The `GEModelTools` Package for Solving HANK Models in Python

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

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/JeppeDruedahl/GEModelTools
Notebooks: github.com/JeppeDruedahl/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 Aiygari-model is used as an example.

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

$$\begin{aligned}
\boldsymbol{v}_t &= v(\boldsymbol{v}_{t+1}, \boldsymbol{X}_t^{hh}) && (3)\\
\boldsymbol{a}_t^* &= a^*(\boldsymbol{v}_{t+1}, \boldsymbol{X}_t^{hh}) && (4)\\
\boldsymbol{D}_t &= \Pi(\boldsymbol{X}_t^{hh})'\underline{\boldsymbol{D}}_t && (5)\\
\boldsymbol{Y}_t^{hh} &= \boldsymbol{y}(\boldsymbol{v}_{t+1}, \boldsymbol{X}_t^{hh})'\boldsymbol{D}_t && (6)\\
\underline{\boldsymbol{D}}_{t+1} &= \tilde{\Lambda}(\boldsymbol{v}_{t+1}, \boldsymbol{X}_t^{hh})'\boldsymbol{D}_t && (7)\\
&\boldsymbol{D}_0 \text{ is given}\\
&\boldsymbol{X}_t^{hh} = \{\boldsymbol{X}_t^i\}_{i\in\mathcal{I}_{hh}}\\
&\boldsymbol{Y}_t^{hh} = \{\boldsymbol{X}_t^o\}_{o\in\mathcal{O}_{hh}},
\end{aligned}$$

where $\boldsymbol{v}_t$ and $a_t^\star$ are the value and policy functions over some grid of the idiosyncratic states, $\boldsymbol{D}_t$ is the distribution over the idiosyncratic states in terms of the probability mass at each grid point with *endogenous* transition matrix $\tilde{\Lambda}(\boldsymbol{v}_{t+1}, \boldsymbol{X}_t^{hh})$ and *exogenous* transition matrix $\Pi(\boldsymbol{X}_t^{hh})$, $\boldsymbol{Y}_t$ is aggregated outputs with $y(\boldsymbol{v}_{t+1}, \boldsymbol{X}_t)$ as individual level measures. The full transition matrix is

$$\begin{aligned}
\Lambda(\boldsymbol{v}_{t+1}, \boldsymbol{X}_t^{hh}) &= \tilde{\Lambda}(\boldsymbol{v}_{t+1}, \boldsymbol{X}_t^{hh})\Pi(\boldsymbol{X}_{t+1}^{hh})\\
\boldsymbol{D}_{t+1} &= \Lambda(\boldsymbol{v}_{t+1}, \boldsymbol{X}_t^{hh}, \boldsymbol{X}_{t+1}^{hh})'\boldsymbol{D}_t
\end{aligned}$$

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)

$$\boldsymbol{H_U}d\boldsymbol{U} + \boldsymbol{H_Z}d\boldsymbol{Z} = 0 \Leftrightarrow d\boldsymbol{U} = \underbrace{-\boldsymbol{H_U}^{-1}\boldsymbol{H_Z}}_{\equiv \boldsymbol{G_U}}d\boldsymbol{Z}, \tag{8}$$

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

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

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

1.2 Example: A simple Aiygari-model

We now consider how a simple Aiygari-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

$$Y_t = \Gamma_t K_{t-1}^{\alpha} L_t^{1-\alpha}, \tag{9}$$

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

$$r_t^k = \alpha \Gamma_t (K_{t-1}/L_t)^{\alpha\text{-}\mathbf{1}} \tag{10}$$

$$w_t = (1-\alpha) \Gamma_t (K_{t-1}/L_t)^{\alpha} \tag{11}$$

where r_t^k is rental rate of capital and w_t is the wage rate. The implied (real) interest rate is $r_t = r_t^k - \delta$.

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 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

$$\begin{aligned} v_t(\beta_i, z_t, a_{t-1}) = \max_{a_t, c_t} & \frac{c_t^{1-\sigma}}{1-\sigma} + \beta \mathbb{E}_t \left[v_{t+1}(\beta_i, z_{t+1}, a_t) \right] \\ \text{s.t.} & \\ a_t + c_t &= (1 + r_t) a_{t-1} + w_t z_t \\ \log z_t &= \rho \log z_{t-1} + \varepsilon_t^z \quad , \varepsilon_t^z \sim \mathcal{N}(\mu_z, \sigma_z), \, \mathbb{E}[z_t] = 1 \\ a_t &\geq 0, \end{aligned} \tag{12}$$

where implicitly $v_t(\beta_i, z_t, a_{t-1}) = v(\beta_i, z_t, a_{t-1}, \{r_\tau, w_\tau\}_{\tau=t}^{\infty})$.

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 $\pi_{i_{z+},i_z} = \Pr[z^{i_{z+}}|z^{i_z}]$. Let $\boldsymbol{D}_t$ denote the distribution of households in terms of the probability mass on each grid point (i.e. a discrete histogram). The following updating algorithm can now be used:

1. Set $\underline{\boldsymbol{D}}_{t+1}(\beta^{i_{\beta+}}, z^{i_{z+}}, a^{i_{a+}}) = 0$ for all $i_{\beta+}$, i_{z+} and i_{a+}.
2. For each i_β, i_z and i_{a+}:
   (a) Find largest $\iota \in \{0, 1, \ldots, \#_a - 2\}$ such that $a^\star(i_\beta, i_z, i_a) \geq a^\iota$.
   (b) Calculate $\omega = \frac{a^{\iota+1} - a^*(i_\beta, i_z, i_a)}{a^{\iota+1} - a^\iota} \in [0, 1]$.
   (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 $(1-\omega)\boldsymbol{D}_t(\beta^{i_\beta}, z^{i_z}, a^{i_a})$
3. For each i_β, i_z and i_{a+}: $\boldsymbol{D}_{t+1}(\beta^{i_{\beta+}}, z^{i_{z+}}, a^{i_{a+}}) = \sum_{i_z=0}^{\#_z - 1} \pi_{i_{z+},i_z} \underline{\boldsymbol{D}}_{t+1}(\beta^{i_\beta}, z^{i_z}, a^{i_a})$

This algorithm first distributes probability mass from an existing point in $\boldsymbol{D}_t$, to the neighboring points of the optimal savings choice in the endogenous dimension, weighted by ω, to get the beginning-of-period distribution $\underline{\boldsymbol{D}}_t$. Second, it accounts for the transition to decision-relevant distribution, $\boldsymbol{D}_{t+1}$, by the exogenous transition probabilities, π_{i_{z+},i_z}. In matrix form this is $\boldsymbol{D}_{t+1} = \Pi_t' \tilde{\Lambda}_t' \boldsymbol{D_t}$, where the $\tilde{\Lambda}_t$ is constructed from ι and ω, and Π_t is constructed from π_{i_{z+},i_z}.

The aggregate supply of savings can be calculated as

$$\begin{aligned} A_t^{hh} &= \int a_t^*(\beta_i, z_t, a_{t-1}) d\boldsymbol{D}_t \\ &= \sum_{i_\beta} \sum_{i_z} \sum_{i_a} a_t^*(\beta_i, z_t, a_{t-1}) \boldsymbol{D}_t(\beta^{i_\beta}, z^{i_z}, a^{i_a}) \\ &= \boldsymbol{a}_t' \boldsymbol{D}_t \end{aligned} \tag{13}$$

and aggregate consumption can be calculated as

$$\begin{aligned} C_t^{hh} &= \int c_t^*(\beta_i, z_t, a_{t-1}) d\boldsymbol{D}_t \\ &= \sum_{i_\beta} \sum_{i_z} \sum_{i_a} c_t^*(\beta_i, z_t, a_{t-1}) \boldsymbol{D}_t(\beta^{i_\beta}, z^{i_z}, a^{i_a}) \\ &= \boldsymbol{c}_t' \boldsymbol{D}_t \end{aligned} \tag{14}$$

Market clearing. Market clearing requires

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

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 a_{t-1} and z_t
4. and policy functions $a_{ss}^*(z_t, a_{t-1})$ and $c_{ss}^*(z_t, a_{t-1})$

are such that

1. $a_{ss}^*(\bullet)$ and $c_{ss}^*(\bullet)$ solves the household problem with $\{r_{ss}, w_{ss}\}_{t=0}^{\infty}$
2. $\boldsymbol{D}_{ss} = \Lambda_{ss}' \boldsymbol{D}_{ss}$ is the invariant distribution implied by the household problem
3. Firms maximize profits, $r_{ss} = \alpha A_{ss}(K_{ss}/L_{ss})^{\alpha-1}$ and $w_{ss} = (1-\alpha)A_{ss}\left(\frac{r_t+\delta}{\alpha A_{ss}}\right)^{\frac{\alpha}{\alpha-1}}$
4. The labor market clears, i.e. $L_{ss} = \int z_{ss} dD_{ss} = 1$
5. The capital market clears, i.e. $K_{ss} = \int a_{ss}^*(\beta_i, z_t, a_{t-1}) dD_{ss}$
6. The goods market clears, i.e. $Y_{ss} = \int c_{ss}^*(\beta_i, z_t, a_{t-1}) dD_{ss} + \delta K_{ss}$

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 supply $A_{ss}^{hh} = \int a_{ss}^*(\beta_i, z_t, a_{t-1}) d\boldsymbol{D}_{ss}$
6. Calculate demand $K_{ss} = \left(\frac{r_{ss}+\delta}{\alpha Z_{ss}}\right)^{\frac{1}{\alpha-1}} L_{ss}$
7. If for some tolerance ϵ
$$\left|A_{ss}^{hh} - K_{ss}\right| < \epsilon$$
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: $\boldsymbol{U} = \{\boldsymbol{K}\}$
3. Targets: $\{K_t - A_t^{hh}\}$ (asset market clearing)
4. Aggregate variables: $\boldsymbol{X} = \{\boldsymbol{\Gamma}, \boldsymbol{K}, \boldsymbol{r}, \boldsymbol{w}, \boldsymbol{L}, \boldsymbol{C}, \boldsymbol{Y}, \boldsymbol{A}^{hh}, \boldsymbol{C}^{hh}\}$
5. Household inputs: $\boldsymbol{X}_t^{hh} = \{\boldsymbol{r}, \boldsymbol{w}\}$
6. Household outputs: $\boldsymbol{Y}_t^{hh} = \{\boldsymbol{A}^{hh}, \boldsymbol{C}^{hh}\}$

This implies the equation system

$$\begin{aligned} \boldsymbol{H}(\boldsymbol{K}, \boldsymbol{\Gamma}) = \boldsymbol{0} \Leftrightarrow \\ \left[\begin{array}{c} K_t - A_t^{hh} \end{array} \right] = \left[\begin{array}{c} 0 \end{array} \right], \ \forall t \in \{0, 1, \ldots, T-1\} \end{aligned} \tag{15}$$

where we have

$$\begin{aligned} L_t &= 1 \\ r_t &= \alpha \Gamma_t (K_{t-1}/L_t)^{\alpha - 1} \\ w_t &= (1-\alpha)\Gamma_t \left(\frac{r_t + \delta}{\alpha \Gamma_t} \right)^{\frac{\alpha}{\alpha - 1}} \\ A_t^{hh} &= \boldsymbol{a}_t' \boldsymbol{D}_t \\ \boldsymbol{D}_{t+1} &= \Lambda_t \boldsymbol{D}_t \\ \boldsymbol{D}_0 &= \Pi_{ss} \Lambda_{ss}' \boldsymbol{D}_{ss} \end{aligned}$$

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

$$\boldsymbol{H}_{\boldsymbol{K}} = \mathcal{J}^{A^{hh}, r} \mathcal{J}^{r, K} + \mathcal{J}^{A^{hh}, w} \mathcal{J}^{w, K} - \boldsymbol{I} \tag{16}$$

$$\boldsymbol{H}_{\boldsymbol{Z}} = \mathcal{J}^{A^{hh}, r} \mathcal{J}^{r, Z} + \mathcal{J}^{A^{hh}, w} \mathcal{J}^{w, Z} \tag{17}$$

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

$$d\tilde{\boldsymbol{Z}}_t = \sum_{s=0}^{T} d\boldsymbol{Z}_s \tilde{\boldsymbol{\epsilon}}_{t-s} \tag{18}$$

$$d\tilde{\boldsymbol{X}}_t = \sum_{s=0}^{T} d\boldsymbol{X}_s \tilde{\boldsymbol{\epsilon}}_{t-s} \tag{19}$$

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 $\partial \boldsymbol{a}^{\star}_{i_g} / \partial X^{hh}_k$ 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 Aiygari-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 formulate 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

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

1. **Aggregate variables:** .varlist.
   Used as path.VARNAME.
2. **Household grids:** .grid_hh.
   Used as par.VARNAME_grid.
3. **Household inputs:** .inputs_hh.
   Must be in .varlist.
4. **Household outputs:** .outputs_hh.
   Must *not* be in .varlist.
   The aggregate variable VARNAME.upper()_hh must be in .varlist_hh.
5. **Household policy functions:** .pols_hh.
   Must be subset of .outputs_hh.
6. **Household intertemporal variables:** .pols_hh.
   Must *not* be in .varlist.
7. **Shocks:** .shocks.
   Should be in .varlist.
8. **Unknowns:** .unknowns.
   Should be in .varlist.
9. **Targets:** .targets.
   Should be in .varlist.

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. **Length of transition period:** par.T
5. **Length of simulation:** par.simT

6. **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`

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

1. **Exogenous grids and transition matrices:**
   `par.z_grid_ss`, `shape=(par.Nz,)`
   `par.z_trans_ss`, `shape=(par.Nz,)`
   `par.z_ergodic_ss`, `shape=(par.Nz,)`
   `par.z_grid_path`, `shape=(par.T,par.Nz)`
   `par.z_trans_path`, `shape=(par.T,par.Nz,par.Nz)`
   Remark: `path.z_trans_path[t]` is the transition matrix from $\underline{D}_t$ to D_t
2. **Distribution:**
   `ss.D`, `shape=sol_shape`
   `path.D`, `shape=(par.T,*sol_shape)`
   `sim.D`, `shape=(par.simT,*sol_shape)`
3. **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`)
4. **Aggregate variables in** `.varlist`**:**
   `ss.VARNAME`, scalar`path.VARNAME`
   `shape=(par.T,len(inputs_endo)×par.T)`
   `sim.VARNAME`, `shape=(par.simT,)`
   Remark:
   `path.VARNAME[0,t]` is the value in period t.
   `path.VARNAME[i,t]` for $i > 0$ should be considered *undefined behavior*.
5. **Household Jacobian**, `.jac_hh`**:**
   `jac_hh[(OUTPUTNAME.upper()_hh,INPUTNAME)]`, each `shape=(par.T,par.T)`
6. **Full Jacobian**, `.jac`**:**
   `jac[(OUTPUTNAME,INPUTNAME)]`, each `shape=(par.T,par.T)`

7. **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)`
8. **Impulse-responses of linearized model**, `.IRF`**:**
   `IRF[(OUTPUTNAME,SHOCKNAME)]`, each `shape=(par.T,)`

Finally, the user must also provide the following **methods**:

1. `.create_grids()` (method), which creates the grids for the endogenous variables, the grid and transition matrix for the exogenous variable, and compute guesses for all variables in `.intertemps_hh()` in `.ss`. This is called each time we solve for the steady state of the household problem (in `.solve_hh_ss()`, see below)
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).
3. `.create_grids_path()` (method), which creates the grid and the transition matrix for the exogenous variable along the transition path. Can rely on paths in `.inputs_hh`. This is called each time we solve the household problem backwards along the transition path (in `.solve_hh_path()`, see below).

Compared to the general model formulation above the code, currently, only allows for a structure with the household block and pre- and post-blocks. The user must provide the following **jitted**[1] **functions**:

1. **Bellman iteration** (function), `.solve_hh_bakcwards_step()` which iterates one step backwards in the household problem. Inputs must by: `par` and `z_grid` (grid for z in period t), `z_trans_plus` (transition matrix in $t+1$).
   All variables in `.inputs_hh`, `.outputs_hh` and `.intertemps_hh`.
   All variables in `.intertemps_hh` with suffice `_plus`.
2. **Evaluate block before household block** (function), `.block_pre()`. Inputs must be: `(par,ss,path,ncols)`, where `ncols` is 1 or `len(unknowns)`$\times$`par.T`, and the latter is used to evaluate the path simultaneously for all one step changes. After this block has been evaluated all paths in `.inputs_hh` must be filled.
3. **Evaluate block after household block** (function), `.block_post()`. Same requirements as `.block_pre()`. Can rely on paths in `.ouputs_hh`.

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

1. `.solve_hh_ss()`: Solve household problem in steady state $\rightarrow$ `ss.VARNAME`.

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

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()`: Find transition path → `path`.
7. `.find_IRFs(reuse_G_U=False)`: Find linearized impulse-response → `IRF`[VARNAME].
8. `.simulate(skip_hh=False,reuse_G_U=False)`: Simulate model → `sim.VARNAME`.

The default for the *impulses-responses* is to consider AR(1) shocks given by `par.jump_VARNAME` and `par.rho_VARNAME`. Both solution methods `.find_transition_path()` and `.find_IRFs_path()` also accepts a dictionary like `shock_specs={dSHOCKE:CUSTOMPATH}` as input, where `CUSTOMPATH` is an arbitrary path for the shocks for which to compute the solution.

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
        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.varlist = [] # all variables

        self.solve_hh_backwards_step = solve_hh_backwards_step
        self.block_pre = block_pre
        self.block_post = block_post

    def setup(self):

        par = self.par
        par.Nfix = 1
        par.Nz = 2
        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 = 500 # length of simulation

    def allocate(self):

        self.allocate_GE()

    def find_ss(self): pass
    def create_grids(self): pass
    def create_grids_path(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. Throughout the section we for simplicity assume $\Pi(\boldsymbol{X}_t^{hh}) = \Pi_{ss}$, such that we have $\boldsymbol{D}_0 = \boldsymbol{D}_{ss}$ as given.[2]

The household block can be summarized as

$$\boldsymbol{Y}^{hh} = hh(\boldsymbol{X}^{\boldsymbol{hh}}). \tag{20}$$

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

$$\mathcal{J}^{hh} = \frac{dhh(\boldsymbol{X}_{ss}^{hh})}{d\boldsymbol{X}^{\boldsymbol{hh}}}. \tag{21}$$

We let $\mathcal{J}_{t,s}^{hh,o,i}$ denote the derivative of output o to input i at time t for a shock at time s. Let $\boldsymbol{v}_t^{s,i}$, $\boldsymbol{D}_t^{s,i}$ and $\boldsymbol{Y}_t^{s,i}$ denote the solution to the equation system (3)-(6) when all inputs are at their steady state value *except in period* s, where there is an infinitesimal shock dx to input variable i. Let $\Lambda_t^{s,i} = \Lambda(\boldsymbol{v}_{t+1}^{s,i}, X_t^{s,i})$ and $\boldsymbol{y}_t^{s,i} = y(\boldsymbol{v}_{t+1}^{s,i}, X_t^{s,i})$ denote the transition matrix and the individual measures associated with this solution. We can therefore write

$$\boldsymbol{D}_t^{s,i} = \left(\Lambda_{t-1}^{s,i}\right)' \boldsymbol{D}_{t-1}^{s,i} \tag{22}$$

$$\boldsymbol{Y}_t^{s,i} = \left(\boldsymbol{y}_t^{s,i}\right)' \boldsymbol{D}_t^{s,i}. \tag{23}$$

Building blocks. The value function equation (3) is forward looking so $\boldsymbol{v}_t^{s,i} = \boldsymbol{v}_{ss}$ after the shock has arrived, i.e. for $t > s$. Additionally, only the time span until the shock arrives matter so $\boldsymbol{v}_t^{s,i} = \boldsymbol{v}_{t-1}^{s-1,i}$ when $t \leq s$. This carries over to $\Lambda_t^{s,i}$ and $\boldsymbol{y}_t^{s,i}$ such that for all $t, s \geq 1$

$$\boldsymbol{y}_t^{s,i} = \begin{cases} \boldsymbol{y}_{ss} & t > s \\ \boldsymbol{y}_{T-1-(s-t)}^{T-1,i} & t \leq s \end{cases} \text{ and } \Lambda_t^{s,i} = \begin{cases} \Lambda_{ss} & t > s \\ \Lambda_{T-1-(s-t)}^{T-1,i} & t \leq s \end{cases}. \tag{24}$$

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

For later use, we define the following objects:

[2] Should be generalized in future version.

$$\mathcal{Y}_s^{o,i} \equiv \frac{d\boldsymbol{Y}_0^{o,s,i}}{dx} = \frac{(d\boldsymbol{y}_t^{o,s,i})'}{dx}\boldsymbol{D}_{ss}$$
$$\mathcal{D}_s^i \equiv \frac{d\boldsymbol{D}_1^{s,i}}{dx} = \frac{(d\Lambda_0^{s,i})'}{dx}\boldsymbol{D}_{ss}$$
$$\mathcal{E}_t^o \equiv (\Lambda_{ss})^t \, \boldsymbol{y}_{ss}^o,$$

where $\mathcal{Y}_s^{o,i}$ and $\mathcal{D}_s^i$ 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 $\mathcal{Y}_s^{o,i}$ and $\mathcal{D}_s^i$ 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, $\mathcal{E}_t^o = \Lambda_{ss}\mathcal{E}_{t-1}^o$ with $\mathcal{E}_0^o = \boldsymbol{y}_{ss}^o$.

The task is to build the full Jacobian from these building blocks. To do this, we first need to take the total derivative of equations (22) and (23) around the steady state,

$$d\boldsymbol{D}_t^{s,i} = \Lambda_{ss}' d\boldsymbol{D}_{t-1}^{s,i} + (d\Lambda_{t-1}^{s,i})'\boldsymbol{D}_{ss} \tag{25}$$
$$d\boldsymbol{Y}_t^{o,s,i} = (\boldsymbol{y}_{ss}^o)' d\boldsymbol{D}_t^s + (d\boldsymbol{y}_t^{o,s,i})'\boldsymbol{D}_{ss}. \tag{26}$$

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

$$\mathcal{J}_{0,s}^{hh,i,o} = \frac{d\boldsymbol{Y}_0^{o,s,i}}{dx} = \mathcal{Y}_s^{i,o} \tag{27}$$

Next consider the effect on output at time $t \geq 1$ form a shock at time 0. Combining equation (25) with only the time span mattering in equation (24) implies for $t \geq 2$

$$\begin{aligned} d\boldsymbol{D}_t^{0,i} &= \Lambda_{ss}' d\boldsymbol{D}_{t-1}^{0,i} + \underbrace{\left(d\Lambda_{t-1}^{0,i}\right)'}_{=0}\boldsymbol{D}_{ss} \\ &= \Lambda_{ss}' d\boldsymbol{D}_{t-1}^{0,i} \\ &= (\Lambda_{ss}')^{t-1} \, d\boldsymbol{D}_1^{0,i}. \end{aligned} \tag{28}$$

Combining (26) with only the time span mattering in equation (24), and the above equation (31) implies for $t \geq 1$,

$$dY_t^{o,0,i} = (\boldsymbol{y}_{ss}^o)' d\boldsymbol{D}_t^{0,i} + \underbrace{(d\boldsymbol{y}_t^{o,0,i})'}_{=0}\boldsymbol{D}_{ss} = (\boldsymbol{y}_{ss}^o)' d\boldsymbol{D}_t^{0,i} = (\boldsymbol{y}_{ss}^o)' \left(\Lambda_{ss}'\right)^{t-1} d\boldsymbol{D}_1^{0,i}. \tag{29}$$

Combining equation (28) and (29) implies for $t \geq 1$

$$\mathcal{J}_{t,0}^{hh,i,o} = \frac{dY_t^{o,0,i}}{dx} = \left(\mathcal{E}_{t-1}^o\right)' \mathcal{D}_0^i \tag{30}$$

Inner parts of the Jacobian. Combining (25) with only the time span mattering in equation (24), implies for $t, s \geq 1$

$$\begin{aligned} d\boldsymbol{D}_{t+1}^{s,i} - d\boldsymbol{D}_{t}^{s-1,i} &= \Lambda_{ss}' \left(d\boldsymbol{D}_{t}^{s,i} - d\boldsymbol{D}_{t-1}^{s-1,i} \right) + \underbrace{\left(d\Lambda_{t-1}^{s,i} - d\Lambda_{t-2}^{s-1,i} \right)'}_{=0} \boldsymbol{D}_{ss} \\ &= \Lambda_{ss}' \left(d\boldsymbol{D}_{t}^{s,i} - d\boldsymbol{D}_{t-1}^{s-1,i} \right) \\ &= \left(\Lambda_{ss}' \right)^2 \left(d\boldsymbol{D}_{t-2}^{s,i} - d\boldsymbol{D}_{t-2}^{s-1,i} \right) \\ &= \vdots \\ &= \left(\Lambda_{ss}' \right)^{t-1} \left(d\boldsymbol{D}_{1}^{s,i} - \underbrace{d\boldsymbol{D}_{0}^{s-1,i}}_{=0} \right) \\ &= \left(\Lambda_{ss}' \right)^{t-1} d\boldsymbol{D}_{1}^{s,i} \end{aligned} \tag{31}$$

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

Combining (26) with only the time span mattering in equation (24), and the above equation (31), implies for $t, s \geq 1$

$$\begin{aligned} dY_{t}^{o,s,i} - dY_{t-1}^{o,s-1,i} &= \left(\boldsymbol{y}_{ss}^{o} \right)' \left(d\boldsymbol{D}_{t}^{s,i} - d\boldsymbol{D}_{t-1}^{s-1,i} \right) + \underbrace{\left(d\boldsymbol{y}_{t}^{o,s,i} - d\boldsymbol{y}_{t-1}^{o,s-1,i} \right)'}_{=0} \boldsymbol{D}_{ss} \\ &= \left(\boldsymbol{y}_{ss}^{o} \right)' \left(d\boldsymbol{D}_{t}^{s,i} - d\boldsymbol{D}_{t-1}^{s-1,o} \right) \\ &= \left(\boldsymbol{y}_{ss}^{o} \right)' \left(\Lambda_{ss}' \right)^{t-1} d\boldsymbol{D}_{1}^{s} \\ &= \left(\mathcal{E}_{t-1}^{o} \right)' d\boldsymbol{D}_{1}^{s,i} \end{aligned} \tag{32}$$

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

$$\begin{aligned} \mathcal{J}_{t,s}^{hh,i,o} - \mathcal{J}_{t-1,s-1}^{hh,i,o} = \frac{dY_{t}^{o,s,i} - dY_{t-1}^{o,s-1,i}}{dx} \Leftrightarrow \\ \mathcal{J}_{t,s}^{hh,i,o} = \mathcal{J}_{t-1,s-1}^{hh,i,o} + \left(\mathcal{E}_{t-1}^{o} \right)' \mathcal{D}_{s}^{i} \end{aligned} \tag{33}$$

Recursive formulation Define the object

$$\mathcal{F}_{t,s}^{i,o} = \begin{cases} \mathcal{Y}_{s}^{o,i} & t = 0 \\ \left(\mathcal{E}_{t-1}^{o} \right)' \mathcal{D}_{s}^{i} & t \geq 1 \end{cases} \tag{34}$$

Combining equations (27), (30) and (33), the household Jacobian can be written recursively by

$$\mathcal{J}_{t,s}^{i,o} = \sum_{k=0}^{\min\{t,s\}} \mathcal{F}_{t-k,s-k}^{i,o} \tag{35}$$

4 Additional features

The following methods are available:

1. `.show_IRFS((...)`
   Show IRFs.
2. `.compare_IRFS(...)`
   Compare IRFs across models.
3. `.print_unpack_varlist(...)`:
   Print unpacking of all variables for use in `.block_pre(...)` and `.block_post(...)`.
4. `.test_hh_path(...)`:
   Test time-invariance when inputs are at their steady state values.
5. `.test_jac_hh(...)`:
   Compare Jacobians calculated with a direct and the fake news algorithm.
6. `.test_path(...)`:
   Test time-invariance when inputs are at their steady state values.

5 Troubleshooting

The transition path cannot be found. Considering the following

1. Use finer tolerances for finding the steady state
   `par.tol_solve`↓
   `par.tol_simulate`↓
2. Extend the transition period
   `par.T`↑
3. Decrease the size and persistence of the size
   `par.jump_VARNAME`↓
   `par.rho_VARNAME`↓
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.