4. Transition Path

Adv. Macro: Heterogenous Agent Models

Nicolai Waldstrøm

2024

Introduction

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

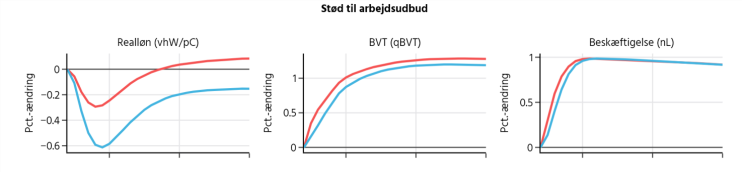
- **Last time:** *Stationary equilibrium (steady states)*
- **Today:** *Transition path (dynamic responses away from steady state)*
- **Model:** Heterogeneous Agent Neo-Classical (HANC) model
- **Code:**
  1. Based on the GEModelTools package
  2. Example from GEModelToolsNotebooks/HANC (except stuff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

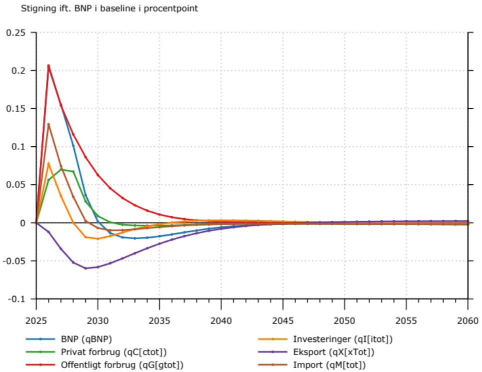


- Note: Permanent shock, so transition path *between* two different steady states

Example II

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

Figure 6.0.1: BNP og underkomponenter



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

Standard Ramsey model

Ramsey: Summary

- **Simplified form:**

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

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

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

Ramsey: As an equation system

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

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:

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

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):

$$f\left(x\right)=0$$

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

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

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

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

Recap: Newton's method II

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

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

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

Recap: Newton's method II

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

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

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

Recap: Newton's method II

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

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

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

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

  - 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:

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

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

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

  - 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 basis of attraction

Recap: Multivariate Newton's method

- Generalize to vector-valued, multivariate functions $[f_1(x_{1,}x_2), f_2(x_{1,}x_2)]' = \boldsymbol{f}(\boldsymbol{x})$ with $\boldsymbol{x} = (x_{1,}x_2)'$:

$$\boldsymbol{x}^{i+1} = \boldsymbol{x}^i - \boldsymbol{J}\left(\boldsymbol{x}^i\right)^{-1} \boldsymbol{f}\left(\boldsymbol{x}^i\right)$$

Recap: Multivariate Newton's method

- Generalize to vector-valued, multivariate functions $\left[f_1\left(x_{1,}x_2\right), f_2\left(x_{1,}x_2\right)\right]' = \boldsymbol{f}\left(\boldsymbol{x}\right)$ with $\boldsymbol{x} = \left(x_{1,}x_2\right)'$:

$$\boldsymbol{x}^{i+1} = \boldsymbol{x}^i - \boldsymbol{J}\left(\boldsymbol{x}^i\right)^{-1} \boldsymbol{f}\left(\boldsymbol{x}^i\right)$$

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

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

Recap: Multivariate Newton's method

- Generalize to vector-valued, multivariate functions $\left[f_1\left(x_{1,} x_2\right), f_2\left(x_{1,} x_2\right)\right]' = \boldsymbol{f}\left(\boldsymbol{x}\right)$ with $\boldsymbol{x} = \left(x_{1,} x_2\right)'$:

$$\boldsymbol{x}^{i+1} = \boldsymbol{x}^i - \boldsymbol{J}\left(\boldsymbol{x}^i\right)^{-1} \boldsymbol{f}\left(\boldsymbol{x}^i\right)$$

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

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

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

Recap: Multivariate Newton's method

- Generalize to vector-valued, multivariate functions $[f_1(x_{1,}x_2), f_2(x_{1,}x_2)]' = \boldsymbol{f}(\boldsymbol{x})$ with $\boldsymbol{x} = (x_{1,}x_2)'$:

$$\boldsymbol{x}^{i+1} = \boldsymbol{x}^i - \boldsymbol{J}\left(\boldsymbol{x}^i\right)^{-1} \boldsymbol{f}\left(\boldsymbol{x}^i\right)$$

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

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

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

Recap: Broyden's method I

- Newton's method updates Jacobian $\boldsymbol{J}$ in **every iteration**

Recap: Broyden's method I

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

Recap: Broyden's method I

- Newton's method updates Jacobian $\boldsymbol{J}$ in **every iteration**
- If $\boldsymbol{J}$ 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 $\boldsymbol{J}$ in **every iteration**
- If $\boldsymbol{J}$ 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 $f'\left(x^{i+1}\right)$ at every iteration i:

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

Recap: Broyden's method II

1. Guess $\boldsymbol{x}^0$ and set $i = 0$
2. Calculate the Jacobian around initial point $\boldsymbol{J_0}$
3. Calculate $\boldsymbol{f}^i = \boldsymbol{f}(\boldsymbol{x}^i)$.
4. Stop if $\left|\boldsymbol{f}^i\right|$ below tolerance ϵ
5. Calculate Jacobian by
$$\boldsymbol{J}^i = \begin{cases} \boldsymbol{J_0} & \text{if } i = 0 \\ \boldsymbol{J}^{i-1} + \frac{(\boldsymbol{f}^i - \boldsymbol{f}^{i-1}) - \boldsymbol{J}^{i-1}(\boldsymbol{x}^i - \boldsymbol{x}^{i-1})}{\left|\boldsymbol{x}^i - \boldsymbol{x}^{i-1}\right|_2} \left(\boldsymbol{x}^i - \boldsymbol{x}^{i-1}\right)' & \text{if } i > 0 \end{cases}$$
6. Update guess by $\boldsymbol{x}^{i+1} = \boldsymbol{x}^i - \left(\boldsymbol{J}^i\right)^{-1} \boldsymbol{f}^i$
7. Increment i and return to step 3

- **Go through code**

Back to Ramsey

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

- 2 issues:

Back to Ramsey

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

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

Back to Ramsey

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

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

Truncated Ramsey, reduced vector form

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

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

$$\begin{aligned}
r_t^K &= \alpha \Gamma_t (K_{t-1}/L_t)^{\alpha-1} \\
w_t &= (1-\alpha)\Gamma_t (K_{t-1}/L_t)^{\alpha} \\
A_t &= K_t \\
r_t &= r_t^K - \delta \\
C_t^{hh} &= (\beta(1+r_{t+1}))^{-\sigma} \, C_{t+1}^{hh} \text{ (backwards)} \\
L_t^{hh} &= 1 \\
A_t^{hh} &= (1+r_t)A_{t-1}^{hh} + w_t L_t^{hh} - C_t^{hh} \text{ (forwards)}
\end{aligned}$$

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

Further reduced

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

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

$$\begin{aligned}
L_t = L_t^{hh} &= 1 \\
r_t^K &= \alpha \Gamma_t (K_{t-1}/L_t)^{\alpha-1} \\
w_t &= (1-\alpha)\Gamma_t (K_{t-1}/L_t)^{\alpha} \\
A_t &= K_t \\
r_t &= r_t^K - \delta \\
C_t^{hh} &= (\beta(1+r_{t+1}))^{-\sigma} \, C_{t+1}^{hh} \text{ (backwards)} \\
A_t^{hh} &= (1+r_t)A_{t-1}^{hh} + w_t L_t^{hh} - C_t^{hh} \text{ (forwards)}
\end{aligned}$$

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

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 $C_t = a + bY_t$:

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

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 $C_t = a + bY_t$:

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

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

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

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 $C_t = a + bY_t$:

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

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

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

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

Solution in sequence space

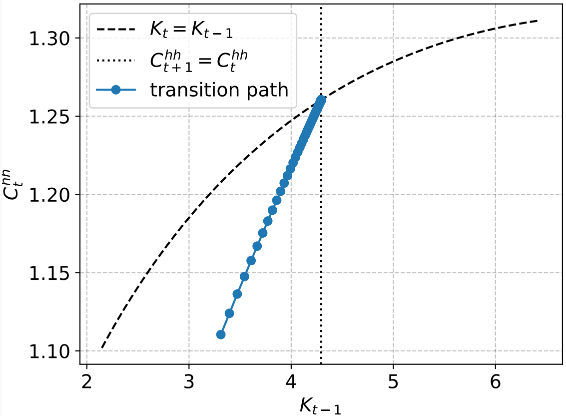
- **Truncation:** $T = 200$ (transition path should have converged to ss by then)
- **Jacobian:** Find $\boldsymbol{H}_{\boldsymbol{K}}$ by *numerical differentiation*

$$\boldsymbol{H}_{\boldsymbol{K}} = \begin{bmatrix} \frac{\partial(A_0 - A_0^{hh})}{\partial K_0} & \frac{\partial(A_0 - A_0^{hh})}{\partial K_1} & \cdots \\ \frac{\partial(A_1 - A_1^{hh})}{\partial K_0} & \ddots & \ddots \\ \vdots & \ddots & \ddots \end{bmatrix}$$

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

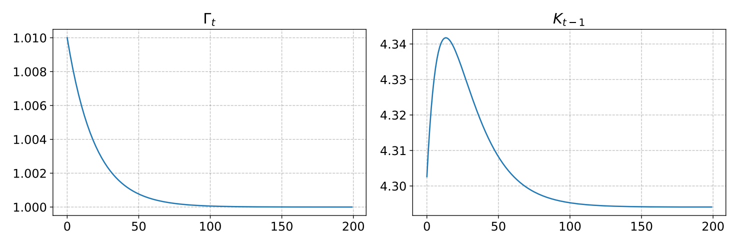
Example 1: Initially low capital

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



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**

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

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

Remember: Policies and choice transitions depend on prices

1. Policy function: $x_t^* = x^*\left(\{r_\tau, w_\tau\}_{\tau\geq t}\right)$ and $X_t^{hh} = \sum_i x_{it}^* D_{it} = \boldsymbol{x}_t^{*\prime} \boldsymbol{D}_t$
2. Choice transition: $\Lambda_t = \Lambda\left(\{r_\tau, w_\tau\}_{\tau\geq t}\right)$

Transition path - close to verbal definition

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

1. Quantities $\{K_t\}$ and $\{L_t\}$,
2. prices $\{r_t\}$ and $\{w_t\}$,
3. the distributions $\{\boldsymbol{D}_t\}$ over β_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. $\boldsymbol{D}_t$ is implied by simulating the household problem forwards from $\underline{\boldsymbol{D}}_0$
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 $\boldsymbol{J}$ 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 $\boldsymbol{J}$ has size $(T \times N, T \times N,)$
  - $\Rightarrow$ Ekspensive 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

$$\boldsymbol{H}(\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{\Gamma}, \underline{\boldsymbol{D}}_0) = \left[\begin{array}{c} A_t - A_t^{hh} \\ L_t - L_t^{hh} \\ \forall t \in \{0, 1, \ldots, T-1\} \end{array} \right] = \boldsymbol{0}$$

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

$$\begin{aligned} r_t^K &= \alpha \Gamma_t (K_{t-1}/L_t)^{\alpha-1} \\ w_t &= (1-\alpha)\Gamma_t (K_{t-1}/L_t)^{\alpha} \\ r_t &= r_t^K - \delta \\ A_t &= K_t \\ \boldsymbol{D}_t &= \Pi_z' \underline{\boldsymbol{D}}_t \\ \underline{\boldsymbol{D}}_{t+1} &= \Lambda_t' \boldsymbol{D}_t \\ A_t^{hh} &= \boldsymbol{a}_t^{*\prime} \boldsymbol{D}_t \\ L_t^{hh} &= \boldsymbol{\ell}_t^{*\prime} \boldsymbol{D}_t \\ &\forall t \in \{0, 1, \ldots, T-1\} \end{aligned}$$

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 ⇒ Firm block ⇒ Mutual fund ⇒HHs ⇒ Residuals

Further reduction

$$\boldsymbol{H}\left(\boldsymbol{K}, \boldsymbol{\Gamma}, \underline{\boldsymbol{D}}_0\right) = \left[\begin{array}{c} A_t - A_t^{hh} \\ \forall t \in \{0, 1, \dots, T-1\} \end{array}\right] = \boldsymbol{0}$$

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

$$\begin{aligned}
L_t &= 1 \\
r_t^K &= \alpha \Gamma_t (K_{t-1}/L_t)^{\alpha-1} \\
w_t &= (1-\alpha)\Gamma_t (K_{t-1}/L_t)^{\alpha} \\
A_t &= K_t \\
r_t &= r_t^K - \delta \\
\boldsymbol{D}_t &= \Pi_z' \underline{\boldsymbol{D}}_t \\
\underline{\boldsymbol{D}}_{t+1} &= \Lambda_t' \boldsymbol{D}_t \\
A_t^{hh} &= \boldsymbol{a}_t^{*\prime} \boldsymbol{D}_t \\
&\forall t \in \{0, 1, \dots, T-1\}
\end{aligned}$$

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 ($\boldsymbol{H}$)
  - And T unknowns ($\boldsymbol{K}$)
- If we can calculate the jacobian of $\boldsymbol{H}$ w.r.t $\boldsymbol{K}$ we can solve with Broyden's method as before

How to compute Jacobian?

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

$$\boldsymbol{w}, \boldsymbol{r}^K = Firm(\boldsymbol{K}), \quad \boldsymbol{A}, \boldsymbol{r} = MutFund\left(\boldsymbol{K}, \boldsymbol{r}^K\right)$$

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

How to compute Jacobian?

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

$$\boldsymbol{w}, \boldsymbol{r}^K = Firm\left(\boldsymbol{K}\right), \quad \boldsymbol{A}, \boldsymbol{r} = MutFund\left(\boldsymbol{K}, \boldsymbol{r}^K\right)$$

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

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

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

How to compute Jacobian?

- If we have individuals Jacobians, easy to compute $\boldsymbol{H_K}$
  - Also very efficient - just matrix mulitiplication

How to compute Jacobian?

- If we have individuals Jacobians, easy to compute $\boldsymbol{H_K}$
  - Also very efficient - just matrix mulitiplication

- 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 $s \in \{0, 1, ..., T-1\}$
    1. Shock input i in period s by small amount Δ
    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

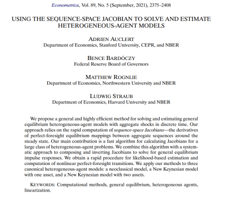
Bottleneck: We need T^2 solution steps and simulation steps for each input $\{r, w\}$!

Bottleneck: How do we find the Jacobian?

- **Naive approach:** For each input i into HH block $i \in \{r, w\}$
  - For each $s \in \{0, 1, ..., T-1\}$
    1. Shock input i in period s by small amount Δ
    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)



Econometrica, Vol. 89, No. 5 (September, 2021), 2375–2408

USING THE SEQUENCE-SPACE JACOBIAN TO SOLVE AND ESTIMATE HETEROGENEOUS-AGENT MODELS

ADRIEN AUCLERT
Department of Economics, Stanford University, CEPR, and NBER

BENCE BARDÓCZY
Federal Reserve Board of Governors

MATTHEW ROGNLIE
Department of Economics, Northwestern University and NBER

LUDWIG STRAUB
Department of Economics, Harvard University and NBER

We propose a general and highly efficient method for solving and estimating general equilibrium heterogeneous-agent models with aggregate shocks in discrete time. Our approach relies on the rapid computation of *sequence-space Jacobians*—the derivatives of perfect-foresight equilibrium mappings between aggregate sequences around the steady state. Our main contribution is a fast algorithm for calculating Jacobians for a large class of heterogeneous-agent problems. We combine this algorithm with a systematic approach to composing and inverting Jacobians to solve for general equilibrium impulse responses. We obtain a rapid procedure for likelihood-based estimation and computation of nonlinear perfect-foresight transitions. We apply our methods to three canonical heterogeneous-agent models: a neoclassical model, a New Keynesian model with one asset, and a New Keynesian model with two assets.

KEYWORDS: Computational methods, general equilibrium, heterogeneous agents, linearization.

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 $\boldsymbol{H}$ w.r.t unknowns $\boldsymbol{K}$
  4. Formulate shock (i.e. TFP increases by 1% for 4 years)
  5. Use Broyden's method to solve for transition path

Assumptons and interpretation

- **Underlying assumption:** No aggregate uncertainty

Assumptons and interpretation

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

Assumptons and interpretation

- **Underlying assumption:** No aggregate uncertainty
- **»Shock«, Γ:** A fully unexpected non-recurrent event $\equiv$ *MIT shock*
    - Unexpected before occuring at time 0
    - From time 0 and onwards agents have perfect foresight w.r.t transition dynamics
- **Transition path, $\boldsymbol{K}$:** Non-linear perfect foresight response to
    1. Initial distribution, $\underline{\boldsymbol{D}}_0 \neq \boldsymbol{D}_{ss}$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?*

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

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

Decomposition of GE response

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

Fake News Algorithm

Fake news algorithm

- **Household block:**

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

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

Fake news algorithm

- **Household block:**

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

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

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

Fake news algorithm

- **Household block:**

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

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

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

- **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:**

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

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

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

- **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 $\boldsymbol{y}_t$ and distribution $\boldsymbol{D}_t$.
- Linearize (first-order Taylor) around ss:

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

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

Pertubation of policy function

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

Pertubation of policy function

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

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

Pertubation of policy function

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

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

- Verbally: The response of the policy function $\boldsymbol{y}$ 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, $s-t$.

Pertubation of policy function

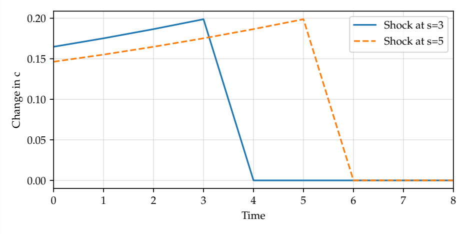
- The heart of the fake news algorithm is a central insight that allow us to compute $d\boldsymbol{y}_t/d\boldsymbol{X}^{hh}$ efficiently
- Let $\boldsymbol{y}_t^s$ be policy function at time t following a shock in period s. Then:

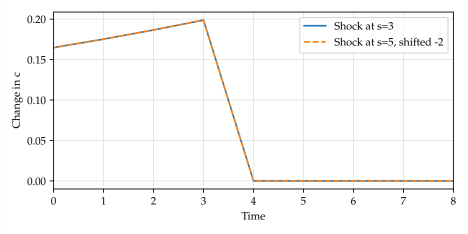
$$\boldsymbol{y}_t^s = \begin{cases} \boldsymbol{y}_{ss} & t > s \\ \boldsymbol{y}_{t+j}^{s+j} & t \leq s \end{cases}$$

- Verbally: The response of the policy function $\boldsymbol{y}$ 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 $d\boldsymbol{y}_t^s/d\boldsymbol{X}^{hh}$ 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 aggreregate 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 ($\boldsymbol{D}_t^s = (\boldsymbol{\Lambda}_t^s \Pi_{ss})' \, \boldsymbol{D}_{t-1}^s$) so number of periods t since announcement matters

Aggregate Jacobian

- Can we use same logic for aggreregate 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 ($\boldsymbol{D}_t^s = \left(\boldsymbol{\Lambda}_t^s \Pi_{ss}\right)' \boldsymbol{D}_{t-1}^s$) 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 aggreregate 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 ($\boldsymbol{D}_t^s = (\boldsymbol{\Lambda}_t^s \Pi_{ss})' \boldsymbol{D}_{t-1}^s$) 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 aggreregate 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 ($\boldsymbol{D}_t^s = \left(\boldsymbol{\Lambda}_t^s \Pi_{ss}\right)' \boldsymbol{D}_{t-1}^s$) 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

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

Aggregate Jacobian

- Can we use same logic for aggreregate 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 ($\boldsymbol{D}_t^s = (\boldsymbol{\Lambda}_t^s \Pi_{ss})' \boldsymbol{D}_{t-1}^s$) 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

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

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

Fake News Matrix

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

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

Fake News Matrix

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

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

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

Fake News Matrix

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

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

- $t = 0$ element: Easy to compute when we have $d\boldsymbol{y}_0^s/d\boldsymbol{X}^{hh}$
  - 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 $d\boldsymbol{D}_1^s/d\boldsymbol{X}^{hh}$
  - Since we have derivatves of policy function for all t, s $d\boldsymbol{y}_t^s/d\boldsymbol{X}^{hh}$ can get $d\boldsymbol{D}_1^s/d\boldsymbol{X}^{hh}$ 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 alogirthm 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 alogirthm 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 alogirthm 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 alogirthm 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 τ_{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** $G_t = G_{ss} + 0.01 G_{ss} 0.95^t$
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