Continuous Time Model for Pension Reform

Kai-Jyun Wang[*]

Fall 2024

1. Model

Time is continuous and starts from $t = 0$ to T. The model consists of four states: asset a_t, income level y_t, labor supply $\{e, u\}$ in extensive margin. The utility flow is given by $u^e(c) = \frac{\alpha_e}{1-\gamma}c^{1-\gamma}$ for employed agents and $u^u(c) = \frac{1}{1-\gamma}c^{1-\gamma}$ for unemployed. The agents die at rate h_t, when the death occurs, the agents receive utility from bequest $b(a) = \frac{\alpha_b}{1-\gamma}(\kappa + a)^{1-\gamma}$.

The asset states and the income level are governed by the following laws of motions:

$$da_t = [ra_t + y_t - c_t]\,dt, \tag{1}$$

$$dy_t = y_t(\exp(\mu_t + \sigma e_t) - 1)dJ_t, \tag{2}$$

where J_t is the Poisson process. y_t is modelled as a compound Poisson process with intensity λ_y and e_t drawn from $N(0, 1)$. If the agent is unemployed, then $y_t = y_f$ is fixed.

Now assume that the quit opportunity comes at rate λ^e and the job offer arrives at rate λ^u with the income level drawn from F. We can write the value function recursively as

$$\begin{aligned}&(\rho + h_t)\begin{bmatrix} V^e(t, a_t, y_t) \\ V^u(t, a_t)\end{bmatrix} \\ &= \begin{bmatrix} \max_{c_t^e} u^e(c_t^e) + h_t b(a_t) + \lambda_y \int V^e(t, a_t, y_t \exp(\mu_t + \sigma e)) - V^e(t, a_t, y_t) d\Phi(e) \\ +\lambda^e \max\{V^u(t, a_t) - V^e(t, a_t, y_t), 0\} + V_t^e + V_a^e(ra_t + y_t - c_t^e) \\ \max_{c_t^u} u^u(c_t^u) + h_t b(a_t) + \lambda^u \int \max\{V^e(t, a_t, y) - V^u(t, a_t), 0\}\,dF(y) + V_t^u + V_a^u(ra_t + y_f - c_t^u)\end{bmatrix}.\end{aligned} \tag{3}$$

Solving the maximization problem in equation (3) gives us the optimal consumption when the asset is not binding, i.e., $a > 0$. The first-order conditions yield that

$$V_a^e = \alpha_e c^{-\gamma}, \quad V_a^u = c^{-\gamma} \quad \Rightarrow \quad c^e = \left(\frac{\alpha_e}{V_a^e}\right)^{1/\gamma}, \quad c^u = \left(\frac{1}{V_a^u}\right)^{1/\gamma}. \tag{4}$$

[*]National Taiwan University, Department of Economics.

Hence,

$$
\begin{aligned}
(\rho + h_t + \lambda_y + \lambda^e)V^e = \frac{\alpha_e}{1-\gamma}\left(\frac{\alpha_e}{V_a^e}\right)^{1/\gamma-1} &+ h_t b(a) + \lambda_y \int V^e(t,a,y\exp(\mu_t + \sigma e))d\Phi(e) \\
&+ \lambda^e \max\{V^u, V^e\} + V_t^e + V_a^e\left(ra + y - \left(\frac{\alpha_e}{V_a^e}\right)^{1/\gamma}\right), \qquad (5)
\end{aligned}
$$

$$
\begin{aligned}
(\rho + h_t + \lambda^u)V^u = \frac{1}{1-\gamma}\left(\frac{1}{V_a^u}\right)^{1/\gamma-1} &+ h_t b(a) + \lambda^u \int \max\{V^e(t,a,y), V^u(t,a)\}\, dF(y) \\
&+ V_t^u + V_a^u\left(ra + y_f - \left(\frac{1}{V_a^u}\right)^{1/\gamma}\right). \qquad (6)
\end{aligned}
$$

The terminal conditions are given by the bequest.

$$
\lim_{t\to T} V^e(t,a,y) = b(a), \quad \lim_{t\to T} V^u(t,a) = b(a) \text{ for all } a, y. \qquad (7)
$$

Also, the asset constraint leads to that

$$
V_a^e(t,0,y) \geq (u^e)'(y) = \alpha_e y^{-\gamma}, \quad V_a^u(t,0) \geq (u^u)'(y_f) = y_f^{-\gamma} \text{ for all } t, y. \qquad (8)
$$

Note that the V_a may be negative while solving it numerically, which becomes problematic. To prevent such issue, we can augment the above equation.

$$
f^e = \log(c^e) = \frac{1}{\gamma}\log(\alpha_e) - \frac{1}{\gamma}\log(V_a^e).
$$

So

$$
V_a^e = \log(\alpha_e) - \gamma f^e
$$

2. Solution Method

We are going to solve the model using PINN (Physics-Informed Neural Network). Since the model is a time-inhomogeneous PDE, we will assume that for sufficiently large T, the value function is closed enough to the bequest utility $b(a)$. Also, to avoid the local minimum, we first train the neural network on the domain closed to the terminal time and then gradually expand the domain to the whole time interval.

The PINN loss function is composed of three parts,

$$\mathcal{L}_{DE} = \frac{1}{N_{DE}} \sum_{i=1}^{N_{DE}} E_1(x_i) + E_2(x_i), \tag{9}$$

$$\begin{aligned}\mathcal{L}_a = \frac{1}{N_a} \sum_{i=1}^{N_a} &\left|V_a^e(x_i^a) - \alpha_e(c^e(x_i^a))^{-\gamma}\right| + \left|V_a^u(x_i^a) - (c^u(x_i^a))^{-\gamma}\right| \\ &+ \max\left\{0, c^e(x_i^a) - y_i\right\}^2 + \max\left\{0, c^u(x_i^a) - y_f\right\}^2,\end{aligned} \tag{10}$$

$$\begin{aligned}\mathcal{L}_t = \frac{1}{N_t} \sum_{i=1}^{N_t} &\left|V^e(x_i^t) - b(a_i)\right|^2 + \left|V^u(x_i^t) - b(a_i)\right|^2 \\ &+ \left|V_a^e(x_i^t) - \alpha_e(c^e(x_i^t))^{-\gamma}\right| + \left|V_a^u(x_i^t) - (c^u(x_i^t))^{-\gamma}\right| \\ &+ \left|b'(a_i) - \alpha_e(c^e(x_i^a))^{-\gamma}\right|^2 + \left|b'(a_i) - (c^u(x_i^a))^{-\gamma}\right|^2,\end{aligned} \tag{11}$$

with E_1 and E_2 defined as the residuals of the PDEs equations (5) and (6). To avoid the issue that the approximator may have negative derivatives so that the residuals are not properly defined, we use the consumption combined with the first order conditions to compute the residuals and add loss terms to ensure that the first order conditions are satisfied. [1]

2.1. Hard-Constraint Boundary

Sometimes the neural network may struggle to learn the boundary conditions. In such case, we use the hard-constraint method. By taking some transformation of our neural network, we may enforce the boundary conditions exactly. In our case, we take the following:

$$V^e(t,a,y) = f^e(t,a,y) - af_a^e(t,0,y) + a(\alpha \exp(-\gamma y) + \exp(f_a^e(t,0,y))), \tag{12}$$

$$V^u(t,a) = f^u(t,a) - af_a^u(t,0) + a(\alpha \exp(-\gamma y_f) + \exp(f_a^u(t,0))). \tag{13}$$

One can easily verify that the asset boundary conditions are satisfied, and

$$V_t^e = f_t^e - af_{ta}^e(t,0,y) + a\exp(f_a^e(t,0,y))f_{ta}^e(t,0,y), \tag{14}$$

$$V_t^u = f_t^u - af_{ta}^u(t,0) + a\exp(f_a^u(t,0))f_{ta}^u(t,0). \tag{15}$$

To deal with the second order derivatives, we use the FO-PINN method proposed by Gladstone, Nabian, Sukumar, Srivastava, and Meidani (2023).

2.2. Quasi-Finite Difference Physics-Informed Neural Network

In this subsection, we introduce a new approach for solving the PDE with terminal conditions, the Quasi-Finite Difference Physics-Informed Neural Network (QFD-PINN). The key

[1]Which loss to use is somewhat debatable. In general the L^2 loss is preferrable, while it is possible to use the L^1 loss shrinking some parts in the loss function, for example, the first order condition parts.

idea is to combine the finite difference method with the neural network to conquer the curse of dimensionality.

Given a small, prespecified length of time, dt, we rewrite our PDE into the following semi-discretized form. To avoid the issue of negative derivatives, we replace V_a^e and V_a^u with the consumption policy and the first order conditions. The consumption policies are transformed to ensure that they are always positive.

$$\begin{aligned} V^e(t+dt) &= V^e + dt\left[(\rho + h_t + \lambda_y + \lambda^e)V^e - \frac{\alpha_e}{1-\gamma}(c^e)^{1-\gamma} - h_t b(a) - \lambda_y \int V^e(t,a,y\exp(\mu_t + \sigma e))d\Phi(e)\right] \\ &\quad - dt\left[\lambda^e \max\{V^u, V^e\} + \alpha_e (c^e)^{-\gamma}(ra + y - c^e)\right], \qquad (16) \\ V^u(t+dt) &= V^u + dt\left[(\rho + h_t + \lambda^u)V^u - \frac{1}{1-\gamma}(c^u)^{1-\gamma} - h_t b(a) - \lambda^u \int \max\{V^e(t,a,y), V^u(t,a)\}\, dF(y)\right] \\ &\quad - dt\left[(c^u)^{-\gamma}(ra + y_f - c^u)\right], \qquad (17) \\ V_a^e &= \alpha_e (c^e)^{-\gamma}, \qquad (18) \\ V_a^u &= (c^u)^{-\gamma}. \qquad (19) \end{aligned}$$

We use the auto-differentiation to compute the derivative in the state space.

Alternatively, we can use the explicit scheme as well, which gives

$$\begin{aligned} V^e(t) &= V^e - dt\left[(\rho + h_t + \lambda_y + \lambda^e)V^e - \frac{\alpha_e}{1-\gamma}(c^e)^{1-\gamma} - h_t b(a) - \lambda_y \int V^e(t+dt,a,y\exp(\mu_t + \sigma e))d\Phi(e)\right] \\ &\quad + dt\left[\lambda^e \max\{V^u, V^e\} + \alpha_e (c^e)^{-\gamma}(ra + y - c^e)\right], \qquad (20) \\ V^u(t) &= V^u - dt\left[(\rho + h_t + \lambda^u)V^u - \frac{1}{1-\gamma}(c^u)^{1-\gamma} - h_t b(a) - \lambda^u \int \max\{V^e(t+dt,a,y), V^u(t+dt,a)\}\, dF(y)\right] \\ &\quad + dt\left[(c^u)^{-\gamma}(ra + y_f - c^u)\right], \qquad (21) \\ V_a^e(t) &= \alpha_e (c^e(t))^{-\gamma}, \qquad (22) \\ V_a^u(t) &= (c^u(t))^{-\gamma}, \qquad (23) \end{aligned}$$

where the right hand sides are all evaluated at time $t + dt$.

3. Methods to Try

1. DeepONet for the inversion problem in the implicit scheme iteration.
2. Hard-constraint QFD-PINN with quasi-implicit scheme to speed up.

References

Gladstone, R. J., Nabian, M. A., Sukumar, N., Srivastava, A., & Meidani, H. (2023). *Fo-pinns: A first-order formulation for physics informed neural networks.*