Solving the Krusell-Smith (1998) model

Jesús Fernández-Villaverde — University of Pennsylvania
Samuel Hurtado — Banco de España
Galo Nuño — Banco de España

October 2019

1 Model: an Aiyagari-Bewley-Hugget economy with aggregate uncertainty

In order to get acquainted with the techniques, we analyze first the workhorse model, a continuous-time version of Krusell-Smith (1998).

Firms. There is a representative firm with a constant returns to scale production function

$$Y_t = e^{Z_t} K_t^{\alpha} L_t^{1-\alpha},$$

where K is the aggregate capital, L is aggregate labor and Z_t is an aggregate TFP shock. We assume that Z_t follows an Ornstein–Uhlenbeck process:

$$dZ_t = \theta \left(\bar{Z} - Z_t \right) dt + \sigma dW_t,$$

where θ, σ and $\bar{Z}$ are positive constants and W_t is a Brownian motion. Capital depreciates at rate δ_K. Since factor markets are competitive, the interest rate and wage are given by

$$\begin{aligned} r_t &= \frac{\partial F(K_t, 1)}{\partial K} - \delta = \alpha \frac{Y_t}{K_t} - \delta, \\ w_t &= \frac{\partial F(K_t, 1)}{\partial L} = (1-\alpha) \frac{Y_t}{L_t}. \end{aligned} \tag{1}$$

Households. There is a continuum of mass unity of infinitely-lived agents that are heterogeneous in their wealth a and labor supply z. Agents have standard preferences over utility flows from future consumption c_t discounted at rate $\rho > 0$. We assume CRRA preferences, such that

$u(c) = \frac{c^{1-\gamma}-1}{1-\gamma}$. The expected discounted utility is

$$U_0 = \mathbb{E}_0 \left[\int_0^\infty e^{-\rho t} u(c_t) dt \right]. \tag{2}$$

When employed, each agent supplies z_t units of labor valued at wage w_t. An agent's wealth evolves according to

$$da_t = [w_t z_t + r_t a_t - c_t] \, dt = s(a_t, z_t, w_t, r_t, c_t) \, dt, \tag{3}$$

where $s(a_t, z_t, w_t, r_t, c_t)$ is the *drift* of the detrended wealth process.

Individual labor productivity evolves stochastically over time following a two-state Markov chain: $z_t \in \{z_1, z_2\}$, with $z_1 < z_2$. The process jumps from state 1 to state 2 with intensity λ_1 and vice versa with intensity λ_2.

Agents also face a borrowing limit,

$$a_t \geq 0. \tag{4}$$

individual agents decide their individual consumption levels. The optimal value function results in

$$V(t, a, z) = \max_{\{c_t\}_{t\geq 0}, a_t \geq 0} U_0(c(\cdot)), \tag{5}$$

subject to evolution of individual wealth (3).

Density. The state of the economy is the joint density of wealth and labor, $f(t, a, z)$. The dynamics of the density are given by the Kolmogorov Forward (KF) equation

$$\frac{\partial f_i}{\partial t} = -\frac{\partial}{\partial a}\left(s(a, z_t, w_t, r_t, c_t) f_i(t, a)\right) - \lambda_i f_i(t, a) + \lambda_j f_j(t, a), \;\; i \neq j = 1, 2, \tag{6}$$

where $f_i(t, a) \equiv f(t, a, z_i)$, $i = 1, 2$. The density satisfies the normalization

$$\sum_{i=1}^{2} \int_0^\infty f_i(t, a) da = 1.$$

Perceived law of motion. Agents consider a perceived law of motion (PLM) of aggregate capital

$$dK_t = h(K_t, Z_t) \, dt,$$

where $h(K, Z)$ is the conditional expectation of the increase in aggregate capital given available information (K_t, Z_t):

$$h(K_t, Z_t) = \frac{\mathbb{E}\left[dK_t | K_t, Z_t\right]}{dt}$$

In the original Krusell-Smith methodology and in most of the literature, the PLM is assumed to be log-linear:

$$d\log(K_t) = [\varpi_0 + \varpi_Z Z_t + \varpi_K \log(K_t) + \varpi_{ZK} Z_t \log(K_t)]\, dt, \tag{7}$$

where ϖ_0, ϖ_Z, ϖ_K, ϖ_{ZK} are positive constants. The value of these constants was then obtained by simulating the economy and running a OLS.

Hamiton-Jacobi-Bellman equation. Instead here we propose a more flexible methodology, in which the functional form $h(\cdot,\cdot)$ is not specified but obtained directly from the simulated data by employing a neural network. The household's HJB in this case is

$$\begin{aligned}\rho V_i(a,K,Z) \quad &= \quad \max_{c,a\geq 0} \frac{c^{1-\gamma}-1}{1-\gamma} + s_i(a,K,Z,c)\frac{\partial V_i}{\partial a} + \lambda_i\left[V_j(a,K,Z) - V_i(a,K,Z)\right] \\ &\quad + h(K,Z)\frac{\partial V_i}{\partial K} + \theta\left(\bar{Z} - Z_t\right)\frac{\partial V_i}{\partial Z} + \frac{\sigma^2}{2}\frac{\partial^2 V_i}{\partial Z^2}\ ,\end{aligned}$$

$i \neq j = 1,2$.

Market clearing. There are two market clearing conditions. First, the total amount of capital supplied in the economy equals the total amount of wealth

$$K_t = \sum_{i=1}^{2} \int_0^\infty a f_i(t,a) da. \tag{8}$$

Second, the total amount of labor supplied in the economy equals the supply,

$$L_t = \sum_{i=1}^{2} \int_0^\infty z f_i(t,a) da.$$

Notice that we make the parametric assumptions about the ergodic mean of z such that $L_t = 1$.

2 Algorithm

We describe the numerical algorithm used to jointly solve for the equilibrium value function, $v(a,z,K,Z)$, the density $f(a,z,K,Z)$ and the PLM $h(K,Z)$. The algorithm proceeds in 3 steps. We describe each step in turn.

2.1 Step 1: Solution to the Hamilton-Jacobi-Bellman equation

The HJB equation is solved using an *upwind finite difference* scheme similar to Achdou et al. (2017). It approximates the value function $v_i(a, Z.K)$, $i = 1,2$ on a finite grid with steps Δa,

ΔK and ΔZ : $a \in \{a_1, ..., a_J\}$, where $a_j = a_{j-1} + \Delta a = a_1 + (j-1)\Delta a$ for $2 \leq j \leq J$ and similarly for $K \in \{K_1, ..., K_L\}$, $Z \in \{Z_1, ..., Z_M\}$. The bounds are $a_1 = 0$ and $a_I = \varkappa$, such that $\Delta a = \varkappa/(J-1)$. We use the notation $v_{i,j,l,m} \equiv v_i(a_j, K_l, Z_m)$, $i = 1, 2$, and similarly for the policy function $c_{i,j}$.

Notice first that the HJB equation involves first and second derivatives of the value function, v. At each point of the grid, the first order derivatives with respect to a can be approximated with a forward (F) or a backward (B) approximation,

$$\frac{\partial_i v(a_j, K_l, Z_m)}{\partial a} \approx \partial_F v_{i,j,l,m} \equiv \frac{v_{i,j+1,l,m} - v_{i,j,l,m}}{\Delta a}, \tag{9}$$

$$\frac{\partial_i v(a_j, K_l, Z_m)}{\partial a} \approx \partial_B v_{i,j,l,m} \equiv \frac{v_{i,j,l,m} - v_{i,j-1,l,m}}{\Delta a}. \tag{10}$$

In an upwind scheme, the choice of forward or backward derivative depends on the sign of the *drift function* for the state variable, given by

$$s_i(a) \equiv wz_i + ra - c_i(a), \tag{11}$$

for $\phi \leq a \leq 0$, where

$$c_i(a) = [v_i'(a)]^{-1/\gamma}. \tag{12}$$

The derivatives with respect to the aggregate variables are approximated as

$$\frac{\partial_i v(a_j, K_l, Z_m)}{\partial K} \approx \partial_K v_{i,j,l,m} \equiv \frac{v_{i,j,l+1,m} - v_{i,j,l,m}}{\Delta K},$$

$$\frac{\partial_i v(a_j, K_l, Z_m)}{\partial Z} \approx \partial_Z v_{i,j,l,m} \equiv \frac{v_{i,j,l,m+1} - v_{i,j,l,m}}{\Delta Z},$$

$$\frac{\partial_i^2 v(a_j, K_l, Z_m)}{\partial Z^2} \approx \partial_{ZZ}^2 v_{i,j,l,m} \equiv \frac{v_{i,j,l,m+1} + v_{i,j,l,m-1} - 2v_{i,j,l,m}}{(\Delta Z)^2}$$

Let superscript n denote the iteration counter. The HJB equation is approximated by the following upwind scheme,

$$\begin{aligned}\frac{v_{i,j,l,m}^{n+1} - v_{i,j,l,m}^n}{\Delta} + \rho v_{i,j,l,m}^{n+1} &= \frac{(c_{i,j,n,m}^n)^{1-\gamma} - 1}{1-\gamma} + \partial_F v_{i,j,l,m}^{n+1} s_{i,j,l,m,F}^n \mathbf{1}_{s_{i,j,n,mF}^n > 0} + \partial_B v_{i,j,l,m}^{n+1} s_{i,j,l,m,B}^n \mathbf{1}_{s_{i,j,l,m,B}^n < 0} \\ &\quad + \lambda_i \left(v_{-i,j,l,m}^{n+1} - v_{i,j,l,m}^{n+1}\right) + h_{l,m} \partial_K v_{i,j,l,m} + \theta\left(\bar{Z} - Z_m\right) \partial_Z v_{i,j,l,m} + \frac{\sigma^2}{2} \partial_{ZZ}^2 v_{i,j,l,m}\end{aligned}$$

for $i = 1, 2$, $j = 1, ..., J$, $l = 1, .., L$, $m = 1, ..., M$, where $v_{i,j,l,m}^n \equiv v^n(a_j, z_i, K_l, Z_m)$, $h_{l,m} \equiv$

$h(K_l, Z_m)$, $\mathbf{1}(\cdot)$ is the indicator function and

$$s^n_{i,j,n,m,F} = w_{l,m} z_i + r_{l,m} a_j - \left[\frac{1}{\partial_F v^n_{i,j,l,m}} \right]^{1/\gamma}, \tag{13}$$

$$s^n_{i,j,n,m,B} = w_{l,m} z_i + r_{l,m} a_j - \left[\frac{1}{\partial_B v^n_{i,j,l,m}} \right]^{1/\gamma}. \tag{14}$$

Therefore, when the drift is positive ($s^n_{i,j,n,m,F} > 0$) we employ a forward approximation of the derivative, $\partial_F v^n_{i,j,l,m}$; when it is negative ($s^n_{i,j,n,m,B} < 0$) we employ a backward approximation, $\partial_B v^n_{i,j,l,m}$. The term $\frac{v^{n+1}_{i,j,l,m} - v^n_{i,j,l,m}}{\Delta} \to 0$ as $v^{n+1}_{i,j,l,m} \to v^n_{i,j,l,m}$. Notice how interest rates and wages depend on aggregate variables

$$\begin{aligned} r_{l,m} &= \alpha K_l^{\alpha-1} Z_m - \delta, \\ w_{l,m} &= (1-\alpha) K_l^{\alpha} Z_m. \end{aligned}$$

Moving all terms involving v^{n+1} to the left hand side and the rest to the right hand side, we obtain

$$\begin{aligned} \frac{v^{n+1}_{i,j,l,m} - v^n_{i,j,l,m}}{\Delta} + \rho v^{n+1}_{i,j,l,m} &= \frac{\left(c^n_{i,j,n,m}\right)^{1-\gamma}}{1-\gamma} + v^{n+1}_{i,j-1,l,m} \alpha^n_{i,j,l,m} + v^{n+1}_{i,j,l,m} \beta^n_{i,j,l,m} + v^{n+1}_{i,j+1,l,m} \xi^n_{i,j,l,m} + \lambda_i v^{n+1}_{-i,j,l,m} \\ &\quad + v^{n+1}_{i,j,l+1,m} \frac{h_{l,m}}{\Delta K} + v^{n+1}_{i,j,l,m+1} \varkappa_m + v^{n+1}_{i,j,l,m-1} \varrho \end{aligned} \tag{15}$$

where

$$\begin{aligned} \alpha^n_{i,j,l,m} &\equiv -\frac{s^n_{i,j,l,m,B} \mathbf{1}_{s^n_{i,j,l,m,B} < 0}}{\Delta a}, \\ \beta^n_{i,j,l,m} &\equiv -\frac{s^n_{i,j,l,m,F} \mathbf{1}_{s^n_{i,j,n,m,F} > 0}}{\Delta a} + \frac{s^n_{i,j,l,m,B} \mathbf{1}_{s^n_{i,j,l,m,B} < 0}}{\Delta a} - \lambda_i - \frac{h_{l,m}}{\Delta K} - \frac{\theta\left(\bar{Z} - Z_m\right)}{\Delta Z} - \frac{\sigma^2}{\left(\Delta Z\right)^2}, \\ \xi^n_{i,j,l,m} &\equiv \frac{s^n_{i,j,l,m,F} \mathbf{1}_{s^n_{i,j,n,m,F} > 0}}{\Delta a}, \\ \varkappa_m &\equiv \frac{\theta\left(\bar{Z} - Z_m\right)}{\Delta Z} + \frac{\sigma^2}{2\left(\Delta Z\right)^2}, \\ \varrho &\equiv \frac{\sigma^2}{2\left(\Delta Z\right)^2}, \end{aligned}$$

for $i = 1, 2$, $j = 1, ..., J$, $l = 1, .., L$, $m = 1, ..., M$. Notice that the state constraints $\phi \leq a \leq 0$ mean that $s_{i,1,B} = s_{i,J,F} = 0$. We consider reflections in K and Z such that

$$\partial_K v_{i,j,1,m} = \partial_K v_{i,j,L,m} = \partial_Z v_{i,j,l,1} = \partial_Z v_{i,j,l,M} = 0.$$

In equation (15), the optimal consumption is set to

$$c^n_{i,j,l,m} = \left(\partial v^n_{i,j,l,m}\right)^{-1/\gamma}. \tag{16}$$

where

$$\partial v^n_{i,j,l,m} = \partial_F v^n_{i,j,l,m} \mathbf{1}_{s^n_{i,j,F}>0} + \partial_B v^n_{i,j,l,m} \mathbf{1}_{s^n_{i,j,B}<0} + \partial \bar{v}^n_{i,j,l,m} \mathbf{1}_{s^n_{i,F}\leq 0} \mathbf{1}_{s^n_{i,B}\geq 0}.$$

In the above expression, $\partial \bar{v}^n_{i,j,l,m} = (\bar{c}^n_{i,j,l,m})^{-\gamma}$ where $\bar{c}^n_{i,j,l,m}$ is the consumption level such that $s(a_i) \equiv s^n_i = 0$:

$$\bar{c}^n_{i,j,l,m} = w_{l,m} z_i + r_{l,m} a_j.$$

Equation (15) is a system of $2 \times J \times L \times M$ linear equations which can be written in matrix notation as:

$$\frac{1}{\Delta}\left(\mathbf{v}^{n+1} - \mathbf{v}^n\right) + \rho \mathbf{v}^{n+1} = \mathbf{u}^n + \mathbf{A}^n \mathbf{v}^{n+1}$$

Matrix $\mathbf{A}^n$ and the vectors v^{n+1} and $\mathbf{u}^n$ are defined by

$$\mathbf{A}^n = -\begin{bmatrix} (\mathbf{A}^n_1 + \varrho \mathbf{I}_{2J\times L}) & \varkappa_1 \mathbf{I}_{2J\times L} & \mathbf{0}_{2J\times L} & \cdots & \mathbf{0}_{2J\times L} & \mathbf{0}_{2J\times L} \\ \varrho \mathbf{I}_{2J\times L} & \mathbf{A}^n_2 & \varkappa_2 \mathbf{I}_{2J\times L} & \cdots & \mathbf{0}_{2J\times L} & \mathbf{0}_{2J\times L} \\ \mathbf{0}_{2J\times L} & \varrho \mathbf{I}_{2J\times L} & \mathbf{A}^n_3 & \cdots & \mathbf{0}_{2J\times L} & \mathbf{0}_{2J\times L} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ & & & \varrho \mathbf{I}_{2J\times L} & \mathbf{A}^n_{M-1,m} & \varkappa_{M-1} \mathbf{I}_{2J\times L} \\ \mathbf{0}_{2J\times L} & \mathbf{0}_{2J\times L} & \cdots & \mathbf{0}_{2J\times L} & \varrho \mathbf{I}_{2J\times L} & (\mathbf{A}^n_M + \varkappa_M \mathbf{I}_{2J\times L}) \end{bmatrix},$$

$$\mathbf{u}^n = \begin{bmatrix} \frac{(c^n_{1,1})^{1-\gamma}}{1-\gamma} \\ \frac{(c^n_{1,2})^{1-\gamma}}{1-\gamma} \\ \vdots \\ \frac{(c^n_{1,J})^{1-\gamma}}{1-\gamma} \\ \frac{(c^n_{2,1})^{1-\gamma}}{1-\gamma} \\ \vdots \\ \frac{(c^n_{2,J})^{1-\gamma}}{1-\gamma} \end{bmatrix}, \mathbf{v}^{n+1} = \begin{bmatrix} \mathbf{v}^{n+1}_1 \\ \mathbf{v}^{n+1}_2 \\ \vdots \\ \mathbf{v}^{n+1}_M \end{bmatrix},$$

where $\mathbf{I}_n$ and $\mathbf{0}_n$ are the identity matrix and the zero matrix of dimension $n \times n$, respectively, and

$$\mathbf{A}_m^n = -\begin{bmatrix} \mathbf{A}_{1,m}^n & \frac{h_{1,m}}{\Delta K}\mathbf{I}_{2J} & \mathbf{0}_{2J} & \cdots & \mathbf{0}_{2J} & \mathbf{0}_{2J} \\ \mathbf{0}_{2J} & \mathbf{A}_{2,m}^n & \frac{h_{2,m}}{\Delta K}\mathbf{I}_{2J} & \cdots & \mathbf{0}_{2J} & \mathbf{0}_{2J} \\ \mathbf{0}_{2J} & \mathbf{0}_{2J} & \mathbf{A}_{3,m}^n & \cdots & \mathbf{0}_{2J} & \mathbf{0}_{2J} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ & & & \mathbf{0}_{2J} & \mathbf{A}_{L-1,m}^n & \frac{h_{L-1,m}}{\Delta K}\mathbf{I}_{2J} \\ \mathbf{0}_{2J} & \mathbf{0}_{2J} & \cdots & \mathbf{0}_{2J} & \mathbf{0}_{2J} & \left(\mathbf{A}_{L,m}^n + \frac{h_{L,m}}{\Delta K}\mathbf{I}_{2J}\right) \end{bmatrix}, \quad \mathbf{v}_m^{n+1} = \begin{bmatrix} \mathbf{v}_{1,m}^{n+1} \\ \mathbf{v}_{2,m}^{n+1} \\ \vdots \\ \mathbf{v}_{L,m}^{n+1} \end{bmatrix},$$

$$\mathbf{A}_{l.m}^n = -\begin{bmatrix} \beta_{1,1,l,m}^n & \xi_{1,1,l,m}^n & 0 & 0 & \cdots & 0 & \lambda_1 & 0 & \cdots & 0 \\ \alpha_{1,2,l,m}^n & \beta_{1,2,l,m}^n & \xi_{1,2,l,m}^n & 0 & \cdots & 0 & 0 & \lambda_1 & \ddots & 0 \\ 0 & \alpha_{1,3,l,m}^n & \beta_{1,3,l,m}^n & \xi_{1,3,l,m}^n & \cdots & 0 & 0 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & \alpha_{1,J-1,l,m}^n & \beta_{1,J-1,l,m}^n & \xi_{1,J-1,l,m}^n & 0 & \cdots & \lambda_1 & 0 \\ 0 & 0 & \cdots & 0 & \alpha_{1,J,l,m}^n & \beta_{1,J,l,m}^n & 0 & 0 & \cdots & \lambda_1 \\ \lambda_2 & 0 & \cdots & 0 & 0 & 0 & \beta_{2,1,l,m}^n & \xi_{2,1,l,m}^n & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & \lambda_2 & 0 & \cdots & \alpha_{2,J,l,m}^n & \beta_{2,J,l,m}^n \end{bmatrix},$$

$$\mathbf{v}_{l,m}^{n+1} = \begin{bmatrix} \mathbf{v}_{1,1,l,m}^{n+1} \\ \mathbf{v}_{1,2,l,m}^{n+1} \\ \vdots \\ \mathbf{v}_{1,J,l,m}^{n+1} \\ \mathbf{v}_{2,1,l,m}^{n+1} \\ \vdots \\ \mathbf{v}_{2,J,l,m}^{n+1} \end{bmatrix}.$$

The system in turn can be written as

$$\mathbf{B}^n \mathbf{v}^{n+1} = \mathbf{d}^n \tag{17}$$

where , $\mathbf{B}^n = \left(\frac{1}{\Delta} + \rho\right)\mathbf{I} - \mathbf{A}^n$ and $\mathbf{d}^n = \mathbf{u}^n + \frac{1}{\Delta}\mathbf{v}^n$.

The algorithm to solve the HJB equation runs as follows. Begin with an initial guess $\{v_{i,j}^0\}_{j=1}^J$, $i = 1, 2$. Set $n = 0$. Then:

1. Compute $\{\partial_F v_{i,j}^n, \partial_B v_{i,j}^n\}_{j=1}^J$, $i = 1, 2$ using (9)-(10).

2. Compute $\{c_{i,j}^n\}_{j=1}^J$, $i=1,2$ using (12) as well as $\{s_{i,j,F}^n, s_{i,j,B}^n\}_{j=1}^J$, $i=1,2$ using (13) and (14).

3. Find $\{v_{i,j}^{n+1}\}_{j=1}^J$, $i=1,2$ solving the linear system of equations (17).

4. If $\{v_{i,j}^{n+1}\}$ is close enough to $\{v_{i,j}^{n+1}\}$, stop. If not set $n := n+1$ and proceed to 1.

Most computer software packages, such as Matlab, include efficient routines to handle sparse matrices such as $\mathbf{A}^n$.

2.2 Step 2: Solution to the Kolmogorov Forward equation

The instantaneous change in the income-wealth density $f_t(a,z)$ (given the aggregate variables K and Z) is given by the Kolmogorov Forward equation:

$$\frac{\partial}{\partial t} f_{t,i}(a,K,Z) = -\frac{d}{da}\left[s_i(a) f_{t,i}(a,K,Z)\right] - \lambda_i f_{t,i}(a,K,Z) + \lambda_{-i} f_{t,-i}(a,K,Z), \; i=1,2. \tag{18}$$

We also solve this equation using an finite difference scheme. We use the notation $f_{i,j,l,m} \equiv f_i(a_j|K_l,Z_m)$, as the density is conditional on the current state of K and Z. The system can be now expressed as

$$\begin{aligned} \frac{f_{t+1,i,j,l,m} - f_{t,i,j,l,m}}{\Delta t} &= -\frac{f_{t,i,j,l,m} s_{i,j,l,m,F} \mathbf{1}_{s_{i,j,n,m,F}>0} - f_{t,i,j-1,l,m} s_{i,j-1,l,m,F} \mathbf{1}_{s_{i,j-1,n,m,F}>0}}{\Delta a} \\ &\quad - \frac{f_{t,i,j+1,l,m} s_{i,j+1,l,m,B} \mathbf{1}_{s_{i,j+1,l,m,B}<0} - f_{t,i,j,l,m} s_{i,j,l,m,B}^n \mathbf{1}_{s_{i,j,l,m,B}^n<0}}{\Delta a} \\ &\quad - \lambda_i f_{t,i,j,l,m} + \lambda_{-i} f_{t,-i,j,l,m}, \end{aligned}$$

where we have defined the time step Δt. This equation can be expressed as

$$\mathbf{f}_{t+1} = \left(\mathbf{I} - \Delta t \mathbf{A}_{l,m}^{\mathbf{T}}\right)^{-1} \mathbf{f}_t,$$

where $\mathbf{A}_{l,m}^{\mathbf{T}}$ is the transpose matrix of $\mathbf{A}_{l,m} = \lim_{n\to\infty} \mathbf{A}_{l,m}^n$, defined above and

$$\mathbf{f}_t = \begin{bmatrix} f_{1,1,t} \\ f_{1,2,t} \\ \vdots \\ f_{1,J,t} \\ f_{2,1,t} \\ \vdots \\ f_{2,J,t} \end{bmatrix}.$$

We assume that the initial density $\mathbf{f}_0$ is the the one in the deterministic steady state.

2.3 Step 3: Update of the PLM

The algorithm to update the PLM $h(K, Z)$ proceeds as follows. First, we simulate T periods of the economy with a constant time step Δt.[1] If the time step is small enough then

$$K_{t_k+\Delta t} = K_{t_k} + \int_{t_k}^{t_k+\Delta t} dK_s = K_{t_k} + \int_{t_k}^{t_k+\Delta t} h\left(K_s, Z_s\right) ds \approx K_{t_k} + h\left(K_{t_k}, Z_{t_k}\right) \Delta t.$$

We consider a vector of inputs

$$\mathbf{X} = \left\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, ..., \mathbf{x}^{(K)}\right\},$$

where $\mathbf{x}^{(k)} = \left\{x_1^{(k)}, x_2^{(k)}\right\} = \{K_{t_k}, Z_{t_k}\}$ are samples of aggregate debt and equity at random times $t_k \in [0, T]$, and a vector of outputs

$$\mathbf{h} = \left\{\hat{h}_1, \hat{h}_2 ..., \hat{h}_K\right\},$$

where $\hat{h}_k \equiv \frac{K_{t_k+\Delta t} - K_{t_k}}{\Delta t}$ are samples of the growth rate of capital.

2.3.1 Regression-based approximation (KS-LR)

In the case of the original Krusell-Smith methodology (KS-LR version), the PLM is assumed to be linear in the endogenous variable K and nonlinear in Z. It can be computed with a linear regression to approximate equation (7), using a constant, $\log(K)$, Z and $Z \log(K)$ as regressors.

2.3.2 Neural-network-based approximation (KS-NN)

In the KS-NN version, a neural network is used instead to approximate the PLM. Before working with the neural network, we introduce a pre-processing phase that avoids the need to use stochastic gradient descent (SGD) to find the coefficients. While SGD has proved to be a fairly successful algorithm in training large neural networks, in our particular setting it is a source of simulation noise that may increase the numerical error of the model or to decrease its speed. To this end, we define a finer 2-dimensional grid on the state space. We label all our data points according to the closest grid point. Then, for each of these subgroups, we run a linear regression and we use the obtained coefficients to infer the value of the PLM *exactly* at the grid point. Then we replace our large dataset $\mathbf{X}$ by the new synthetic dataset comprising the estimated values of the PLM in

[1] The initial income-wealth distribution is that at the deterministic steady state. A number of initial samples is discarded.

the grid points $\hat{\mathbf{X}}$. These inputs are normalized, to avoid scale problems, so that they lie in the domain $[-1, 1]$.

Instead of a SGD algorithm with minibatches, we employ a gradient descent algorithm over the whole batch of data. The single hidden layer neural network is a linear combination of fixed nonlinear basis functions $\phi(\cdot)$:

$$h(\mathbf{x}; \boldsymbol{\theta}) = \sum_{j=1}^{J} \theta_j^{(2)} \phi\left(\sum_{i=1}^{2} \theta_{i,j}^{(1)} x_i + \theta_{0,j}^{(1)}\right) + \theta_0^{(2)}, \tag{19}$$

where $\phi(\cdot)$ is an activation function and

$$\boldsymbol{\theta} = \left(\theta_{1,0}^{(1)}, \theta_{1,1}^{(1)}, ..\theta_{1,J}^{(1)}, \theta_{2,0}^{(1)}, \theta_{2,1}^{(1)}, ..\theta_{2,J}^{(1)}, \theta_0^{(2)}, .., \theta_J^{(2)}\right),$$

is a vector of parameters. Different alternatives are typically proposed for the activation function. In our case we choose a *softplus* function, $\phi(x) = \log(1 + e^x)$ because it provides a smooth approximation function with good extrapolation properties. The constant J determines the size of the hidden layer. This is a hyperparameter that can be set by regularization techniques or simply by trial-and-error in relatively simple problems, such as the one presented here.

The neural network provides a *flexible* parametric function h such that

$$h\left(\mathbf{x}^{(k)}; \boldsymbol{\theta}\right) = \hat{h}_k, \;\; k = 1, .., K.$$

The vector of parameters is estimated based on the sample data $(\mathbf{X}, \mathbf{h})$ in order to minimize the quadratic error function $E(\boldsymbol{\theta}) := E(\boldsymbol{\theta}; \mathbf{X}, \mathbf{h})$:

$$\boldsymbol{\theta}^* = \arg\max_{\boldsymbol{\theta}} E(\boldsymbol{\theta}) = \arg\max_{\boldsymbol{\theta}} \sum_{k=1}^{K} \left\| h\left(\mathbf{x}^{(k)}; \boldsymbol{\theta}\right) - \hat{h}_k \right\|^2.$$

We perform this minimization using a *gradient descent* algorithm: the vector of parameters $\boldsymbol{\theta}$ is recursively updated according to

$$\boldsymbol{\theta}_{m+1} = \boldsymbol{\theta}_m - \epsilon_m \nabla E(\boldsymbol{\theta}_m),$$

where $\nabla E(\boldsymbol{\theta}_m)$ is the gradient of the error function

$$\nabla E(\boldsymbol{\theta}) \equiv \left[\frac{\partial E}{\partial \theta_{1,0}^{(1)}}, \frac{\partial E}{\partial \theta_{1,1}^{(1)}}, ..., \frac{\partial E}{\partial \theta_J^{(2)}}\right]^\top.$$

evaluated in the whole batch $E(\boldsymbol{\theta}) = \sum_{k=1}^{K} \left\| h\left(\mathbf{x}^{(k)}; \boldsymbol{\theta}\right) - \hat{h}_k \right\|^2$. The learning rate $\epsilon_m > 0$ is selected in each iteration according to a *line-search* algorithm in order to minimize the error function in the direction of the gradient. An advantage of neural networks is that the error gradient can be efficiently evaluated using a *back-propagation* algorithm, which builds on the standard chain rule of differential calculus. In our case, this results in

$$
\begin{aligned}
\frac{\partial E_k}{\partial \theta_0^{(2)}} &= \left(h\left(\mathbf{x}^{(k)}; \boldsymbol{\theta}\right) - \hat{h}_k\right), \\
\frac{\partial E_k}{\partial \theta_j^{(2)}} &= \left(h\left(\mathbf{x}^{(k)}; \boldsymbol{\theta}\right) - \hat{h}_k\right) \phi\left(\sum_{i=1}^{2} \theta_{i,j}^{(1)} x_i^{(k)} + \theta_{0,j}^{(1)}\right), \\
\frac{\partial E_k}{\partial \theta_{0,j}^{(1)}} &= \theta_j^{(2)} \left(h\left(\mathbf{x}^{(k)}; \boldsymbol{\theta}\right) - \hat{h}_k\right) \phi'\left(\sum_{i=1}^{2} \theta_{i,j}^{(1)} x_i^{(k)} + \theta_{0,j}^{(1)}\right), \\
\frac{\partial E_k}{\partial \theta_{i,j}^{(1)}} &= x_i^{(k)} \theta_j^{(2)} \left(h\left(\mathbf{x}^{(k)}; \boldsymbol{\theta}\right) - \hat{h}_k\right) \phi'\left(\sum_{i=1}^{2} \theta_{i,j}^{(1)} x_i^{(k)} + \theta_{0,j}^{(1)}\right),
\end{aligned}
$$

where $\phi'(x) = \frac{1}{(1+e^{-x})}$.

In the first step of the algorithm ($s = 1$, see next), we draw 10 random initializations of the parameters from a known a random value Θ (we use a Gaussian), $\boldsymbol{\theta}_0 \sim \Theta$, and select the best-performing one, whereas in posterior steps ($s > 1$) we consider as a first guess the network parameteres resulting from the previous iteration.

2.4 Complete algorithm

The algorithm proceeds as follows. We begin a guess of the PLM $h^0(K, Z)$. Set $s := 1$:

Step 1: Household problem. Given $h^{s-1}(K, Z)$, solve the HJB equation to obtain an estimate of the value function $\mathbf{v}$ and of the matrix $\mathbf{A}$. In the code, file `b3_HJB.m` solves the HJB equation.

Step 2: Distribution. Given $\mathbf{A}$, simulate T periods of the economy using the KF equation and obtain the aggregate capital $\{K_t\}_{t=0}^{T}$.[2] File `b5_KFE.m` solves the Kolmogorov forward equation.

Step 3: PLM. Updates the PLM: h^s. This is done in `b7_PLM`. If $\|h^s - h^{s-1}\| < \varepsilon$, where ε is a small positive constant, then stop. if not return to Step 1.

[2]Notice that in the code we consider a linear interpolation scheme to improve the accuracy of the algorithm.

2.5 Description of scripts

2.5.1 Krusell-Smith: regression

Name	Description
`a2_launch.m`	Launches the program
`b1_parameters.m`	Stores all the parameters of the model
`b2_Klm.m`	Runs the main loop
`b3_HJB.m`	Solves the HJB equation
`b5_KFE.m`	Solves the KF equation
`b7_PLM.m`	Updates the PLM using a regression

2.5.2 Krusell-Smith: neural network

Name	Description
`a2_launch.m`	Launches the program
`b1_parameters.m`	Stores all the parameters of the model
`b2_Klm.m`	Runs the main loop
`b3_HJB.m`	Solves the HJB equation
`b5_KFE.m`	Solves the KF equation
`b7_PLM.m`	Updates the PLM using a neural network
`b7_PLM_iter.m`	Subroutine of `b7_PLM.m` in charge of running the line search
`b9_plot.m`	Creates basic plots to assess convergence and performance
`f1_NN_loss.m`	Computes the loss function $E(\boldsymbol{\theta})$ of the neural network
`f2_NN_eval.m`	Evaluates the output of the neural network $h(K, Z)$ for a given set of data
`f5_NN_gradient.m`	Computes the gradient $\nabla E(\boldsymbol{\theta})$ of the neural network

References

[1] Achdou, Y., J. Han, J.-M. Lasry, P.-L. Lions and B. Moll (2017), "Income and Wealth Distribution in Macroeconomics: A Continuous-Time Approach," mimeo.

[2] Krusell, P. and Smith, A. A. (1998). Income and wealth heterogeneity in the macroeconomy.J*ournal of Political Economy*, 106(5):867-896