The `GEModelTools` Package v0.1

Jeppe Druedahl
Emil Holst Partsch

Abstract

This note provides an overview of the `GEModelTools` package for solving general equilibrium models easily in Python using the sequence-space method.

You can learn how to use the package following these steps:

1. Read this document
2. Install the package
3. Run the example notebooks
4. Read the commented code for the example notebooks
5. Implement your own model

Course: Adv. Macro: Heterogenous Agent Models
Literature: Boppart et al. (2018) and Auclert et al. (2021).

Structure:

Section 1 describes the class of models considered

Section 2 explains the required user inputs and available methods

Section 3 explains how to efficiently compute the household Jacobian

Section 4 explains additional features

Section 5 provides basic troubleshooting

Code:

Package: github.com/NumEconCopenhagen/GEModelTools
Notebooks: github.com/NumEconCopenhagen/GEModelToolsNotebooks

Requirements: Rely on `EconModel` and `ConSav`.

Packages:
github.com/NumEconCopenhagen/EconModel
github.com/NumEconCopenhagen/ConsumptionSaving

Notebooks:
github.com/NumEconCopenhagen/EconModelNotebooks
github.com/NumEconCopenhagen/ConsumptionSavingNotebooks

1 Model class

In this section, we describe the class of general equilibrium models with heterogeneous agents the package is designed solve, and explain how to use the *sequence space method* developed in Auclert et al. (2021) to solve them. The starting point is a model with *perfect foresight*, where the *non-linear transition path* can be found given the initial distribution of agents and a sequence of exogenous shocks. Next, we show how to solve for the *linearized impulse responses.* These impulse responses are equal to those from a model with *aggregate risk* once it is linearized and certainty equivalence holds. This implies that the sequence space method can be used to simulate time-series data for aggregate variables and the distribution of agents. Throughout this note a simple *Heterogeneous Agent Neo-Classical* (HANC) is used as an example. Additional undocumented models are included in the example repository.

1.1 Model class

We consider economies where:

1. Time is discrete (index t).

2. There is a continuum of households (index i, when needed).

3. There is *perfect foresight* wrt. all aggregate variables, $\boldsymbol{X}$, indexed by $\mathcal{N}$,

$$\boldsymbol{X} = \{\boldsymbol{X}_t\}_{t=0}^{\infty} = \{\boldsymbol{X}^j\}_{j\in\mathcal{N}} = \{X_t^j\}_{t=0,j\in\mathcal{N}}^{\infty},$$

   where $\mathcal{N} = \mathcal{Z} \cup \mathcal{U} \cup \mathcal{O}$, and $\mathcal{Z}$ are *exogenous shocks,* $\mathcal{U}$ are *unknowns*, $\mathcal{O}$ are outputs, and $\mathcal{H} \in \mathcal{O}$ are *targets.*

4. The model structure is described in terms of a set of *blocks* indexed by $\mathcal{B}$, where each block has inputs, $\mathcal{I}_b \subset \mathcal{N}$, and outputs, $\mathcal{O}_b \subset \mathcal{O}$, and there exists functions $h^o(\{\boldsymbol{X}^i\}_{i\in\mathcal{I}_b})$ for all $o \in \mathcal{O}_b$.

5. The blocks are *ordered* such that (i) each output is *unique* to a block, (ii) the first block only have shocks and unknowns as inputs, and (iii) later blocks only additionally take outputs of previous blocks as inputs. This implies the blocks can be structured as a *directed acyclical graph* (DAG).

6. The number of targets are equal to the number of unknowns, and an *equilibrium* implies $\boldsymbol{X}^o = 0$ for all $o \in \mathcal{H}$. Equivalently, the model can be summarized by an *target equation system* from the unknowns and shocks to the targets,

$$\boldsymbol{H}(\boldsymbol{U}, \boldsymbol{Z}) = \boldsymbol{0}, \tag{1}$$

and an *auxiliary model equation* to infer all variables

$$\boldsymbol{X} = \boldsymbol{M}(\boldsymbol{U}, \boldsymbol{Z}). \tag{2}$$

A *steady state* satisfy

$$\boldsymbol{H}(\boldsymbol{U}_{ss}, \boldsymbol{Z}_{ss}) = 0 \text{ and } \boldsymbol{X}_{ss}{=}\boldsymbol{M}(\boldsymbol{U}_{ss}{,}\boldsymbol{Z}_{ss}).$$

7. The *discretized household block* can be written recursively as

$$\begin{aligned}
\boldsymbol{v}_t &= v(\underline{\boldsymbol{v}}_{t+1}, \boldsymbol{X}_t^{hh}) && (3)\\
\underline{\boldsymbol{v}}_t &= \Pi(\boldsymbol{X}_t^{hh})\boldsymbol{v}_t && (4)\\
\boldsymbol{D}_t &= \Pi(\boldsymbol{X}_t^{hh})'\underline{\boldsymbol{D}}_t && (5)\\
\underline{\boldsymbol{D}}_{t+1} &= \Lambda(\underline{\boldsymbol{v}}_{t+1}, \boldsymbol{X}_t^{hh})'\boldsymbol{D}_t && (6)\\
\boldsymbol{a}_t^* &= \boldsymbol{a}^*(\underline{\boldsymbol{v}}_{t+1}, \boldsymbol{X}_t^{hh}) && (7)\\
\boldsymbol{Y}_t^{hh} &= \boldsymbol{y}(\underline{\boldsymbol{v}}_{t+1}, \boldsymbol{X}_t^{hh})'\boldsymbol{D}_t && (8)
\end{aligned}$$

where

$\underline{\boldsymbol{D}}_0$ is given

$\boldsymbol{X}_t^{hh} = \{\boldsymbol{X}_t^i\}_{i\in\mathcal{I}_{hh}}$

$\boldsymbol{Y}_t^{hh} = \{\boldsymbol{X}_t^o\}_{o\in\mathcal{O}_{hh}}$,

where respectively $\underline{\boldsymbol{v}}_t$ and $\underline{\boldsymbol{D}}_t$ and $\boldsymbol{v}_t$ and $\boldsymbol{D}_t$ are the value functions and distributions *before* and *after* the realization of the idiosyncratic states with transition matrix $\Pi(\boldsymbol{X}_t^{hh})$, $\boldsymbol{a}_t^*$ is the policy functions, $\boldsymbol{Y}_t$ is aggregated outputs with $y(\underline{\boldsymbol{v}}_{t+1}, \boldsymbol{X}_t^{hh})$ as individual level measures.

8. Given the sequence of shocks, $\boldsymbol{Z}$, there exists a *truncation period*, T, such all variables return to steady state beforehand.

It is straightforward to numerically evaluate the model starting from the shocks and unknowns going forward block by block along the directed acyclical graph (DAG). Derivatives can also be calculated along this graph to construct Jacobians. Computationally, the central challenge is to compute the derivatives of the household block. This can be done efficiently as explained below in section 3 with the so-called »fake new algorithm« from Auclert et al. (2021). With the sequence-space Jacobians, the truncated equation system, $\boldsymbol{H}(\boldsymbol{U}, \boldsymbol{Z}) = 0$, can be solved straightforwardly with a quasi-Newton solver (e.g. using Broyden's method).

Alternatively, the model can be solved to a first order by total differentiating equation (1)

$$\boldsymbol{H_U} d\boldsymbol{U} + \boldsymbol{H_Z} d\boldsymbol{Z} = 0 \Leftrightarrow d\boldsymbol{U} = \underbrace{-\boldsymbol{H}_{\boldsymbol{U}}^{-1} \boldsymbol{H_Z}}_{\equiv \boldsymbol{G_U}} d\boldsymbol{Z}, \tag{9}$$

where we refer to $\boldsymbol{G}_U$ as the *general equilibrium solution matrix*. By also total differentiating equation (2), the remaining linearized impulse responses can be calculated as

$$\begin{aligned} d\boldsymbol{X} &= M_{\boldsymbol{U}} d\boldsymbol{U} + M_{\boldsymbol{Z}} d\boldsymbol{Z} \\ &= \underbrace{(-M_{\boldsymbol{U}} \boldsymbol{H}_{\boldsymbol{U}}^{-1} \boldsymbol{H_Z} + M_{\boldsymbol{Z}})}_{\equiv \boldsymbol{G}} d\boldsymbol{Z}. \end{aligned}$$

where $\boldsymbol{G}$ is the full *general equilibrium matrix*.

1.2 Example: A simple HANC-model

We now consider how a simple HANC-model fits into this setup.

Firms. A representative firm rent capital, K_{t-1}, and hire labor, L_t, to produce goods, with the production function

$$Y_t = \Gamma_t K_{t-1}^{\alpha} L_t^{1-\alpha}, \tag{10}$$

where Γ_t is technology and considered an exogenous shock. Capital depreciates with the rate δ. Profit maximization by

$$\max_{K_{t-1}, L_t} Y_t - w_t L_t - r_t^k K_{t-1}$$

implies the standard pricing equations

$$r_t^k = \alpha \Gamma_t (K_{t-1}/L_t)^{\alpha\text{-}\mathbf{1}} \tag{11}$$

$$w_t = (1-\alpha)\Gamma_t (K_{t-1}/L_t)^{\alpha} \tag{12}$$

where r_t^k is rental rate of capital and w_t is the wage rate.

Mutual fund. A zero-profit mutual fund owns all the capital. It take deposits from households, A_t, and pay a real return of $r_t = r_t^k - \delta$. It balance sheet is $A_t = K_t$.

Households. Households are heterogeneous ex ante with respect to their discount factor, β_i, and ex post with respect to their productivity, z_t, and assets, a_{t-1}. Each period household exogenously supply $\ell_t = z_t$ units of labor, and choose consumption c_t subject to a no-borrowing constraint. Households have *perfect foresight* wrt. to the interest rate and the wage rate, $\{r_t, w_t\}_{t=0}^{\infty}$, and solve the problem

$$
\begin{aligned}
v_t(\beta_i, z_t, a_{t-1}) = \max_{a_t, c_t} & \frac{c_t^{1-\sigma}}{1-\sigma} + \beta \mathbb{E}_t \left[v_{t+1}(\beta_i, z_{t+1}, a_t) \right] \\
\text{s.t.} & \\
\ell_t &= z_t \\
a_t + c_t &= (1 + r_t) a_{t-1} + w_t z_t \\
\log z_t &= \rho_z \log z_{t-1} + \psi_t \ \ , \psi_t \sim \mathcal{N}(\mu_\psi, \sigma_\psi), \, \mathbb{E}[z_t] = 1 \\
a_t &\geq 0,
\end{aligned}
\tag{13}
$$

where implicitly $v_t(\beta_i, z_t, a_{t-1}) = v(\beta_i, z_t, a_{t-1}, \{r_\tau, w_\tau\}_{\tau=t}^{\infty})$. We denote optimal policy functions by $a_t^*(\beta_i, z_t, a_{t-1})$, $\ell_t^*(\beta_i, z_t, a_{t-1})$, and $c_t^*(\beta_i, z_t, a_{t-1})$.

The household problem is discretized, and optimal savings function, a^*, is computed on sorted grids for β_i, z_t and a_{t-1} generically denoted $\mathcal{G}_x = \{x^0, x^1, \ldots, x^{\#_x - 1}\}$. The transition probabilities for z_t are denoted $\pi_{i_{z-1}, i_z} = \Pr[z_t = z^{i_z} | z_{t-1} = z^{i_{z-}}]$. We consider $\underline{\boldsymbol{D}}_t$ and $\boldsymbol{D}_t$ to be histograms in terms of probability masses at each grid point. The following updating algorithm can now be used given $\underline{\boldsymbol{D}}_t$

1. Stochastic simulation: For each i_β, i_z and i_{a-} calculate

$$
\boldsymbol{D}_t(\beta^{i_\beta}, z^{i_z}, a^{i_{a-}}) = \sum_{i_{z-}=0}^{\#_z - 1} \pi_{i_{z-}, i_z} \underline{\boldsymbol{D}}_t(\beta^{i_\beta}, z^{i_{z-}}, a^{i_{a-}})
$$

2. Initial zero mass: Set $\underline{\boldsymbol{D}}_{t+1}(\beta^{i_\beta}, z^{i_z}, a^{i_a}) = 0$ for all i_β, i_{z+} and i_a

3. Choice simulation: For each i_β, i_z and i_{a-} do

   (a) Find $\iota \equiv$ largest $i_a \in \{0, 1, \ldots, \#_a - 2\}$ such that $a^{i_a} \leq a_t^*(\beta^{i_\beta}, z^{i_z}, a^{i_{a-}})$

   (b) Calculate $\omega = \frac{a^{\iota+1} - a^*(z^{i_z}, a^{i_{a-}})}{a^{\iota+1} - a^{\iota}} \in [0, 1]$

   (c) Increment $\underline{\boldsymbol{D}}_{t+1}(\beta^{i_\beta}, z^{i_z}, a^{\iota})$ with $\omega \boldsymbol{D}_t(\beta^{i_\beta}, z^{i_z}, a^{i_{a-}})$

   (d) Increment $\underline{\boldsymbol{D}}_{t+1}(\beta^{i_\beta}, z^{i_z}, a^{\iota+1})$ with $(1 - \omega) \boldsymbol{D}_t(\beta^{i_\beta}, z^{i_z}, a^{i_{a-}})$

This algorithm first use the exogenous transition probabilities, π_{i_{z-}, i_z}, to simulate forward from the beginning-of-period distribution, $\underline{\boldsymbol{D}}_t$, to the choice-relevant distribution, $\boldsymbol{D}_t$. It secondly derives the next-period beginning-of-period distribution, $\underline{\boldsymbol{D}}_{t+1}$, by distributes probability mass to the neighboring grids points of the optimal savings choice (indexed by ι) using linear weights, ω. In matrix form the simulation can be written asThe histogram method can be

$$
\begin{aligned}
\boldsymbol{D}_t &= \Pi_z' \underline{\boldsymbol{D}}_t \\
\underline{\boldsymbol{D}}_{t+1} &= \Lambda_t \boldsymbol{D}_t
\end{aligned}
$$

where the stochastic transition matrix Π_z' is derived from the π_{i_{z-},i_z}'s, and the choice transition matrix is derived from the ι's and ω's.

The aggregate supply of savings, labor supply and consumption can be calculated as

$$\begin{aligned}
A_t^{hh} &= \int a_t^*(\beta_i, z_t, a_{t-1}) d\boldsymbol{D}_t \qquad (14)\\
&= \sum_{i_\beta}\sum_{i_z}\sum_{i_a} a_t^*(\beta_i, z_t, a_{t-1})\boldsymbol{D}_t(\beta^{i_\beta}, z^{i_z}, a^{i_a})\\
&= \boldsymbol{a}_t^{*\prime}\boldsymbol{D}_t\\
L_t^{hh} &= \int \ell_t^*(\beta_i, z_t, a_{t-1}) d\boldsymbol{D}_t \qquad (15)\\
&= \sum_{i_\beta}\sum_{i_z}\sum_{i_a} \ell_t^*(\beta_i, z_t, a_{t-1})\boldsymbol{D}_t(\beta^{i_\beta}, z^{i_z}, a^{i_a})\\
&= \boldsymbol{a}_t^{*\prime}\boldsymbol{D}_t\\
C_t^{hh} &= \int c_t^*(\beta_i, z_t, a_{t-1}) d\boldsymbol{D}_t \qquad (16)\\
&= \sum_{i_\beta}\sum_{i_z}\sum_{i_a} c_t^*(\beta_i, z_t, a_{t-1})\boldsymbol{D}_t(\beta^{i_\beta}, z^{i_z}, a^{i_a})\\
&= \boldsymbol{c}_t^{*\prime}\boldsymbol{D}_t
\end{aligned}$$

Market clearing. Market clearing requires

$$\begin{aligned}
\text{Capital: } & K_t = A_t = A_t^{hh}\\
\text{Labour: } & L_t = L_t^{hh} = 1\\
\text{Goods: } & Y_t = C_t^{hh} + \underbrace{K_t - K_{t-1} + \delta K_{t-1}}_{=I_t}
\end{aligned}$$

Stationary equilibrium. The *stationary equilibrium* (steady state for aggregate variables) for a given Γ_{ss} is

1. Quantities K_{ss} and L_{ss},
2. prices r_{ss} and w_{ss},
3. a distribution $\boldsymbol{D}_{ss}$ over z_{t-1} and a_{t-1}
4. and policy functions $a_{ss}^*(z_t, a_{t-1})$, $\ell_{ss}^*(z_t, a_{t-1})$ and $c_{ss}^*(z_t, a_{t-1})$

are such that

1. Firms maximize profits,

$$r_{ss} = \alpha\Gamma_{ss}(K_{ss}/L_{ss})^{\alpha-1} - \delta$$

   and

$$w_{ss} = (1-\alpha)\Gamma_{ss}\left(K_{ss}/L_{ss}\right)^{\alpha}$$

2. $a^*_{ss}(\bullet)$, $\ell^*_{ss}(\bullet)$, $c^*_{ss}(\bullet)$ solves the household problem with $\{r_{ss}, w_{ss}\}_{t=0}^{\infty}$
3. $\boldsymbol{D}_{ss} = \Lambda'_{ss}\Pi'_{ss}\boldsymbol{D}_{ss}$ is the invariant distribution implied by the household problem
4. Mutual fund balance sheet is satisfied, i.e. $A_{ss} = K_{ss}$
5. The capital market clears, i.e. $A_{ss} = \int a^*_{ss}(\beta_i, z_t, a_{t-1})d\boldsymbol{D}_{ss}$
6. The labor market clears, i.e. $L_{ss} = \int \ell^*_{ss}(\beta_i, z_t, a_{t-1})d\boldsymbol{D}_{ss} = \int z_{i,ss}d\boldsymbol{D}_{ss} = 1$
7. The goods market clears, i.e. $Y_{ss} = \int c^*_{ss}(\beta_i, z_t, a_{t-1})d\boldsymbol{D}_{ss} + \delta K_{ss}$

This is a root-finding problem, which can be solved as follows:

1. Guess on r_{ss}
2. Calculate w_{ss}
3. Solve the infinite horizon household problem
4. Simulate until convergence of $\boldsymbol{D}_{ss}$
5. Calculate household supply $A^{hh}_{ss} = \boldsymbol{a}^{*\prime}_{ss}\boldsymbol{D}_{ss}$ and $L_{ss} = L^{hh}_{ss} = \boldsymbol{\ell}^{*\prime}_{ss}\boldsymbol{D}_{ss}$
6. Calculate firm demand $A_{ss} = K_{ss} = \left(\frac{r_{ss}+\delta}{\alpha Z_{ss}}\right)^{\frac{1}{\alpha-1}} L_{ss}$
7. If for some tolerance ϵ
$$\left|A_{ss} - A^{hh}_{ss}\right| < \epsilon$$
then stop, otherwise update r_{ss} appropriately and return to step 2

Transition path. In terms of the general formulation above, we can write the model in terms of

1. Shocks: $\boldsymbol{Z} = \{\boldsymbol{\Gamma}\}$
2. Unknowns: $\boldsymbol{U} = \{\boldsymbol{K}, \boldsymbol{L}\}$
3. Targets: $\{A_t - A^{hh}_t\}$ (asset market clearing) and $\{L_t - L^{hh}_t\}$ (labor market clearing)
4. Aggregate variables: $\boldsymbol{X} = \{\boldsymbol{\Gamma}, \boldsymbol{K}, \boldsymbol{r}, \boldsymbol{w}, \boldsymbol{L}, \boldsymbol{C}, \boldsymbol{Y}, \boldsymbol{A}, \boldsymbol{A}^{hh}, \boldsymbol{C}^{hh}, \boldsymbol{L}^{hh}\}$
5. Household inputs: $\boldsymbol{X}^{hh}_t = \{\boldsymbol{r}, \boldsymbol{w}\}$
6. Household outputs: $\boldsymbol{Y}^{hh}_t = \{\boldsymbol{A}^{hh}, \boldsymbol{C}^{hh}, \boldsymbol{L}^{hh}\}$

This implies the equation system

$$\boldsymbol{H}(\boldsymbol{K}, \boldsymbol{L}, \boldsymbol{\Gamma}) = \boldsymbol{0} \Leftrightarrow \tag{17}$$
$$\begin{bmatrix} A_t - A^{hh}_t \\ L_t - L^{hh}_t \end{bmatrix} = \begin{bmatrix} 0 \end{bmatrix}, \quad \forall t \in \{0, 1, \ldots, T-1\}$$

where we have

$$
\begin{aligned}
r_t &= \alpha\Gamma_t(K_{t-1}/L_t)^{\alpha-1} - \delta \\
w_t &= (1-\alpha)\Gamma_t\left(\frac{r_t+\delta}{\alpha\Gamma_t}\right)^{\frac{\alpha}{\alpha-1}} \\
A_t &= K_t \\
A_t^{hh} &= \boldsymbol{a}_t^{*\prime}\boldsymbol{D}_t \\
L_t^{hh} &= \boldsymbol{\ell}_t^{*\prime}\boldsymbol{D}_t \\
\boldsymbol{D}_t &= \Pi_z^{\prime}\underline{\boldsymbol{D}}_t \\
\underline{\boldsymbol{D}}_{t+1} &= \Lambda_t\boldsymbol{D}_t \\
\underline{\boldsymbol{D}}_0 &\text{ is given}
\end{aligned}
$$

The sequence-space solution method described above can therefore be used to find the non-linear transition for an arbitrary sequence for for Γ_t. The full Jacobians can be written as

$$\boldsymbol{H_K} = \mathcal{J}^{A^{hh},r}\mathcal{J}^{r,K} + \mathcal{J}^{A^{hh},w}\mathcal{J}^{w,K} - \boldsymbol{I} \tag{18}$$

$$\boldsymbol{H_Z} = \mathcal{J}^{A^{hh},r}\mathcal{J}^{r,Z} + \mathcal{J}^{A^{hh},w}\mathcal{J}^{w,Z} \tag{19}$$

where $\mathcal{J}^{A^{hh},\bullet}$ are the Jacobians of the household problem, which must be found numerically, and $\mathcal{J}^{\bullet,K}$ and $\mathcal{J}^{\bullet,Z}$ are the Jacobians of the firm block, which can in principle be found analytically.

1.3 Aggregate risk and simulation

The sequence-space solution method above was used to solve with *perfect foresight* with respect to all aggregate variables. The linearized impulse responses can, however, be shown to also be the linearized impulse responses in a model with aggregate risk, where $\boldsymbol{Z}_t$ is a $MA(\infty)$ process with coefficient $d\boldsymbol{Z}_s$ for $s \in \{0, 1, \dots\}$ driven by the innovation $\boldsymbol{\epsilon}_t$.

For a time series of the innovations, $\tilde{\boldsymbol{\epsilon}}_t$, the resulting time series of the shocks and all endogenous variables can be computed *with truncation* by

$$d\tilde{\boldsymbol{Z}}_t = \sum_{s=0}^{T} d\boldsymbol{Z}_s\tilde{\boldsymbol{\epsilon}}_{t-s} \tag{20}$$

$$d\tilde{\boldsymbol{X}}_t = \sum_{s=0}^{T} d\boldsymbol{X}_s\tilde{\boldsymbol{\epsilon}}_{t-s} \tag{21}$$

where $d\boldsymbol{X}_s$ is the value of the impulse response function s periods after the shock has arrived.

To simulate a panel of household, we need to know how the policy functions change. Let $\partial\boldsymbol{a}_{i_g}^{\star}/\partial X_k^{hh}$ be the derivative of the policy function at grid point i_g to a k periods ahead shock to input X^{hh}. The impulse responses for each grind point then is then computed using the product rule

$$da^{\star}_{i_g,s} = \sum_{s'=s}^{T-1} \sum_{X^{hh} \in \boldsymbol{X}^{hh}} \frac{\partial a^{\star}_{i_g}}{\partial X^{hh}_{s'-s}} dX^{hh}_{s'}.$$

The time path of the policy can then be computed as

$$\boldsymbol{a}^{\star}_{i_g} = \sum_{s=0}^{T} da^{\star}_{i_g,s} \tilde{\boldsymbol{\epsilon}}_{t-s},$$

and a panel of households can be simulated using the standard updating rule for the distribution.

1.3.1 Simple HANC-model

Assume that Γ_t is an AR(1) process driven by Gaussian shocks with standard deviation σ then $d\boldsymbol{Z} = d\boldsymbol{\Gamma} = \begin{bmatrix} 1 & \rho & \rho^2 & \cdots \end{bmatrix}'$ and $\epsilon_t \sim \mathcal{N}(0,1)$ and the general formular above applies.

2 Using the `GEModelClass`

The central tool in `GEModelTools` is the `GEModelClass`, which is an add-on to the basic `EconModelClass` (documented here). An example of the setup is shown in Listing 1. The three methods `.settings()`, , `.setup()` and `.allocate()` are all called automatically when the model is created.

A `model` of the `GEModelClass` consists of the following list of `namespaces`:

1. **Parameters:** `.par`
2. **Steady state:** `.ss`
3. **Transition path:** `.path`
4. **Simulation:** `.sim`
5. **Initial state:** `.ini`

The user is required to specify a **list of ordered blocks** in `.settings()`. This is a list of strings with paths to jitted[1] functions for each block. Each block-function must have the format `function(par,ini,ss,var1,var2,...)`. The string `hh` designates the household block. The list of variables, `.varlist`, are derived from the blocks.

The user is required to specify some **variable lists** in `.settings()` for:

1. **Household grids:** `.grids_hh`.
   Used as `par.VARNAME_grid`.
   Must be in `.varlist`.
2. **Household inputs, direct:** `.inputs_hh`.
   Must be in `.varlist`.
3. **Household inputs, to transition matrix:** `.inputs_hh_z`.
   Must be in `.varlist`.
4. **Household outputs:** `.outputs_hh`.
   Must *not* be in `.varlist`.
   The aggregate variable `VARNAME.upper()_hh` is added to `.varlist`.
5. **Household policy functions:** `.pols_hh`.
   Must be subset of `.outputs_hh`.
6. **Household intertemporal variables:** `.intertemps_hh`.
   Must *not* be in `.varlist`.
7. **Shocks:** `.shocks`.
   Must be in `.varlist`.
8. **Unknowns:** `.unknowns`.
   Must be in `.varlist`.
9. **Targets:** `.targets`.
   Must be in `.varlist`.

[1] The function should be decorated with `@numba.njit`.

The user must choose the following **settings** in `.setup()`:

1. **Number of exogenous *fixed* states:** `par.Nfix`
2. **Number of exogenous *stochastic* states:** `par.Nz`
3. **Number of grid points for endogenous variables:** `par.Nendo1`, `par.Nendo2`,...
   where `endo1`, `endo2`,..., is in `.grids_hh`
4. (Optional) **Length of transition period:** `par.T=500`
5. (Optional) **Length of simulation:** `par.simT=1000`
6. (Optional) **For each shock in** `.shocks`**:**
   Initial jump: `par.jump_SHOCKNAME`
   Persistence: `par.rho_SHOCKNAME`
   Standard deviation: `par.std_SHOCKNAME`
7. (Optional) **Solver settings:**
   `par.max_iter_solve`, `par.max_iter_simulate`, `par.max_iter_broyden`
   `par.tol_solve`, `par.tol_simulate`, `par.tol_broyden`
8. (Optional) **Code settings:**
   `par.py_hh=True`: Python (no numba) when solving household problem.
   `par.py_blocks=True`: Python (no numba) when evaluating blocks.
   `par.full_z_trans=False`: Endogenous states in transition matrix.

Define `sol_shape = (par.Nfix,par.Nfix,par.Nendo1,par.Nendo2,...)`.

In `.allocate()` the internal **method** `.allocate_GE()` can now be called to allocate:

1. **Exogenous grids and transition matrices:**
   `par.z_grid`, `shape=(par.Nz,)`
   `ss.z_trans`, `shape=(par.Nfix,par.Nz,par.Nz)`
   `path.z_trans`, `shape=(par.T,par.Nz,par.Nz)`
   (or `shape=(par.T,par.Nendo1,...,par.Nz,par.Nz)`)
   `path.Dz`, `shape=(par.T,par.Nfix,par.Nz,)`
   `sim.z_trans`, `shape=(par.simT,par.Nz,par.Nz)`
   Remark: `path.z_trans[t]` is the transition matrix from $\underline{\boldsymbol{D}}_t$ to $\boldsymbol{D}_t$
2. **Beginning-of-period distribution,** $\underline{\boldsymbol{D}}_t$**:**
   `ss.Dbeg`, `shape=sol_shape`
   `ini.Dbeg`, `shape=sol_shape`
   `path.Dbeg`, `shape=(par.T,*sol_shape)`
   `sim.Dbeg`, `shape=(par.simT,*sol_shape)`
3. **Choice-relevant distribution,** $\boldsymbol{D}_t$**:**
   `ss.D`, `shape=sol_shape`
   `path.D`, `shape=(par.T,*sol_shape)`
   `sim.D`, `shape=(par.simT,*sol_shape)`

4. **Household outputs in `.outputs_hh`:**
   `ss.OUTPUTNAME`, `shape=sol_shape`
   `path.OUTPUTNAME`, `shape=(par.T,*sol_shape)`
   `sim.OUTPUTNAME`, `shape=(par.simT,*sol_shape)` (only `.pols_hh`)
   Aggregated variables:
   `ss.OUTPUTNAME.upper()_hh`, scalar
   `path.OUTPUTNAME.upper()_hh`, `shape=(par.T,)`
   `sim.OUTPUTNAME.upper()_hh`, `shape=(par.simT,)` (only `.pols_hh`)
   `sim.OUTPUTNAME.upper()_hh_from_D`, `shape=(par.simT,)` (only `.pols_hh`)
5. **Aggregate variables in `.varlist`:**
   `ss.VARNAME`, scalar
   `ini.VARNAME`, scalar
   `path.VARNAME`, `shape=(par.T,1)`
   `sim.VARNAME`, `shape=(par.simT,)`
   Remark:
   `path.VARNAME[t]` is the value in period t.
6. **Household Jacobian**, **`.jac_hh`:**
   `jac_hh[(OUTPUTNAME.upper()_hh,INPUTNAME)]`, each `shape=(par.T,par.T)`
7. **Full Jacobian**, **`.jac`:**
   `jac[(OUTPUTNAME,INPUTNAME)]`, each `shape=(par.T,par.T)`
8. **Solution matrix:**
   $\boldsymbol{H}_U$: `H_U`, with shape `shape=(par.T,par.T)`
   $\boldsymbol{H}_Z$: `H_Z`, with shape `shape=(par.T,par.T)`
   $\boldsymbol{G}_U$: `G_U`, with shape `shape=(par.T,par.T)`
9. **Impulse-responses of linearized model**, **`.IRF`:**
   `IRF[(OUTPUTNAME,SHOCKNAME)]`, each `shape=(par.T,)`

The user must also provide the following **methods**:

1. `.prephare_hh_ss()` (method), which creates the grids for all the variables in `.par`, choose the initial distribution `ss.Dbeg`, and choose the initial guesses for all variables in `.intertemps_hh()` in `.ss`. This is called each time we solve for the steady state of the household problem using in `.solve_hh_ss()`.

2. `.find_ss()` (method), which solves for the steady state, i.e. fills `ss`, and solve and simulate the household problem in steady state (call `.solve_hh_ss()` and `.simulate_hh_ss()`, see below).

And the following **jitted**[2] **function**:

[2] The function should be decorated with `@numba.njit`.

1. **Bellman iteration** (function), `.solve_hh_bakcwards()` which iterates one step backwards in the household problem. Arguments must be:
   `par` and `z_trans` (transition matrix in period t).
   all variables in `.inputs_hh`, `.inputs_hh_z`, `.outputs_hh` and `.intertemps_hh`.
   all variables in `.intertemps_hh` with suffix `_plus`.

If the transition matrix is time-varying it must be updated in this function. Otherwise the transition matrix from the stationary distribution will be used.

The following internal methods are **now available**:

1. `.solve_hh_ss()`: Solve household problem in steady state → `ss.VARNAME`.
2. `.simulate_hh_ss()`: Simulate household problem in steady state → `ss.D` and `ss.VARNAME.upper()_hh` for all variables in `.outputs_hh`.
3. `.solve_hh_path()`: Solve household problem → `path.VARNAME`
4. `.simulate_hh_path()`: Simulate household problem → `path.D` and `path.VARNAME.upper()_hh` for all variables in `.outputs_hh`.
5. `.compute_jacs(skip_hh=False,skip_shocks=False)`:
   Compute the Jacobians → `jac_hh`, `H_U`, `H_Z`, and `jac`.
6. `.find_transition_path(shocks)`: Find transition path → `path`.
7. `.find_IRFs(shocks,reuse_G_U=False)`:
   Find linearized impulse-response → `IRF`[VARNAME].
8. `.simulate(skip_hh=False,reuse_G_U=False)`: Simulate model → `sim.VARNAME`.

In both solution methods `.find_transition_path()` and `.find_IRFs_path()`, the input `shocks` can firstly be a list of *strings*. In this case, each chosen shock is the AR(1) given by `par.jump_VARNAME` and `par.rho_VARNAME`. Secondly, `shocks` can be a dictionary like `shock_specs={dVARNAME:PATH}`, where `PATH` is an arbitrary deviation from steady state.

The default for the *simulation* is to consider AR(1) shocks given by `par.std_VARNAME` and `par.rho_VARNAME`. The aggregated household variables also exists in a version with suffix `_from_D`, where the response is calculated by linearizing the policy function and then aggregating explicitely.

```python
from EconModel import EconModelClass
from GEModelTools import GEModelClass

class MyModelClass(EconModelClass,GEModelClass):

    def settings(self):

        self.grids_hh = [] # grids
        self.pols_hh = [] # policy functions
        self.inputs_hh = [] # inputs to hh problem, direct
        self.inputs_hh_z = [] # ... inputs to transition matrix
        self.outputs_hh = [] # output of hh problem
        self.intertemps_hh = [] # intertemporal variables in hh problem

        self.shocks = [] # exogenous inputs
        self.unknowns= [] # endogenous inputs
        self.targets = [] # targets
        self.blocks = [] # blocks

        self.solve_hh_backwards_step = solve_hh_backwards_step

    def setup(self):

        par = self.par
        par.Nfix = 1
        par.Nz = 7
        par.NVARNAME = 100 # number of grid points
        par.jump_VARNAME = -0.01 # initial jump
        par.rho_VARNAME = 0.8 # AR(1) coefficeint
        par.std_VARNAME = 0.01 # standard deviation
        par.T = 500 # length of path
        par.simT = 1000 # length of simulation

    def allocate(self):

        self.allocate_GE()

    def prepare_hh_ss(self): pass
    def find_ss(self): pass

```

Listing 1: Example: Setup

3 Efficient computation of the household Jacobian

In this section, we explain how the Jacobian of the household block can be computed efficiently. This algorithm is fully generic, and the package can be used without understanding this section in detail.

The household block can be summarized as

$$\boldsymbol{Y}^{hh} = hh(\boldsymbol{X}^{hh}). \tag{22}$$

We are interested in finding the Jacobian around the steady state, i.e.

$$\mathcal{J}^{hh} = \frac{dhh(\boldsymbol{X}^{hh}_{ss})}{d\boldsymbol{X}^{hh}}. \tag{23}$$

We let $\mathcal{J}^{hh,o,i}_{t,s}$ denote the derivative of output o to input i at time t for a shock at time s. Let $\bullet^{s,i}_t$ denote a variable in the equation system (3)-(8) when all inputs are at their steady state value *except in period* s, where there is an infinitesimal shock dx to input variable i. We then write

$$\underline{\boldsymbol{D}}^{s,i}_{t+1} = \left(\Lambda^{s,i}_t\right)' \left(\Pi^{s,i}_t\right)' \underline{\boldsymbol{D}}^{s,i}_t \tag{24}$$

$$\boldsymbol{Y}^{hh,s,i}_t = \left(\boldsymbol{y}^{s,i}_t\right)' \left(\Pi^{s,i}_t\right)' \underline{\boldsymbol{D}}^{s,i}_t \tag{25}$$

Building blocks. The value function equations (3) and (4) are forward looking so

$$\boldsymbol{v}^{s,i}_t = \boldsymbol{v}_{ss} \text{ for } t > s$$
$$\underline{\boldsymbol{v}}^{s,i}_t = \underline{\boldsymbol{v}}_{ss} \text{ for } t > s$$

Additionally, only the time span until the shock arrives matter so

$$\boldsymbol{v}^{s,i}_t = \boldsymbol{v}^{s-1,i}_{t-1} \text{ for } t \leq s$$
$$\underline{\boldsymbol{v}}^{s,i}_t = \underline{\boldsymbol{v}}^{s-1,i}_{t-1} \text{ for } t \leq s$$

This carries over to $\boldsymbol{y}^{s,i}_t$ and $\Lambda^{s,i}_t$ such that for all $t, s \geq 0$

$$\boldsymbol{y}^{s,i}_t = \begin{cases} \boldsymbol{y}_{ss} & t > s \\ \boldsymbol{y}^{T-1,i}_{T-1-(s-t)} & t \leq s \end{cases} \text{ and } \Lambda^{s,i}_t = \begin{cases} \Lambda_{ss} & t > s \\ \Lambda^{T-1,i}_{T-1-(s-t)} & t \leq s \end{cases}. \tag{26}$$

We finally have for all $t, s \geq 0$ that

$$\Pi^{s,i}_t = \begin{cases} \Pi_{ss} & t \neq s \\ \Pi^{T-1,i}_{T-1,ss} & t = s \end{cases} \tag{27}$$

This implies that $\boldsymbol{y}_t^{s,i}$, $\Lambda_t^{s,i}$ and $\Pi_t^{s,i}$ can all be found for any t and s once $\Pi_{T-1,ss}^{T-1,i}$ is known and $\boldsymbol{y}_t^{T-1,i}$ and $\Lambda_t^{T-1,i}$ is known for $t \in \{0, 1, \ldots, T-1\}$. This only requires a single backwards iteration from a shock in period $T-1$ for each input.

For later use, we define the following objects:

$$\mathcal{Y}_{0,s}^{o,i} \equiv \frac{dY_0^{o,s,i}}{dx} = \frac{\left(d\boldsymbol{y}_0^{o,s,i}\right)'}{dx} \left(\Pi_{ss}\right)' \underline{\boldsymbol{D}}_{ss} + \begin{cases} \boldsymbol{y}_{ss}^o \frac{\left(d\Pi_0^{s,i}\right)'}{dx} \underline{\boldsymbol{D}}_{ss} & \text{if } s = 0 \\ 0 & \text{else} \end{cases}$$

$$\underline{\mathcal{D}}_{1,s}^{i} \equiv \frac{d\underline{\boldsymbol{D}}_1^{s,i}}{dx} = \frac{\left(d\Lambda_0^{s,i}\right)'}{dx} \left(\Pi_{ss}\right)' \underline{\boldsymbol{D}}_{ss} + \begin{cases} \Lambda_{ss}' \frac{\left(d\Pi_0^{s,i}\right)'}{dx} \underline{\boldsymbol{D}}_{ss} & \text{if } s = 0 \\ 0 & \text{else} \end{cases}$$

$$\mathcal{E}_t^o \equiv \left(\Pi_{ss}\Lambda_{ss}\right)^t \Pi_{ss}\boldsymbol{y}_{ss}^o,$$

where $\mathcal{Y}_{0,s}^{o,i}$ and $\underline{\mathcal{D}}_{1,s}^{i}$ are derivatives of the outputs and the distribution at respectively time 0 and time 1 to a shock at time s, and $\mathcal{E}_t^o$ is an expectation vector. The cost of computing $\mathcal{Y}_{0,s}^{o,i}$ and $\underline{\mathcal{D}}_{1,s}^{i}$ for $s \in \{0, 1, \ldots, T-1\}$ are similar to a full forward simulation for T periods. The cost of computing $\mathcal{E}_s^o$ is negligible in comparison and can be done recursively, $\mathcal{E}_t^o = \Pi_{ss}\Lambda_{ss}\mathcal{E}_{t-1}^o$ with $\mathcal{E}_0^o = \Pi_{ss}\boldsymbol{y}_{ss}^o$.

The task is to build the full Jacobian from these building blocks. To do this, we first need to take the total derivative of (24) and (25) around the steady state

$$d\underline{\boldsymbol{D}}_{t+1}^{s,i} = \Lambda_{ss}'\Pi_{ss}' d\underline{\boldsymbol{D}}_t^{s,i} + (d\Lambda_t^{s,i})'\Pi_{ss}'\underline{\boldsymbol{D}}_{ss} + \Lambda_{ss} d\Pi_t^{s,i}\underline{\boldsymbol{D}}_{ss} \tag{28}$$

$$dY_t^{o,s,i} = \left(\boldsymbol{y}_{ss}^o\right)' \Pi_{ss}' d\underline{\boldsymbol{D}}_t^{s,i} + (d\boldsymbol{y}_t^{o,s,i})'\Pi_{ss}'\underline{\boldsymbol{D}}_{ss} + \left(\boldsymbol{y}_{ss}^o\right)' \left(d\Pi_t^{s,i}\right)' \underline{\boldsymbol{D}}_{ss} \tag{29}$$

Edge of the Jacobian First consider the effect on output at time 0 from a shock at time s. We immediately have

$$\mathcal{J}_{0,s}^{hh,i,o} = \frac{d\boldsymbol{Y}_0^{o,s,i}}{dx} = \mathcal{Y}_s^{o,i} \tag{30}$$

Next consider the effect on output at time $t \geq 1$ form a shock at time 0. Combining equation (28) with only the time span mattering in equations (26)-(27) implies for $t \geq 2$

$$\begin{aligned} d\underline{\boldsymbol{D}}_t^{0,i} =& \Lambda_{ss}'\Pi_{ss}' d\underline{\boldsymbol{D}}_{t-1}^{0,i} \\ &+ \underbrace{\left(d\Lambda_{t-1}^{0,i}\right)'}_{=0}\Pi_{ss}'\underline{\boldsymbol{D}}_{ss} + \Lambda_{ss}'\underbrace{\left(d\Pi_t^{0,i}\right)'}_{=0}\underline{\boldsymbol{D}}_{ss} \\ =& \Lambda_{ss}'\Pi_{ss}' d\underline{\boldsymbol{D}}_{t-1}^{0,i} \\ & \vdots \\ =& \left(\Lambda_{ss}'\Pi_{ss}'\right)^{t-1} d\underline{\boldsymbol{D}}_1^{0,i}. \end{aligned} \tag{31}$$

Combining (29) with only the time span mattering in equation (26)-(27) , and the above equation (34) implies for $t \geq 1$,

$$
\begin{aligned}
dY_t^{o,0,i} &= (\boldsymbol{y}_{ss}^o)' \,\Pi_{ss}' d\underline{\boldsymbol{D}}_t^{0,i} \\
&\quad + \underbrace{\left(d\boldsymbol{y}_t^{o,0,i}\right)'}_{=0} \underline{\boldsymbol{D}}_{ss} + \Lambda_{ss}' \underbrace{\left(d\Pi_t^{0,i}\right)'}_{=0} \underline{\boldsymbol{D}}_{ss}. \\
&= (\boldsymbol{y}_{ss}^o)' \,\Pi_{ss}' d\underline{\boldsymbol{D}}_t^{0,i} \\
&= (\boldsymbol{y}_{ss}^o)' \,\Pi_{ss}' \left(\Lambda_{ss}' \Pi_{ss}'\right)^{t-1} d\underline{\boldsymbol{D}}_1^{0,i}
\end{aligned}
\tag{32}
$$

Combining equation (31) and (32) implies for $t \geq 1$

$$
\mathcal{J}_{t,0}^{hh,i,o} = \frac{dY_t^{o,0,i}}{dx} = \left(\mathcal{E}_{t-1}^o\right)' \underline{\mathcal{D}}_0^i \tag{33}
$$

Inner parts of the Jacobian. Combining (28) with only the time span mattering in equations (26)-(27), implies for $t, s \geq 1$

$$
\begin{aligned}
d\underline{\boldsymbol{D}}_t^{s,i} - d\underline{\boldsymbol{D}}_{t-1}^{s-1,i} =& \Lambda_{ss}' \Pi_{ss}' \left(d\underline{\boldsymbol{D}}_{t-1}^{s,i} - d\underline{\boldsymbol{D}}_{t-2}^{s-1,i}\right) \\
&+ \underbrace{\left(d\Lambda_{t-1}^{s,i} - d\Lambda_{t-2}^{s-1,i}\right)'}_{=0} \Pi_{ss}' \boldsymbol{D}_{ss} + \Lambda_{ss}' \underbrace{\left(d\Pi_{t-1}^{s,i} - d\Pi_{t-2}^{s-1,i}\right)'}_{=0} \underline{\boldsymbol{D}}_{ss} \\
=& \Lambda_{ss}' \Pi_{ss}' \left(d\underline{\boldsymbol{D}}_{t}^{s,i} - d\underline{\boldsymbol{D}}_{t-1}^{s-1,i}\right) \\
&\vdots \\
=& \left(\Lambda_{ss}' \Pi_{ss}'\right)^{t-1} \left(d\underline{\boldsymbol{D}}_1^{s,i} - \underbrace{d\underline{\boldsymbol{D}}_0^{s-1,i}}_{=0}\right) \\
=& \left(\Lambda_{ss}' \Pi_{ss}'\right)^{t-1} d\underline{\boldsymbol{D}}_1^{s,i}
\end{aligned}
\tag{34}
$$

where $\underline{\boldsymbol{D}}_0 = \underline{\boldsymbol{D}}_{ss}$ implies $d\underline{\boldsymbol{D}}_0^{s-1,i} = 0$ in the next to last line.

Combining (29) with only the time span mattering in equation (26)-(27) , and the above equation (34), implies for $t, s \geq 1$

$$
\begin{aligned}
dY_t^{o,s,i} - dY_{t-1}^{o,s-1,i} =& (\boldsymbol{y}_{ss}^o)' \Pi_{ss}' \left(d\underline{\boldsymbol{D}}_t^{s,i} - d\underline{\boldsymbol{D}}_{t-1}^{s-1,i}\right) \\
&+ \underbrace{\left(d\boldsymbol{y}_t^{o,s,i} - d\boldsymbol{y}_{t-1}^{o,s-1,i}\right)'}_{=0} \Pi_{ss}' \boldsymbol{D}_{ss} + \boldsymbol{y}_{ss}^o \underbrace{\left(d\Pi_t^{s,i} - d\Pi_{t-1}^{s-1,i}\right)'}_{=0} \underline{\boldsymbol{D}}_{ss} \\
=& (\boldsymbol{y}_{ss}^o)' \Pi_{ss}' \left(d\underline{\boldsymbol{D}}_t^{s,i} - d\underline{\boldsymbol{D}}_{t-1}^{s-1,i}\right) \\
=& (\boldsymbol{y}_{ss}^o)' \Pi_{ss}' \left(\Lambda_{ss}' \Pi_{ss}'\right)^{t-1} d\underline{\boldsymbol{D}}_1^{s,i} \\
=& \left(\mathcal{E}_{t-1}^o\right)' d\underline{\boldsymbol{D}}_1^{s,i}
\end{aligned}
\tag{35}
$$

Combining (34) and (35) for $t, s \geq 1$ implies

$$
\begin{aligned}
\mathcal{J}^{hh,i,o}_{t,s} - \mathcal{J}^{hh,i,o}_{t-1,s-1} &= \frac{dY^{o,s,i}_{t} - dY^{o,s-1,i}_{t-1}}{dx} \Leftrightarrow \\
\mathcal{J}^{hh,i,o}_{t,s} &= \mathcal{J}^{hh,i,o}_{t-1,s-1} + \left(\mathcal{E}^{o}_{t-1}\right)' \underline{\mathcal{D}}^{i}_{s}
\end{aligned}
\tag{36}
$$

Recursive formulation Define the object

$$
\mathcal{F}^{i,o}_{t,s} \equiv \begin{cases} \mathcal{Y}^{o,i}_{s} & t = 0 \\ \left(\mathcal{E}^{o}_{t-1}\right)' \underline{\mathcal{D}}^{i}_{s} & t \geq 1 \end{cases} \tag{37}
$$

Combining equations (30), (33) and (36), the household Jacobian can be written recursively by

$$
\mathcal{J}^{i,o}_{t,s} = \sum_{k=0}^{\min\{t,s\}} \mathcal{F}^{i,o}_{t-k,s-k} \tag{38}
$$

4 Additional features

The following methods are available:

1. `.info(...)`
   Get an overview of the model.
2. `.draw_DAG(...)`
   Draw a DAG of the model.
3. `.show_IRFS(...)`
   Show IRFs.
4. `.compare_IRFS(...)`
   Compare IRFs across models.
5. `.decompose_hh_path(...)`
   Decompose household transition path (varying inputs and initial distribution)
6. `.decompose_hh_path(...)`
   Decompose blocks.
7. `.test_hh_path(...)`:
   Test time-invariance when inputs are at their steady state values.
8. `.test_path(...)`:
   Test time-invariance when inputs are at their steady state values.
9. `.test_jacs(...)`:
   Compare Jacobians calculated with a direct and the fake news algorithm.

10. `.update_aggregate_settings_jacs(...)`:
    Update aggregate settings in terms of shocks, unknowns and targets.

11. `.compress_full(...)`
    Save memory. No new solution can be found (de-allocates Jacobians and policy functions).

5 Troubleshooting

The transition path cannot be found. Considering the following

1. Use finer tolerances for finding the steady state
   `par.tol_solve`$\downarrow$
   `par.tol_simulate`$\downarrow$

2. Extend the transition period
   `par.T`$\uparrow$

3. Decrease the size and persistence of the size
   `par.jump_VARNAME`$\downarrow$
   `par.rho_VARNAME`$\downarrow$

4. Change other parameters making the model more stable
   (e.g. more strict Taylor rule, less sticky prices/wages)

References

Auclert, A., Bardóczy, B., Rognlie, M., and Straub, L. (2021). Using the Sequence-Space Jacobian to Solve and Estimate Heterogeneous-Agent Models. *Econometrica*, 89(5):2375–2408.

Boppart, T., Krusell, P., and Mitman, K. (2018). Exploiting MIT shocks in heterogeneous-agent economies: the impulse response as a numerical derivative. *Journal of Economic Dynamics and Control*, 89:68–92.