x}_t^{*'} \underline{D}_t$
2. Choice transition:  $\Lambda_t = \Lambda \left( \{r_\tau, w_\tau\}_{\tau \geq t} \right)$

# Transition path - close to verbal definition

For a given  $\underline{D}_0$  and a path  $\{\Gamma_t\}$

1. Quantities  $\{K_t\}$  and  $\{L_t\}$ ,
2. prices  $\{r_t\}$  and  $\{w_t\}$ ,
3. the distributions  $\{D_t\}$  over  $\beta_i$ ,  $z_t$  and  $a_{t-1}$
4. and the policy functions  $\{a_t^*\}$ ,  $\{\ell_t^*\}$  and  $\{c_t^*\}$

are such that in all periods

1. Firms maximize profits (prices)
2. Household maximize expected utility (policy functions)
3.  $D_t$  is implied by simulating the household problem forwards from  $\underline{D}_0$
4. Mutual fund balance sheet is satisfied
5. The capital market clears
6. The labor market clears
7. The goods market clears

# Reduce size of equation system

- In the equation system above we have many **unknowns** and many **equations**
- Makes finding the solution with Broyden's method since **Jacobian is large**
  - With truncation  $T$  and  $N$  equations/unknowns  $J$  has size  $(T \times N, T \times N,)$
  - $\Rightarrow$  Ekspensive to calculate

# Reduce size of equation system

- In the equation system above we have many **unknowns** and many **equations**
- Makes finding the solution with Broyden's method since **Jacobian is large**
  - With truncation  $T$  and  $N$  equations/unknowns  $J$  has size  $(T \times N, T \times N,)$
  - $\Rightarrow$  Expensive to calculate
- We can typically **exploit model structure** to reduce size of system
  - Did this earlier for Ramsey
  - Now more formally

## Truncated, reduced vector form

$$H(K, L, \Gamma, \underline{D}_0) = \begin{bmatrix} A_t - A_t^{hh} \\ L_t - L_t^{hh} \\ \forall t \in \{0, 1, \dots, T-1\} \end{bmatrix} = \mathbf{0}$$

where  $\mathbf{X} = (X_0, X_1, \dots, X_{T-1})$ ,  $K_{-1} = \int a_{t-1} d\underline{D}_0$  and

$$r_t^K = \alpha \Gamma_t (K_{t-1}/L_t)^{\alpha-1}$$

$$w_t = (1 - \alpha) \Gamma_t (K_{t-1}/L_t)^\alpha$$

$$r_t = r_t^K - \delta$$

$$A_t = K_t$$

$$\underline{D}_t = \Pi'_z \underline{D}_t$$

$$\underline{D}_{t+1} = \Lambda'_t \underline{D}_t$$

$$A_t^{hh} = \mathbf{a}_t^{*'} \underline{D}_t$$

$$L_t^{hh} = \ell_t^{*'} \underline{D}_t$$

$$\forall t \in \{0, 1, \dots, T-1\}$$

**Truncation:**  $T < \infty$  fine when  $\Gamma_t = \Gamma_{ss}$  for all  $t > \underline{t}$  with  $\underline{t} \ll T$

# DAG - Directed Acyclic Graph

- **Orange square:** Shocks (exogenous)
- **Blue square:** Unknowns (endogenous)
- **Green circles:** Blocks (with variables and targets inside)



- This DAG implies: Exo. input + guess  $\Rightarrow$  Firm block  $\Rightarrow$  Mutual fund  $\Rightarrow$  HHs  $\Rightarrow$  Residuals



## Further reduction

$$\mathbf{H}(\mathbf{K}, \Gamma, \underline{\mathbf{D}}_0) = \begin{bmatrix} A_t - A_t^{hh} \\ \forall t \in \{0, 1, \dots, T-1\} \end{bmatrix} = \mathbf{0}$$

where  $\mathbf{X} = (X_0, X_1, \dots, X_{T-1})$ ,  $K_{-1} = \int a_{t-1} d\underline{\mathbf{D}}_0$  and

$$L_t = 1$$

$$r_t^K = \alpha \Gamma_t (K_{t-1}/L_t)^{\alpha-1}$$

$$w_t = (1 - \alpha) \Gamma_t (K_{t-1}/L_t)^\alpha$$

$$A_t = K_t$$

$$r_t = r_t^K - \delta$$

$$\underline{\mathbf{D}}_t = \Pi'_z \underline{\mathbf{D}}_t$$

$$\underline{\mathbf{D}}_{t+1} = \Lambda'_t \underline{\mathbf{D}}_t$$

$$A_t^{hh} = \mathbf{a}_t^{*'} \underline{\mathbf{D}}_t$$

$$\forall t \in \{0, 1, \dots, T-1\}$$

**Truncation:**  $T < \infty$  fine when  $\Gamma_t = \Gamma_{ss}$  for all  $t > \underline{t}$  with  $\underline{t} \ll T$

# Solve with Broyden

- As with standard Ramsey model from before we have:
  - Equation system with  $T$  equations ( $H$ )
  - And  $T$  unknowns ( $K$ )
- If we can calculate the jacobian of  $H$  w.r.t  $K$  we can solve with Broyden's method as before

# How to compute Jacobian?

- How do we compute the Jacobian of the residuals  $H$  w.r.t unknowns  $K$ ?
  - Before: Compute Jacobian of entire model using num. diff
  - **Now:** Use DAG structure + chain rule

# How to compute Jacobian?

- How do we compute the Jacobian of the residuals  $\mathbf{H}$  w.r.t unknowns  $\mathbf{K}$ ?
  - Before: Compute Jacobian of entire model using num. diff
  - **Now**: Use DAG structure + chain rule
- *Example*. Represent model in block form:

$$\mathbf{w}, \mathbf{r}^K = \text{Firm}(\mathbf{K}), \quad \mathbf{A}, \mathbf{r} = \text{MutFund}(\mathbf{K}, \mathbf{r}^K)$$

$$\mathbf{A}^{hh} = hh(\mathbf{r}, \mathbf{w}), \quad \mathbf{A} - \mathbf{A}^{hh} = \mathbf{H}(\mathbf{A}, \mathbf{A}^{hh})$$

# How to compute Jacobian?

- How do we compute the Jacobian of the residuals  $\mathbf{H}$  w.r.t unknowns  $\mathbf{K}$ ?
  - Before: Compute Jacobian of entire model using num. diff
  - Now:** Use DAG structure + chain rule
- Example.* Represent model in block form:

$$\mathbf{w}, \mathbf{r}^K = \text{Firm}(\mathbf{K}), \quad \mathbf{A}, \mathbf{r} = \text{MutFund}(\mathbf{K}, \mathbf{r}^K)$$

$$\mathbf{A}^{hh} = hh(\mathbf{r}, \mathbf{w}), \quad \mathbf{A} - \mathbf{A}^{hh} = \mathbf{H}(\mathbf{A}, \mathbf{A}^{hh})$$

- Let  $\mathcal{J}^{y,x}$  be Jacobian of  $y$  w.r.t  $x$ . Then:

$$\begin{aligned} \mathbf{H}_K \equiv \mathcal{J}^{A-A^{hh}, K} &= \mathcal{J}^{A-A^{hh}, A} \mathcal{J}^{A, K} \\ &+ \mathcal{J}^{A-A^{hh}, A^{hh}} \mathcal{J}^{A^{hh}, r} \mathcal{J}^{r, r^k} \mathcal{J}^{r^k, K} \\ &+ \mathcal{J}^{A_t-A^{hh}, A^{hh}} \mathcal{J}^{A^{hh}, w} \mathcal{J}^{w, K} \end{aligned}$$

# How to compute Jacobian?

- If we have individuals Jacobians, easy to compute  $H_K$ 
  - Also very efficient - just matrix multiplication

# How to compute Jacobian?

- If we have individual Jacobians, easy to compute  $H_K$ 
  - Also very efficient - just matrix multiplication
- How to get individual Jacobians?
  - Some are easy: For  $\mathcal{J}^{w,K}, \mathcal{J}^{r^k,K}$  we just have to diff.  
 $r_t^K = \alpha \Gamma_t(K_{t-1}/L_t)^{\alpha-1}, w_t = (1 - \alpha) \Gamma_t(K_{t-1}/L_t)^\alpha$ 
    - Cheap, and can often be vectorized
  - What about HH Jacobians  $\mathcal{J}^{A_{hh},r}, \mathcal{J}^{A_{hh},w}$ ?

# Bottleneck: How do we find the Jacobian?

- **Naive approach:** For each input  $i$  into HH block  $i \in \{r, w\}$ 
  - For each  $s \in \{0, 1, \dots, T - 1\}$ 
    1. Shock input  $i$  in period  $s$  by small amount  $\Delta$
    2. Solve household problem backwards along transition path
    3. Simulate households forward along transition path
    4. Calculate column  $s$ , row  $t$  of jacobian as  $\frac{\partial \mathcal{J}_t^{A_{hh}, i}}{\partial i_s} = \frac{A_t^{hh} - A_{ss}^{hh}}{\Delta}$  for all  $t$

**Bottleneck:** We need  $T^2$  solution steps and simulation steps for each input  $\{r, w\}$ !

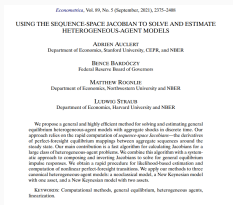


# Bottleneck: How do we find the Jacobian?

- **Naive approach:** For each input  $i$  into HH block  $i \in \{r, w\}$ 
  - For each  $s \in \{0, 1, \dots, T-1\}$ 
    1. Shock input  $i$  in period  $s$  by small amount  $\Delta$
    2. Solve household problem backwards along transition path
    3. Simulate households forward along transition path
    4. Calculate column  $s$ , row  $t$  of jacobian as  $\frac{\partial \mathcal{J}_t^{A_{hh}, i}}{\partial i_s} = \frac{A_t^{hh} - A_{ss}^{hh}}{\Delta}$  for all  $t$

**Bottleneck:** We need  $T^2$  solution steps and simulation steps for each input  $\{r, w\}$ !

- **Solution: Fake news algorithm** - only need  $T$  steps! (later today)



# Summary

- Conditional on being able to compute HH jacobian efficiently we can compute **transition path** through following steps:
  1. Compute stationary state of model
  2. Formulate transition path as DAG
    - Reduce number of unknowns and residual equations
    - Not essential, but often good idea
  3. Compute Jacobian of residuals  $H$  w.r.t unknowns  $K$
  4. Formulate shock (i.e. TFP increases by 1% for 4 years)
  5. Use Broyden's method to solve for transition path

# Assumptions and interpretation

- **Underlying assumption:** No aggregate uncertainty

# Assumptions and interpretation

- **Underlying assumption:** No aggregate uncertainty
- **»Shock«,  $\Gamma$ :** A fully unexpected non-recurrent event  $\equiv$  *MIT shock*
  - Unexpected before occurring at time 0
  - From time 0 and onwards agents have perfect foresight w.r.t transition dynamics

# Assumptions and interpretation

- **Underlying assumption:** No aggregate uncertainty
- **»Shock«,  $\Gamma$ :** A fully unexpected non-recurrent event  $\equiv$  *MIT shock*
  - Unexpected before occurring at time 0
  - From time 0 and onwards agents have perfect foresight w.r.t transition dynamics
- **Transition path,  $K$ :** Non-linear perfect foresight response to
  1. Initial distribution,  $\underline{D}_0 \neq D_{ss}$  or  $K_0 \neq K_{ss}$  (convergence to steady state)
  2. Shock,  $\Gamma_t \neq \Gamma_{ss}$  for some  $t$  (i.e. impulse-response)

# The HANC example from GEModelToolsNotebooks

- **Presentation:** I go through the code

# Interpreting the household Jacobians

- **Jacobian of consumption wrt. wage:** *What happens to consumption in period  $t$  when the wage (and thus income) increases in period  $s$ ?*

$$\mathcal{J}^{C^{hh}, w} = \begin{bmatrix} \frac{\partial C_0^{hh}}{\partial w_0} & \frac{\partial C_0^{hh}}{\partial w_1} & \dots \\ \frac{\partial C_1^{hh}}{\partial w_0} & \ddots & \ddots \\ \vdots & \ddots & \ddots \end{bmatrix}$$

- **Columns:** The full dynamic response to a unit shock in period  $s$

# Decomposition of GE response

- GE transition path:  $\mathbf{r}^*$  and  $\mathbf{w}^*$
- PE response of each:
  1. Set  $(\mathbf{r}, \mathbf{w}) \in \{(\mathbf{r}^*, \mathbf{w}_{ss}), (\mathbf{r}_{ss}, \mathbf{w}^*)\}$
  2. Solve household problem backwards along transition path
  3. Simulate households forward along transition path
  4. Calculate outcomes of interest



# Fake News Algorithm

---

- Household block:

$$\mathbf{Y}^{hh} = hh(\mathbf{X}^{hh})$$

- i.e.  $\mathbf{Y}^{hh} = C^{hh}, A^{hh}$  and  $\mathbf{X}^{hh} = w, r$

# Fake news algorithm

- Household block:

$$\mathbf{Y}^{hh} = hh(\mathbf{X}^{hh})$$

- i.e.  $\mathbf{Y}^{hh} = C^{hh}, A^{hh}$  and  $\mathbf{X}^{hh} = w, r$
- **Goal:** Fast computation of

$$\mathcal{J}^{hh} = \frac{dhh(\mathbf{X}_{ss}^{hh})}{d\mathbf{X}^{hh}}$$

- **Household block:**

$$\mathbf{Y}^{hh} = hh(\mathbf{X}^{hh})$$

- i.e.  $\mathbf{Y}^{hh} = C^{hh}, A^{hh}$  and  $\mathbf{X}^{hh} = w, r$

- **Goal:** Fast computation of

$$\mathcal{J}^{hh} = \frac{dhh(\mathbf{X}_{ss}^{hh})}{d\mathbf{X}^{hh}}$$

- **Naive approach:**

- Shock at time  $s = 0$ , solve + simulate HH block for  $T$  periods
- Repeat until  $s = T - 1$
- Requires  $T^2$  solution and simulation steps

# Fake news algorithm

- **Household block:**

$$\mathbf{Y}^{hh} = hh(\mathbf{X}^{hh})$$

- i.e.  $\mathbf{Y}^{hh} = C^{hh}, A^{hh}$  and  $\mathbf{X}^{hh} = w, r$

- **Goal:** Fast computation of

$$\mathcal{J}^{hh} = \frac{dhh(\mathbf{X}_{ss}^{hh})}{d\mathbf{X}^{hh}}$$

- **Naive approach:**

- Shock at time  $s = 0$ , solve + simulate HH block for  $T$  periods
- Repeat until  $s = T - 1$
- Requires  $T^2$  solution and simulation steps

- **Next slides:** *Sketch of much faster approach*

# Initial step

- Note that aggregate is (matrix) product of individual policy function  $\mathbf{y}_t$  and distribution  $\mathbf{D}_t$ .
- Linearize (first-order Taylor) around ss:

$$\begin{aligned}\mathbf{Y}^{hh} &= (\mathbf{y}'_t) \mathbf{D}_t \\ \Rightarrow \frac{d\mathbf{Y}^{hh}}{d\mathbf{X}^{hh}} &= \left( \frac{d\mathbf{y}'_t}{d\mathbf{X}^{hh}} \right) \mathbf{D}_{ss} + (\mathbf{y}'_{ss}) \frac{d\mathbf{D}_t}{d\mathbf{X}^{hh}}\end{aligned}$$

- What can we say about policy function term  $d\mathbf{y}_t$ ?

# Perturbation of policy function

- The heart of the fake news algorithm is a central insight that allow us to compute  $d\mathbf{y}_t/d\mathbf{X}^{hh}$  efficiently

# Perturbation of policy function

- The heart of the fake news algorithm is a central insight that allow us to compute  $d\mathbf{y}_t/d\mathbf{X}^{hh}$  efficiently
- Let  $\mathbf{y}_t^s$  be policy function at time  $t$  following a shock in period  $s$ .  
Then:

$$\mathbf{y}_t^s = \begin{cases} \mathbf{y}_{ss} & t > s \\ \mathbf{y}_{t+j}^{s+j} & t \leq s \end{cases}$$



# Perturbation of policy function

- The heart of the fake news algorithm is a central insight that allow us to compute  $d\mathbf{y}_t/d\mathbf{X}^{hh}$  efficiently
- Let  $\mathbf{y}_t^s$  be policy function at time  $t$  following a shock in period  $s$ .  
Then:

$$\mathbf{y}_t^s = \begin{cases} \mathbf{y}_{ss} & t > s \\ \mathbf{y}_{t+j}^{s+j} & t \leq s \end{cases}$$

- Verbally: The response of the policy function  $\mathbf{y}$  at time  $t$  to a shock at  $s$  is the as the response at time  $t + j$  to a shock at  $s + j$ 
  - Policy function **does not depend on the absolut time of shock** only the relative distance between »today« and the shock,  $s - t$ .

# Perturbation of policy function

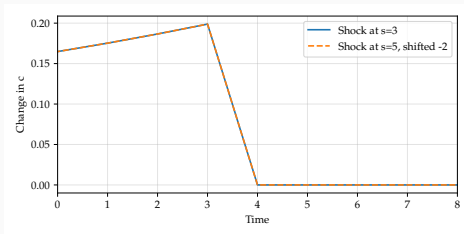
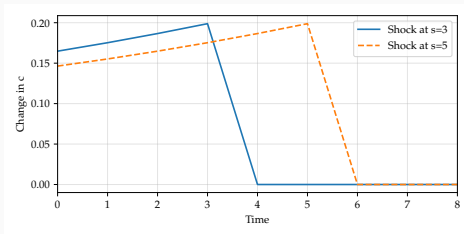
- The heart of the fake news algorithm is a central insight that allow us to compute  $d\mathbf{y}_t/d\mathbf{X}^{hh}$  efficiently
- Let  $\mathbf{y}_t^s$  be policy function at time  $t$  following a shock in period  $s$ . Then:

$$\mathbf{y}_t^s = \begin{cases} \mathbf{y}_{ss} & t > s \\ \mathbf{y}_{t+j}^{s+j} & t \leq s \end{cases}$$

- Verbally: The response of the policy function  $\mathbf{y}$  at time  $t$  to a shock at  $s$  is the as the response at time  $t+j$  to a shock at  $s+j$ 
  - Policy function **does not depend on the absolut time of shock** only the relative distance between »today« and the shock,  $s - t$ .
- **Implication:** We need to only do a single backwards iteration to a shock at  $s = T - 1$ .
  - Can then construct change in policy function  $d\mathbf{y}_t^s/d\mathbf{X}^{hh}$  for different  $s$  by shifting policy function around

# Numerical illustration

Graphically. Response of  $c_t$  to income shock at  $s = 3, 5$



# Aggregate Jacobian

- Can we use same logic for aggregate Jacobian,  $\mathcal{J}_{t,s} = \mathcal{J}_{t-1,s-1}$ ?
  - No - the above is true for *policy* function, but not **distribution**
  - Distribution is backwards looking ( $\mathbf{D}_t^s = (\mathbf{\Lambda}_t^s \Pi_{ss})' \mathbf{D}_{t-1}^s$ ) so number of periods  $t$  since announcement matters

# Aggregate Jacobian

- Can we use same logic for aggregate Jacobian,  $\mathcal{J}_{t,s} = \mathcal{J}_{t-1,s-1}$ ?
  - No - the above is true for *policy* function, but not **distribution**
  - Distribution is backwards looking ( $\mathbf{D}_t^s = (\mathbf{\Lambda}_t^s \Pi_{ss})' \mathbf{D}_{t-1}^s$ ) so number of periods  $t$  since announcement matters
- Can write aggregate Jacobian as:

$$\mathcal{J}_{t,s} = \begin{cases} \mathcal{F}_{t,s} & \text{for } t = 0, s = 0 \\ \mathcal{J}_{t-1,s-1} + \mathcal{F}_{t,s} & \text{for } t, s > 0 \end{cases}$$

# Aggregate Jacobian

- Can we use same logic for aggregate Jacobian,  $\mathcal{J}_{t,s} = \mathcal{J}_{t-1,s-1}$ ?
  - No - the above is true for *policy* function, but not **distribution**
  - Distribution is backwards looking ( $\mathbf{D}_t^s = (\mathbf{\Lambda}_t^s \Pi_{ss})' \mathbf{D}_{t-1}^s$ ) so number of periods  $t$  since announcement matters
- Can write aggregate Jacobian as:

$$\mathcal{J}_{t,s} = \begin{cases} \mathcal{F}_{t,s} & \text{for } t = 0, s = 0 \\ \mathcal{J}_{t-1,s-1} + \mathcal{F}_{t,s} & \text{for } t, s > 0 \end{cases}$$

- where  $\mathcal{F}_{t,s}$  is the **fake news** matrix.

# Aggregate Jacobian

- Can we use same logic for aggregate Jacobian,  $\mathcal{J}_{t,s} = \mathcal{J}_{t-1,s-1}$ ?
  - No - the above is true for *policy* function, but not **distribution**
  - Distribution is backwards looking ( $\mathbf{D}_t^s = (\boldsymbol{\Lambda}_t^s \Pi_{ss})' \mathbf{D}_{t-1}^s$ ) so number of periods  $t$  since announcement matters
- Can write aggregate Jacobian as:

$$\mathcal{J}_{t,s} = \begin{cases} \mathcal{F}_{t,s} & \text{for } t = 0, s = 0 \\ \mathcal{J}_{t-1,s-1} + \mathcal{F}_{t,s} & \text{for } t, s > 0 \end{cases}$$

- where  $\mathcal{F}_{t,s}$  is the **fake news** matrix.
- Element  $(t, s)$  in matrix  $\mathcal{F}$  for  $t > 0$  is

$$\mathcal{F}_{t,s} = (\mathbf{y}_{ss})' (\boldsymbol{\Lambda}'_{ss})^t \frac{d\mathbf{D}_1^s}{d\mathbf{X}^{hh}}$$

# Aggregate Jacobian

- Can we use same logic for aggregate Jacobian,  $\mathcal{J}_{t,s} = \mathcal{J}_{t-1,s-1}$ ?
  - No - the above is true for *policy* function, but not **distribution**
  - Distribution is backwards looking ( $\mathbf{D}_t^s = (\mathbf{\Lambda}_t^s \Pi_{ss})' \mathbf{D}_{t-1}^s$ ) so number of periods  $t$  since announcement matters
- Can write aggregate Jacobian as:

$$\mathcal{J}_{t,s} = \begin{cases} \mathcal{F}_{t,s} & \text{for } t = 0, s = 0 \\ \mathcal{J}_{t-1,s-1} + \mathcal{F}_{t,s} & \text{for } t, s > 0 \end{cases}$$

- where  $\mathcal{F}_{t,s}$  is the **fake news** matrix.
- Element  $(t, s)$  in matrix  $\mathcal{F}$  for  $t > 0$  is

$$\mathcal{F}_{t,s} = (\mathbf{y}_{ss})' (\mathbf{\Lambda}'_{ss})^t \frac{d\mathbf{D}_1^s}{d\mathbf{X}^{hh}}$$

- Why »fake news«?  $\mathcal{F}_{t,s}$  captures effect of announcing a date— $s$  shock at time 0, and retracting the announcement at date 1
  - Policy variables revert to steady state after period 1, but distribution changes since  $d\mathbf{y}_0 \neq 0$



- Can show that the fake news matrix can be computed as:

$$\mathcal{F}_{t,s} \equiv \begin{cases} \left( \frac{dy_0^s}{d\mathbf{X}^{hh}} \right)' \mathbf{D}_{ss} & t = 0 \\ (\mathbf{y}_{ss})' (\boldsymbol{\Lambda}'_{ss})^t \frac{d\mathbf{D}_1^s}{d\mathbf{X}^{hh}} & t > 0 \end{cases}$$

- Can show that the fake news matrix can be computed as:

$$\mathcal{F}_{t,s} \equiv \begin{cases} \left( \frac{d\mathbf{y}_0^s}{d\mathbf{X}^{hh}} \right)' \mathbf{D}_{ss} & t = 0 \\ (\mathbf{y}_{ss})' (\boldsymbol{\Lambda}'_{ss})^t \frac{d\mathbf{D}_1^s}{d\mathbf{X}^{hh}} & t > 0 \end{cases}$$

- $t = 0$  element: Easy to compute when we have  $d\mathbf{y}_0^s/d\mathbf{X}^{hh}$ 
  - Can get this from a single backwards run ( $T$  periods) due to logic from before

- Can show that the fake news matrix can be computed as:

$$\mathcal{F}_{t,s} \equiv \begin{cases} \left( \frac{d\mathbf{y}_0^s}{d\mathbf{X}^{hh}} \right)' \mathbf{D}_{ss} & t = 0 \\ (\mathbf{y}_{ss})' (\boldsymbol{\Lambda}'_{ss})^t \frac{d\mathbf{D}_1^s}{d\mathbf{X}^{hh}} & t > 0 \end{cases}$$

- $t = 0$  element: Easy to compute when we have  $d\mathbf{y}_0^s/d\mathbf{X}^{hh}$ 
  - Can get this from a single backwards run ( $T$  periods) due to logic from before
- $t > 0$  elements: Only involves basic matrix multiplication once we have  $d\mathbf{D}_1^s/d\mathbf{X}^{hh}$ 
  - Since we have derivatives of policy function for all  $t, s$   $d\mathbf{y}_t^s/d\mathbf{X}^{hh}$  can get  $d\mathbf{D}_1^s/d\mathbf{X}^{hh}$  easily
  - Note: Not too expensive since histogram method for distribution is fast and efficient

## Fake news algorithm - summary

- Auclert et. al (2021) introduce an efficient algorithm to compute aggregate jacobians for models with heterogeneous agents
  - Can compute the linearized response of aggregate consumption, savings w.r.t aggregate variables such as wages, interest rates **fast**

## Fake news algorithm - summary

- Auclert et. al (2021) introduce an efficient algorithm to compute aggregate jacobians for models with heterogeneous agents
  - Can compute the linearized response of aggregate consumption, savings w.r.t aggregate variables such as wages, interest rates **fast**
- Central insight: Exploit structure of dynamic programming problems + histogram method

# Fake news algorithm - summary

- Auclert et. al (2021) introduce an efficient algorithm to compute aggregate jacobians for models with heterogeneous agents
  - Can compute the linearized response of aggregate consumption, savings w.r.t aggregate variables such as wages, interest rates **fast**
- Central insight: Exploit structure of dynamic programming problems + histogram method
- Why did we need this?
  - Allows us to compute Jacobian of aggregate model by »chaining« together individual jacobians along DAG
  - Can then use Quasi-Newton methods to solve dynamic GE model!

# Fake news algorithm - summary

- Auclert et. al (2021) introduce an efficient algorithm to compute aggregate jacobians for models with heterogeneous agents
  - Can compute the linearized response of aggregate consumption, savings w.r.t aggregate variables such as wages, interest rates **fast**
- Central insight: Exploit structure of dynamic programming problems + histogram method
- Why did we need this?
  - Allows us to compute Jacobian of aggregate model by »chaining« together individual jacobians along DAG
  - Can then use Quasi-Newton methods to solve dynamic GE model!
- GEModeltools does all of this »under the hood« when you compute HH Jacobians
  - You just tell GEModeltools the inputs and outputs of the household block
  - Entire algorithm is automated

# Exercises

---



## Exercises: HANCGovModel

Same model. Your choice of  $\tau_{ss}$ . New questions:

1. **Define the transition path.**
2. **Plot the DAG**
3. **What do the Jacobians look like?**
4. **Find the transition path for  $G_t = G_{ss} + 0.01G_{ss}0.95^t$**
5. **What explains household savings behavior?**
6. **What happens to consumption inequality?**

## Summary

---

# Summary and next week

- **Today:**
  1. The concept of a transition path
  2. Details of the **GEModelTools** package
- **Homework:** Work on completing the model extension exercise
- **Next week:** Begin working on Assignment 1