Policy Tree Descendant Algorithm for Solving Consumption-Savings Problems with Safety Net

PRELIMINARY AND INCOMPETE DRAFT

Fedor Iskhakov[†] and Wending Liu[‡]

June, 2025

Abstract

This paper develops a new algorithm for solving consumption-savings lifecycle dynamic problems that feature social insurance in the form of an exogenous consumption floor. Despite the proliferation of this setup in the literature, the fact such safety net programs render the dynamic optimization problem non-convex is generally overlooked. Unlike traditional methods, our approach delivers a fast and accurate solution by tracking the resulting discontinuities in the policy function systematically. Based on Monte Carlo integration, our method is particularly suited for the problems with larger state spaces allowing to account for more intricate stochastic processes of income and expenditure. In a series of simulation exercises, we demonstrate the advantages of our proposed solution method in comparison to the endogenous grid method.

Keywords: dynamic programming, life-cycle model, safety net, consumption floor, policy tree descendant algorithm

JEL codes: C61, C63, D15, H55

[†]Australian National University `Fedor.Iskhakov@anu.edu.au`
[‡]Australian National University `Wending.Liu@anu.edu.au`

1 Introduction

A life-cycle model with intertemporal consumption-saving decisions is a workhorse model in many areas of macroeconomics and structural labor/public economics. It is widely used in both single-agent models to study the saving/dis-saving behavior of households, labor supply and retirement decisions, household decision-making under unitary or collective preferences, etc. As a model of consumer decision-making, it is featured in classic general equilibrium macroeconomic models, models with heterogeneous agents, and overlapping generations models. In many of the application areas, the model is extended with a social insurance program that directly affects the consumption-saving decisions of the agents on the lower side of the wealth distribution, and indirectly affects the decision-making of all agents through lifecycle dynamics and general equilibrium effects. The simplest way to model social insurance is to introduce a simple consumption floor, that is to assume that the government makes a means-tested transfer to the agent if their wealth falls below a certain level. Setting the taper rate of this transfer to fully offset any existing wealth, makes for a particularly simple model of social insurance where the wealth level of the next period is simple a maximum of the endogenous savings and the fixed floor.

However, this simple approach to including social insurance program into a dynamic choice model has important implications. In its presence an agent with a low level of wealth, when considering a particular consumption-savings combination, may find themselves better off by either reducing their consumption to save more or by consuming all their wealth and relying on the safety net in the next period. The optimization problem like this has multiple stationary points, and thus the first-order conditions are not sufficient to characterize the optimal policy. This makes the problem non-convex and requires the standard methods of solving dynamic programming problems to be applied with caution.

Non-convex dynamic problems came into the spotlight in the literature in the context of applying endogenous grid method (EGM) (Carroll, 2006) based solely on the first-order conditions to a wider range of problems. Initially Fella (2014) considered a consumption-saving model in the presence of switching costs, and Iskhakov et al. (2017) developed a model with additional discrete choices. In both of these setups the value function, though continuous and increasing, exhibits "downward kinks", in the terminology of Clausen and Strub (2020), where the decision maker is indifferent between discrete choices or incurs a switching cost. The value function becoming non-convex function of wealth in the neighborhood of the kink points. The optimal consumption which is directly linked to the marginal value mirrors these kinks as discontinuities of the policy function where the consumption levels on the two sides of the jump reflect the

"regime switch" at the kink point. Similarly, in the presence of safety net social insurance, the kinks are due to switching between relying on or not relying on the consumption floor in the next period.

One approach to dealing with non-convex problems is to apply a robust optimization algorithm such as a simple grid search over a finitely discretized consumption choice, as part of value function iterations or backward induction. Though inaccurate and slow (the optimization problem has to be solved in each grid point in the state space), this approach is guaranteed to find the global optimizer even when the objective function in the Bellman equation is not concave. The drawback of this approach is the inevitable loss of accuracy due to the discretization of the continuous choice variable. The generalizations of the EGM method proposed by Iskhakov et al. (2017); Druedahl and Jørgensen (2017); Druedahl (2021); Dobrescu and Shanker (2022) offer a much more accurate and faster approach which relies on computing the *value function correspondence* as a nexus of points where the first-order conditions are satisfied, and then recovering the value function itself as an upper envelope of this correspondence.

We propose an alternative solution approach that also targets non-convex dynamic optimization problems, but avoids discretization of not only the choice but also the continuous state variable. Instead, our approach recognizes the recursive nested structure in the space of so-called *policy segments* which are the continuous pieces of the overall discontinuous optimal consumption policy. As discussed in Iskhakov et al. (2017), in a deterministic non-convex dynamic model, the kinks in the value function might multiply and accumulate as the solver progresses backward in time. Correspondingly, the number of policy function segments might also be different for different periods. We show that viewed from the terminal period T, the optimal policy can be represented by a *tree* graph where each policy function segment in period t has a well-defined ancestor segment in period $t+1$ and a descendant segment in period $t-1$.

Consequently, we refer to our solution approach as the *policy tree descendant* (PTD) algorithm. Effectively, it traverses the policy tree from the terminal period T to the initial period in a finite horizon model, or convergence in an infinite horizon model, and builds the policy function segments in each period t based on the segments in the next period $t+1$.

Because our approach does not rely on the discretization of the continuous state variable, namely, neither beginning-of-the-period nor post-decision wealth, the method completely avoids the need for interpolation, and importantly extrapolation, which is replaced by computing the discounted value by direct summation of utility along each drawn path. Thus, our approach avoids all interpolation and extrapolation errors, trading them for the standard error of Monte Carlo integration. We show that in practice, this trade-off is beneficial.

Importantly, the described policy tree exploration procedure can be implemented only for a given realization of exogenous shocks, such as income shocks we consider in the paper. We rely on Monte Carlo integration to compute the expectation of the future value in stochastic models after the policy tree is built for each realization of the stochastic processes in the model. Given the simplicity of the solution of a deterministic model along each drawn realization of the stochastic process, we find our procedure has significantly better performance the EGM-based methods in terms of accuracy, robustness and speed.

A very important benefit of using Monte Carlo integration used in PTD algorithm over Gaussian quadrature used in EGM algorithm in the non-convex stochastic dynamic models is the fact that the former does not suffer from the numerical instability of the latter when integrating non-smooth functions. In practice, applying the off-the-shelf quadrature methods in the non-convex dynamic models results in additional spurious kinds and discontinuities which are impossible to distinguish from the genuine ones. Naturally, our approach is particularly well suited for problems with large state spaces, where Monte Carlo integration is generally preferred to the quadrature methods. An example of such a problem is a consumption-savings model where income is informed by a multidimensional exogenous dynamic stochastic process, such as productivity and medical expenditure in Hubbard et al. (1995).

The Monte Carlo integration error is the only source of inaccuracy in applying the PTD solver to stochastic problems, but it can be controlled by choosing a larger number of draws. Given that our method can be applied to the drawn stochastic paths independently, our method is amenable to parallel computation, which is a good way to increase accuracy as well as decreasing the run time.

The paper is organized as follows. In Section 2, we describe the life-cycle model with social insurance. In Section 3, we show the PTD solver the deterministic model as well as its numerical performance compared with EGM algorithm. In Section 4, we extend the PTD solver to the stochastic model and examine its numerical performance compared with EGM algorithm. In Section 5, we describe the class of life-cycle models that can be solved by the PTD algorithm. Section 6 concludes the paper.

2 An Illustrative Problem: Consumption with Social Insurance

We consider the following consumption-saving model with social insurance. And agent maximizes her lifetime utility. The time horizon is discretized as $t \in \{1, 2, \ldots, T\}$. We denote x_t as the cash-on-hand, c_t as the consumption, y_t as the net income, and a_t as the end-of-period

asset at period t. The net interest rate is a constant r. The consumption floor $\underline{x}$ is the minimal level of cash on hand secured by the government.[1] The cash-on-hand level without government transfer at t is denoted as z_t. The utility function $u : \mathbb{R}_+ \times \mathbb{R}_{++} \to \overline{\mathbb{R}}$ takes the CRRA form widely used in life-cycle literature:

$$u(c;\gamma) = \begin{cases} \frac{c^{1-\gamma}-1}{1-\gamma} & \text{if } \gamma \in \mathbb{R}_{++} \setminus \{1\}, \\ \log(c) & \text{if } \gamma = 1. \end{cases} \tag{1}$$

We will omit the relative risk aversion coefficient γ in the notation if no confusion arises. Moreover, we set $u(0) = \lim_{x \to 0^+} u(x)$ for the remaining paper, i.e., we set $u(0) = -\infty$ if $u(c) = \log(c)$.

Moreover, we define the basic uncertainties in the income process as the IID process $\{\boldsymbol{\varepsilon}_t\}_{t=2}^T$, where the identity mapping $\boldsymbol{\varepsilon}_t$ is on the probability space $(\mathbb{R}^n, \mathcal{B}(\mathbb{R}^n), \mu)$ for all t. We also denote all the exogenous states at t as $\boldsymbol{s_t} \in \mathbb{R}^k$.

For $t \in \{1, ..., T-1\}$, the Bellman equation is:

$$\begin{aligned} \omega_t(x_t, \boldsymbol{s}_t) &= \max_{c_t} \Big\{ u(c_t) + \beta \int \omega_{t+1}\Big(\max\left(z_{t+1}(\boldsymbol{\varepsilon}), \underline{x}\right), \boldsymbol{s}_{t+1} \Big) \mu(d\boldsymbol{\varepsilon}) \Big\}, \\ \text{s.t.} \quad a_t &= x_t - c_t, \\ z_{t+1} &= (1+r)a_t + y_{t+1}, \\ y_{t+1} &= f(\boldsymbol{s}_{t+1}), \\ \boldsymbol{s}_{t+1} &= g(\boldsymbol{s}_t, \boldsymbol{\varepsilon}), \\ 0 &\le c_t \le x_t, \\ \omega_T(x_T, \boldsymbol{s}_T) &= u(x_T) \quad \text{for all } (x_T, \boldsymbol{s}_T) \in \mathbb{R}_+ \times \mathbb{R}^k, \end{aligned} \tag{2}$$

where ω is the value function, $\boldsymbol{\varepsilon}$ is any realization of $\boldsymbol{\varepsilon}_{t+1}$, $f : \mathbb{R}^k \to \mathbb{R}$, $g : \mathbb{R}^{k+n} \to \mathbb{R}^k$.

We consider three special examples of (2) in this paper:

Example 1 (IID)**.** *$n = k = 1$, ε_t has lognormal distribution, $g(s_t, \varepsilon) = \varepsilon$, $f(s_{t+1}) = s_{t+1}$. This is the discrete-continuous dynamic choice models with IID income shocks discussed in Iskhakov et al. (2017). The exogenous states $\boldsymbol{s}_t$ can be removed from the value function. We denote this example by IID for brevity.*

Example 2 (AR1)**.** *$n = k = 1$, ε_t has normal distribution, $g(s_t, \varepsilon) = \rho s_t + \varepsilon$ $(0 < \rho < 1)$, $f(s_{t+1}) = \exp(s_{t+1})$. This is the work-horse consumption-saving model with AR(1) log income shocks discussed in heterogenous agent model literature with social insurance feature as suggested by Krusell and Smith (1998). We denote this example by AR1.*

[1] We use both "consumption floor" and "social insurance" to refer to $\underline{x}$ in this paper.

Example 3 (HSZ)**.** $n = k = 2$, $\boldsymbol{s}_t = [e_t, m_t]'$, $\boldsymbol{\varepsilon}_t = [\varepsilon_t^e, \varepsilon_t^m]'$ *has 2D normal distribution with correlation parameter zero,* $g_1(\boldsymbol{s}_t, \boldsymbol{\varepsilon}) = \rho_e e_t + \varepsilon_t^e$, $g_2(\boldsymbol{s}_t, \boldsymbol{\varepsilon}) = \rho_m m_t + \varepsilon_t^m$ $(0 < \rho_e < 1$ *and* $0 < \rho_m < 1)$, $f(\boldsymbol{s}_t) = \exp(e_t) - \exp(m_t)$. *We interpret* e_t *as the log earning at* t, m_t *as the log medical expenditure at* t. *The net income* y_t *is the difference between the earning and the medical expenditure, thus, it can be negative. This is the model discussed in Hubbard et al. (1995), and it has become a standard feature in many labor economics models, for example, Scholz et al. (2006), De Nardi et al. (2009), De Nardi et al. (2010), Li et al. (2016),Fan et al. (2024). We denote the example by HSZ (the initial letters the three authors' surnames of Hubbard et al. (1995)).*

Our aim is to build a simulator for each example. The simulator should be able to generate the consumption-saving paths for a given group of agents and the realization of the time series of the basic uncertainties $\{\boldsymbol{\varepsilon}_t\}_{t=1}^T$ for each agent.

3 Policy-Tree Descendant Algorithm for Deterministic Models

3.1 DP problem under deterministic income sequence

Solving the DP problem (2) is challenging (especially for Example 3 due to the 3D state space), thus, we consider the deterministic counterpart of the stochastic model first. We will see that the solver for the deterministic model can be used as a key building block for the simulator of stochastic model.

To construct the deterministic counterpart of the original stochastic DP problem. We assume $\{y_t\}_{t=1}^T$ is already known to the agent $t = 1$, i.e., all the uncertainty in the original model is removed. For $t \in \{1, \ldots, T-1\}$, the Bellman equation for the new deterministic DP problem is:

$$\begin{aligned} v_t(x_t) &= \max_{c_t} \Big\{ u(c_t) + \beta v_{t+1}\big(\max(z_{t+1}, \underline{x})\big) \Big\}, \\ \text{s.t.} \quad a_t &= x_t - c_t, \\ z_{t+1} &= (1+r)a_t + y_{t+1}, \\ 0 &\le c_t \le x_t, \\ v_T(x_T) &= u(x_T) \quad \text{for all } x_T \ge 0. \end{aligned} \tag{3}$$

3.2 Main Ideas of the Solution Method

Before introducing the algorithms to solve (3) formally, we first describe the main ideas of our solution methods by analyzing the last three periods (T, $T-1$ and $T-2$) problems. We use

$T = 50,\ \beta = 0.98,\ \gamma = 1,\ r = 0.1,\ \underline{x} = 3,\ y_t = 1$ for all $t \in \{1, \ldots, T\}$ as the parameters setting for the following numerical examples of the deterministic model.

For the terminal period T, it's trivial that the agent consumes all her cash-on-hand, i.e., her policy function $c_T(x) \equiv c_T^0(x_T) = x_T$ for all $x_T \in \mathbb{R}_+$. We use the superscript "0" to denote the policy of consuming all the wealth in the remaining text. The policy function and value function are shown in Figure 1.

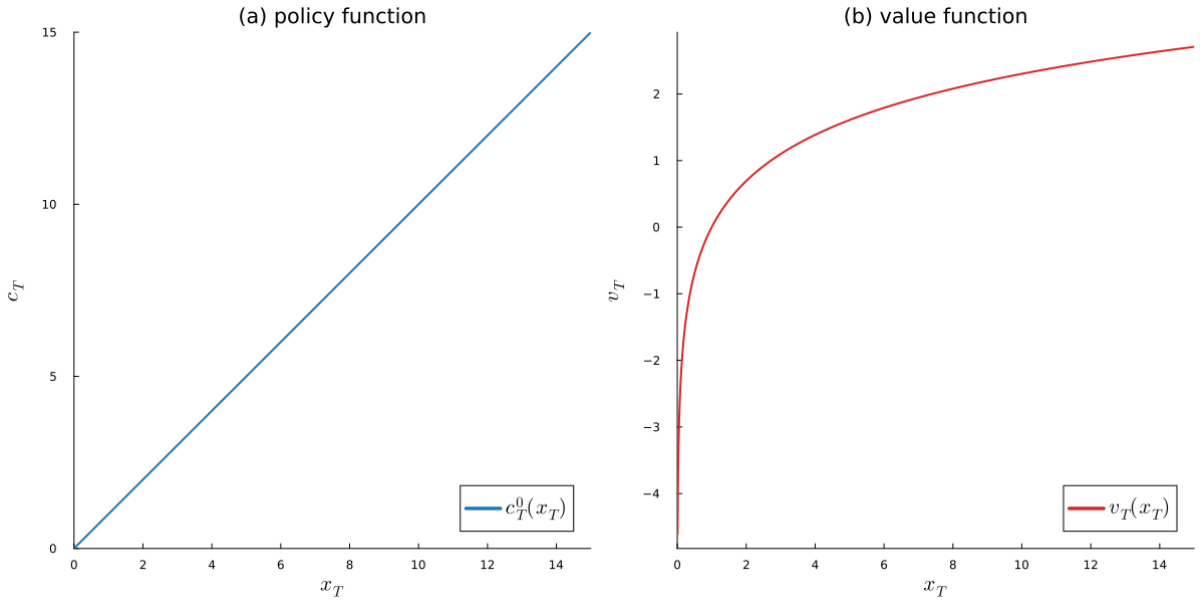

Figure 1: Policy and value functions at T

Notes: $u(c) = log(c)$.

Now we study the agent's DP problem at $t = T-1$. At $T-1$, the agent can still choose the policy of consuming all, which is $c_{T-1}^0(x_{T-1}) = x_{T-1}$, and rely on the social insurance $\underline{x}$ next period. However, the optimal consumption function at $T-1$ is not c_{T-1}^0 since the agent will have a positive saving if her wealth level is sufficiently large, in which case she will rely on her saving at T rather than the consumption floor $\underline{x}$ provided by the government.

If the agent's optimal policy is saving positively, the Euler equation holds:

$$u'\big(c_{T-1}^1(x_{T-1})\big) = \beta(1+r)u'\big(c_T^0(z_T)\big) \tag{4}$$

We define the consumption policy function satisfying (4) by $c_{T-1}^1 \equiv Mc_T^0$, where M is an "descendant" operator[2] that maps the consumption policy c_T^0 at T to the consumption policy c_{T-1}^1 at $T-1$. Since c_{T-1}^1 is derived purely from the Euler equation and pinned down by c_T^0, we call c_{T-1}^1 the "child policy" of the "parent policy" c_T^0. Notice that the domain of c_{T-1}^1 is only a

[2]We will give the formal definitions of M and the following F, G operators as well as describe the algorithms implementing them in Section 3.3.

proper subset of $\mathbb{R}_+$ due to the borrowing constraint, and c^1_{T-1} is not the optimal consumption policy function at $T-1$, thus, we call c^0_{T-1} and c^1_{T-1} two "policy segments".

Now we can consider how to find the optimal policy $c^\star_{T-1}$ at $T-1$. Since the agent's optimal policy at any wealth level x at $T-1$ is either $c^0_{T-1}(x)$ or $c^1_{T-1}(x)$, she only needs to compare the two "value segments" derived from the two "policy segments". The two value segments are defined by $v^0_{T-1}(x) = u\big(c^0_{T-1}(x)\big) + \beta v_T(\underline{x})$, $v^1_{T-1}(x) = u\big(c^1_{T-1}(x)\big) + \beta v_T\Big(z_T\big(c^1_{T-1}(x)\big)\Big)$. We also construct a "fusion" operator $F(c^0_{T-1}, c^1_{T-1}; v_T) = (c^{\star 0}_{T-1}, c^{\star 1}_{T-1})$ by updating the domains of c^0_{T-1} and c^1_{T-1} such that the collection of the new domains is a partition of $\mathbb{R}_+$. Moreover, if $x \in \text{dom}(c^{\star i}_{T-1})$, then $x \notin \text{dom}\left(c^j_{T-1}(x)\right)$ or $v^i_{T-1}(x) \geq v^j_{T-1}(x)$, where $i \in \{0,1\}$, $j \in \{0,1\}\setminus\{i\}$.

Thus, the optimal policy $c^\star_{T-1}$ is the collection of the two fused policy segments $c^{\star 0}_{T-1}$ and $c^{\star 1}_{T-1}$.[3] We visualize the two operators in Figure 2.

In Figure 2 (a), the orange line is the policy segment c^0_{T-1} of consuming all at $T-1$, whose domain is $\mathbb{R}_+$. The blue line is the child policy $c^1_{T-1} = Mc^0_T$, where c^0_T is the blue line in Figure 1. The domain of c^1_{T-1} is from the x-coordinate of the purple dot to positive infinity. The purple dot represents the wealth level on which the consumption policy satisfies the Euler equation and the end-of-period asset level is zero.

In Figure 2 (b), the orange line represents the candidate value function v^0_{T-1} of consuming all at $T-1$. The blue line represents the candidate value function v^1_{T-1} of positive saving at $T-1$. To determine the consumption policy at x, we just need to check whether the corresponding y-coordinate of the blue line or the orange line is higher. Notice that there is only one intersection of the blue line and the orange line. The agent will use c^1_{T-1} if and only if her wealth level is beyond the intersection point. This observation implies that to implement the fusion operator F, we don't need to calculate the upper envelope of the value function by interpolating or scanning the value correspondence as in the EGM algorithm literature. We only need to calculate the accurate position of the intersection point.

Figure 2 (c) shows the result of the fusion operator F, which is a piece-wise affine policy function. Figure 2 (d) shows the value function v_{T-1}, which is differentiable everywhere but the kink point. Following Iskhakov et al. (2017) and Clausen and Strub (2020), we use "kink" to refer to a continuous but non-differentiable point of a function. The x-coordinate of the kink point is the same as that of the intersection point in Figure 2 (b).

Figure 2 also shows an important feature of the consumption-saving model with social insurance: the wealth state with a positive end-of-period asset derived from the Euler equation might be strictly dominated by the corner solution, see Figure 2 (b). This occasionally binding

[3] A formal way is to set $c^\star_{T-1}(x) = \mathbb{1}\big(x \in \text{dom}(c^{\star 0}_{T-1})\big)c^{\star 0}_{T-1}(x) + \mathbb{1}\big(x \in \text{dom}(c^{\star 1}_{T-1})\big)c^{\star 1}_{T-1}(x)$.

constraint problem has been discussed in the EGM literature (Druedahl and Jørgensen, 2017; Druedahl, 2021). And the PTD algorithm handled this issue naturally without any special treatment.

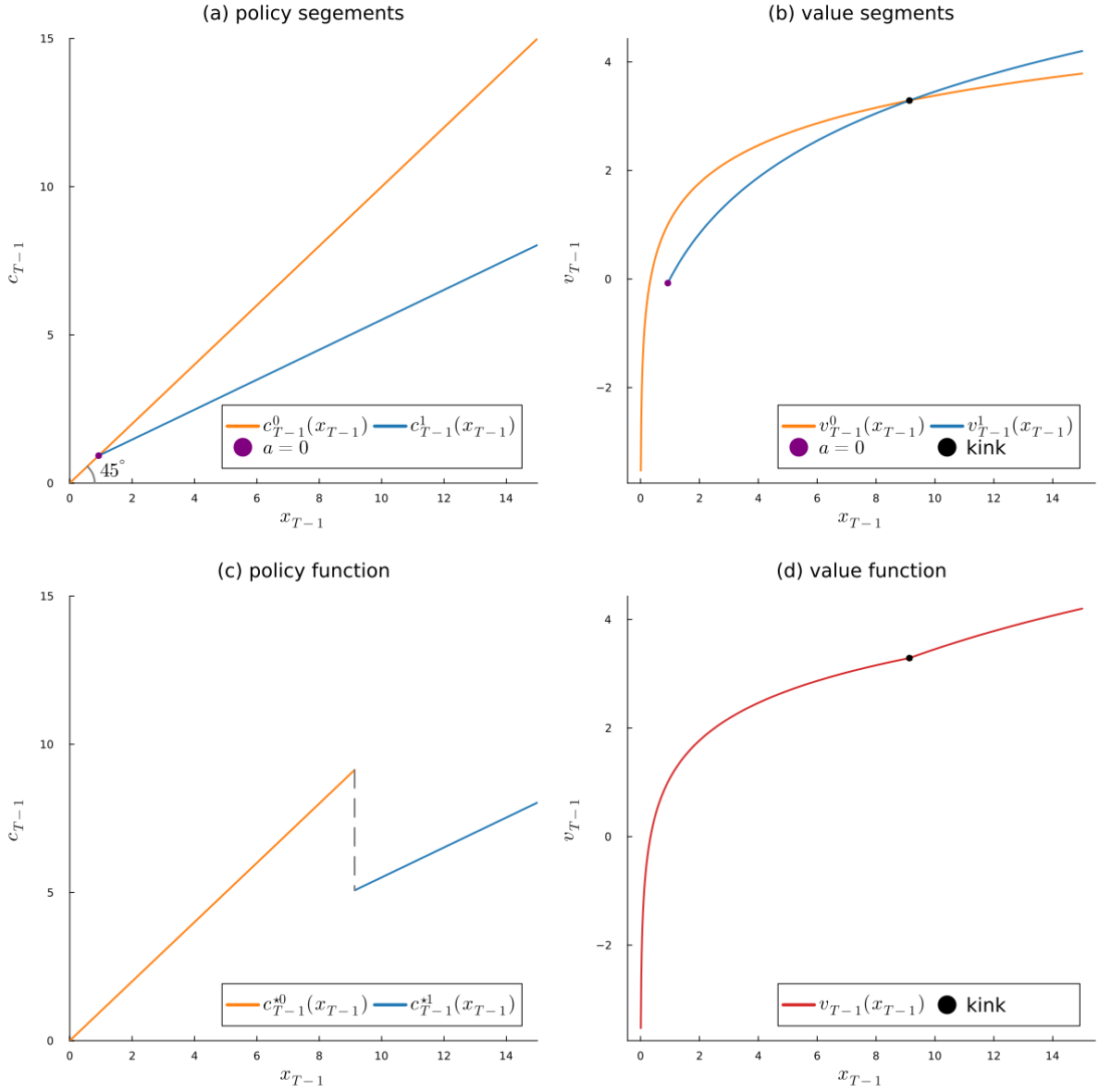

Figure 2: Policy and value functions at $T-1$

Notes: $u(c) = log(c)$, $r = 0.1$, $y_T = 1.0$, $\underline{x} = 3.0$.

Now we consider the agent's problem at $T-2$. The procedure to solve the policy function is shown in Figure 3.

First, we use the descendant operator M to find the child policy segment c^1_{T-2} (orange line in Figure 3 (a)) of the parent $c^{\star 0}_{T-1}$ (orange line in Figure 2 (c)) and the other child policy

segment c^2_{T-1} (blue line in Figure 3 (a)) of the parent $c^{\star 1}_{T-1}$(blue line in Figure 2 (c))[4], which are shown in Figure 3 (a), where we also draw the policy segment of consuming all, c^0_{T-2} (green line in Figure 2 (a)). We draw the value segments derived from the three policy segments in Figure 3 (b).

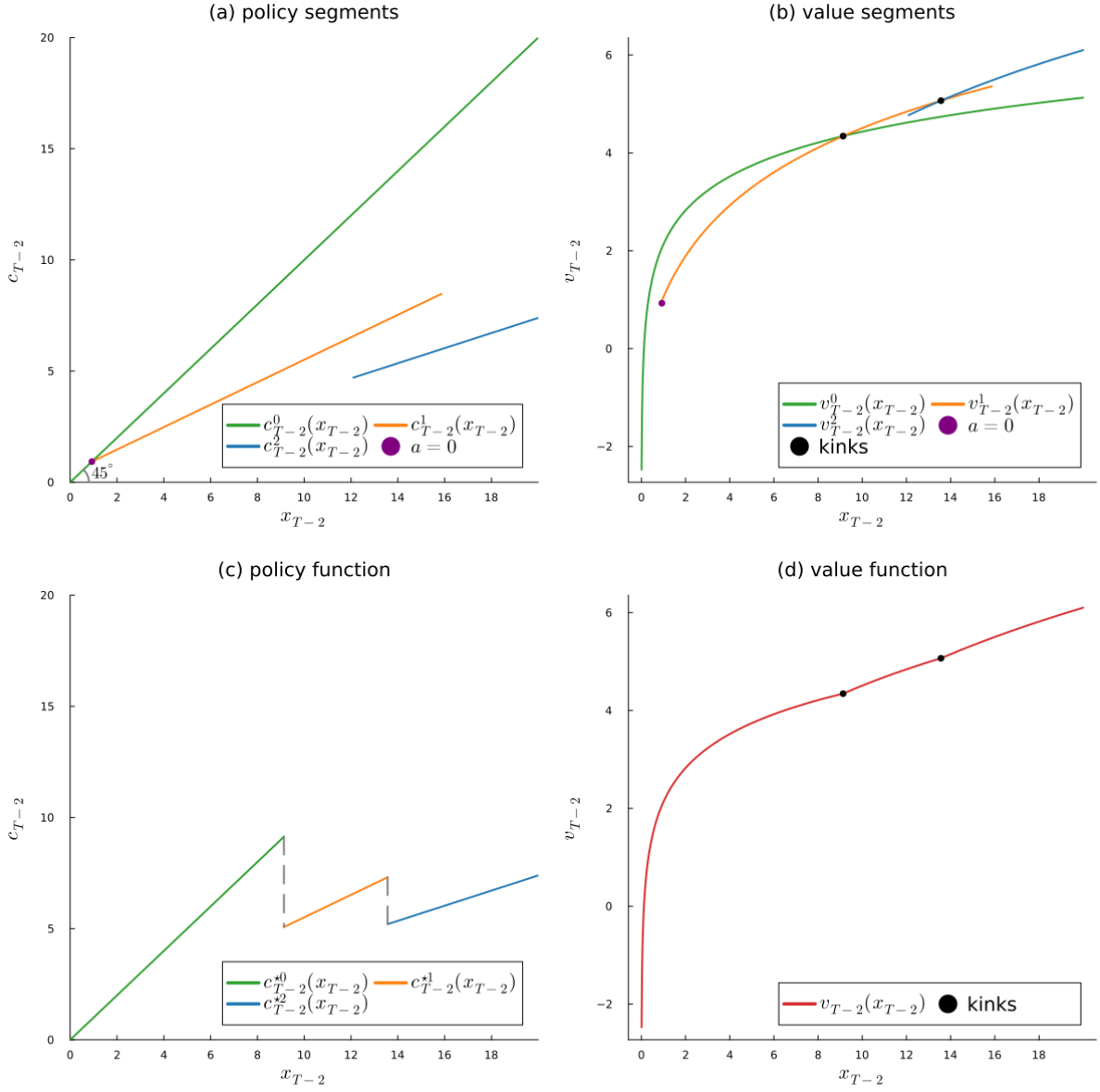

Figure 3: Policy and value functions at $T-2$

Notes: $u(c) = log(c)$, $r = 0.1$, $y_{T-1} = y_T = 1.0$, $\underline{x} = 3.0$.

Notice that there are three candidate policy functions now, we define a "glue" operator G by updating the domains of the policy segments such that the collection of new domains is a partition of $\mathbb{R}_+$: $G(c^0_{T-2}, c^1_{T-2}, c^2_{T-2}; v_{T-1}) = (c^{\star 0}_{T-1}, c^{\star 1}_{T-1}, c^{\star 2}_{T-1})$. Moreover, given any state x,

[4]Another strategy is to derive the children of the parents c^0_{T-1} and c^1_{T-1}, but this will make the computation more complicated since $c^{\star 0}_{T-1}$ and $c^{\star 1}_{T-1}$ are "fused" version of c^0_{T-1} and c^1_{T-1} respectively, which have smaller domains.

$c^{\star}_{T-2}(x) = c^{\star i}_{T-2}(x)$ for some $i \in \{0, 1, 2\}$. The glue operator G takes an array of policy segments as arguments, and it is a natural extension of the fusion operator F, which only accepts two policy segments as arguments. The implementation of G will be described in Section 3.3. The policy function obtained from applying the glue operator is shown in Figure 3 (c), and the value function derived from the policy function is shown in Figure 3 (d).

As the above analysis of the last three periods' problem shows, our solution method is based on the parent-child policy segment structure. The fusion operator F and glue operator G help us generate child policies from parent policies. We can visualize the structure as a policy tree in Figure 4.

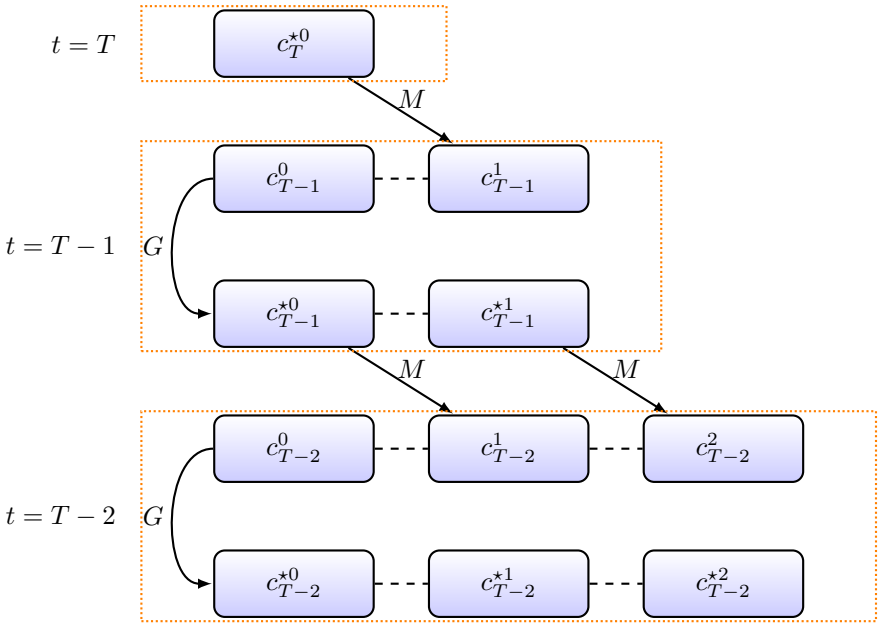

Figure 4: Policy tree of the last three periods

Notes: $u(c) = log(c)$, $r = 0.1$, $y_{T-2} = y_{T-1} = y_T = 1.0$, $\underline{x} = 3.0$. For other parameter settings, there might be fewer leaves in the tree.

The policy tree in Figure 4 grows from the root $c^{\star 0}_{T}$ at the top, which has the child policy segment c^{1}_{T-1} at the second layer, and c^{0}_{T-1} is the policy segment of consuming all at $T-1$. Since c^{0}_{T-1} is not dominated by c^{1}_{T-1}, the second layer still has two leaves $c^{\star 0}_{T-1}$ and $c^{\star 1}_{T-1}$ after we use the operator G to glue them. Now $c^{\star 0}_{T-1}$ generates child policy segment c^{1}_{T-2}, and $c^{\star 1}_{T-1}$ generates child policy segment c^{2}_{T-2}. After gluing the two policies and the policy of consuming all the wealth, we get three leaves at the third layer of the policy tree. The procedure of computing the last three periods' policies can be carried to the remaining periods, i.e., we can continue drawing the policy tree until it reaches the bottom, which is the optimal policy at $t = 1$. In the next subsection, we formally introduce the algorithms that will be used to solve the deterministic DP problem (3).

3.3 PTD algorithm for the deterministic DP problem

Now we give a formal introduction of our algorithm for solving the DP problem defined in (3).

Definition 1. The *policy segment* c_t^k at period t is a continuous function $c_t^k : D \to \mathbb{R}_+$, where D is a closed interval of $\mathbb{R}_+$.[5]

Definition 2. The *refined policy segment* $c_t^{\star k}$ of policy segment c_t^k is a policy segment such that

(i) $\operatorname{dom}(c_t^{\star k}) \subset \operatorname{dom}(c_t^k)$,

(ii) $c_t^{\star k}(x) = c_t^k(x)$ for all $x \in \operatorname{dom}(c_t^{\star k})$,

(iii) $c_t^k(x)$ is the optimal policy of the agent at period t if and only if $x \in \operatorname{dom}(c_t^{\star k})$, and $c_t^k(x)$ is the unique optimal policy if $x \in \mathbf{int}\operatorname{dom}(c_t^{\star k})$.

Remark. By the definitions of policy segment and refined policy segment, there is one or zero intersection point between the domains of two different refined policy segments.

Definition 3. The *policy collection* $\mathbf{c_t}$ at period t is a finite collection of nonempty refined policy segments $\{c_t^{\star 0}, c_t^{\star 1}, \ldots, c_t^{\star n}\}$ such that

(i) $\mathbf{sup}\operatorname{dom}(c_t^{\star i}) = \mathbf{inf}\operatorname{dom}(c_t^{\star i+1})$ for all $i \in \{0, 1, \ldots, n-1\}$,

(ii) for any $x \in \mathbb{R}_+$, there exists $k \in \{0, 1, \ldots, n\}$ such that the optimal policy at x, $c_t^\star(x) = c_t^{\star k}(x)$.

Remark. Policy collection is the data structure of the policy function. It immediately gives us the simple algorithms of evaluate the policy function and value functions (Algorithm 1 and Algorithm 2). Notice that we don't "save" the value function on grids, thus, no interpolation or extrapolation is needed, and this is an important feature of our algorithm distinguished from the existing methods solving non-convex DP problems.

Algorithm 1: Evaluate policy function $c_t^\star$

Input : policy collection $\mathbf{c}_t$, wealth level x;

Output: optimal policy $c_t^\star(x)$;

```
for c_t^{*k} ∈ c_t do
    if x ∈ dom(c_t^{*k}) then
        return c_t^{*k}(x);
    end
end
```

[5] The empty set and unbounded intervals are closed interval of $\mathbb{R}_+$.

Algorithm 2: Evaluate value function v_t

Input : policy collections $\{\mathbf{c}_j\}_{j=t}^{T}$, wealth level x, parameters in DP problem (3);

Output: value $v_t(x)$;

$v \leftarrow 0$; // initialize $v_t(x)$

for $j = t : T$ **do**

  $c \leftarrow c_j^\star(x)$; // using Algorithm 1 to get optimal policy

  $v \leftarrow v + \beta^{j-t} u(c)$;

  $x \leftarrow \max\{(1+r)(x-c) + y_{j+1},\ \underline{x}\}$; // next period wealth level

end

return v;

Notice that in algorithm Algorithm 2, we can save $v_t(x)$ in a preallocated memory if $x = \underline{x}$ to avoid repeatedly calculating value of the consumption floor in the following algorithms.

Definition 4. The *child policy segment* c_{t-1}^{k+1} of a policy segment c_t^k is the policy segment such that let $z_t^k(x) := (1+r)\big(x - c_{t-1}^{k+1}(x)\big) + y_t$, we have:

$$z_t^k(x) \in \mathrm{dom}(c_t^k), \tag{5}$$

$$u'\Big(c_{t-1}^{k+1}(x)\Big) = \beta(1+r)u'\Big(c_t^k\big(z_t^k(x)\big)\Big), \tag{6}$$

where $\mathrm{dom}(c_{t-1}^{k+1})$ is the set of all the x such that (5) and (6) hold. We call c_t^k the *parent policy parent* of c_{t-1}^{k+1}.

Remark. The requirement (5) ensures that the agent's next period wealth without social insurance is in the domain of the parent policy c_t^k. The requirement (6) is the Euler equation.

Now we discuss how to derive the child policy segment given a parent policy segment.

Notations. We write $\mathcal{S}_t$ for the space of all possible policy segments at t, $\mathcal{V}_t$ for the space of all possible value functions at t for all the possible parameters setting in (3), and $\mathfrak{S}_t$ for the space of finite collections of policy segments at t.

Definition 5. For any $t \in \{2, \dots, T\}$, the descendant operator M is a mapping of $\mathcal{S}_t$ into $\mathcal{S}_{t-1}$ defined by $M(c_t^k) = c_{t-1}^{k+1}$, where c_{t-1}^{k+1} is the child policy segment of c_t^k.[6]

To implement the descendant operator M, we first show a very useful property of M.

Theorem 1. *For any $t \in \{2, \dots, T\}$, if the parent policy segment $c_t^k(x)$ is an affine function of state variable x on its domain, then the child policy segment $M(c_t^k)(x)$ is also an affine function of x on its domain.*

[6] We don't add a subscript t to M and the following F, G operators since the definitions are the same for all t.

Proof. See Appendix A. □

Remark. Since $c_T^\star(x) = x$ is a linear function, Theorem 1 implies that all the child policy segment descendant from the "root" $c_T^\star$ are affine functions. Notice that if we add a bequest term to the final period utility function as in De Nardi et al. (2010), we can still apply Theorem 1.

Notations. We use a tuple $c_t^k := (x_1,\ x_2,\ b_0,\ b_1)$ to represent an affine policy segment, where $\mathrm{dom}(c_t^k) = [x_1,\ x_2]$, $c_t^k(x) = b_0 + b_1 x$ for all $x \in \mathrm{dom}(c_t^k)$.

Since the child policy segment is an affine function, we only need to pin down the two endpoints of its domain and the consumption levels at the two endpoints. This can be efficiently done by using the inverse euler equation as shown in Algorithm 3.

Algorithm 3: Descendant operator M

Input : parent policy segment $c_t^k \equiv (x_1,\ x_2,\ b_0,\ b_1)$, parameters in DP problem (3);

Output: child policy segment c_{t-1}^{k+1} of c_t^k;

$a_3 \leftarrow \frac{x_1 - y_t}{1+r}$, $a_4 \leftarrow \frac{x_2 - y_t}{1+r}$; `// end-of-period assets at endpoints`

$c_3 \leftarrow u'^{-1}\Big(\beta(1+r)u'\big(c_t^k(x_1)\big)\Big)$, $c_4 \leftarrow u'^{-1}\Big(\beta(1+r)u'\big(c_t^k(x_2)\big)\Big)$;

$x_3 \leftarrow a_3 + c_3$, $x_4 \leftarrow a_4 + c_4$;

$d_0 = \frac{c_4 - c_3}{x_4 - x_3}$, $d_1 = c_3 - b_1 x_3$;

$c_{t-1}^{k+1} \leftarrow (x_3,\ x_4,\ d_0,\ d_1)$;

return c_{t-1}^{k+1};

Now we show an important property of the child policy segments.

Theorem 2. *For any $t \in \{2, \ldots, T\}$, given child policy segments $\{c_{t-1}^{k+1}\}_{k=0}^n$ of all the parent policy segments in the policy collection $\mathbf{c}_t = \{c_t^{\star k}\}_{k=0}^n$, we have $c_{t-1}^\star(x) = c_{t-1}^{k+1}(x)$ for some $k \in \{0, 1, \ldots, n\}$ or $c_{t-1}^\star(x) = x$, where $c_{t-1}^\star$ is the optimal policy function at $t-1$.*

Proof. See Appendix A. □

Remark. The theorem shows that all the information of policy function is contained in the set of child policy segments $\{c_{t-1}^{k+1}\}_{k=0}^n$. If we can "refine" the child policy segments in the sense that each policy child is a "part" of the policy function at $t-1$, then the policy collection and policy function at $t-1$ are found.

To refine many child policy segments, we first need to know how to refine two child policy segments. This is where the fusion operator comes in.

Definition 6. The *fusion operator* is a mapping $F : \mathcal{S}_t \times \mathcal{S}_t \times \mathcal{V}_{t+1} \to \mathcal{S}_t \times \mathcal{S}_t$ satisfying:

$$\begin{aligned} & F\Big(c_t^k, c_t^j; v_{t+1}\Big) = (\tilde{c}_t^k, \tilde{c}_t^j), \\ \text{s.t. } & \operatorname{dom}(c_t^k) \cup \operatorname{dom}(c_t^j) \supset \operatorname{dom}(\tilde{c}_t^k) \cup \operatorname{dom}(\tilde{c}_t^j) \\ & \tilde{c}_t^k(x) = c_t^k(x) \text{ for all } x \in \operatorname{dom}(\tilde{c}_t^k), \\ & \tilde{c}_t^j(x) = c_t^j(x) \text{ for all } x \in \operatorname{dom}(\tilde{c}_t^j), \\ & v_t^k(x) \geq v_t^j(x) \text{ for all } x \in \operatorname{dom}(\tilde{c}_t^k) \cap \operatorname{dom}(c_t^j), \\ & v_t^j(x) \geq v_t^k(x) \text{ for all } x \in \operatorname{dom}(c_t^k) \cap \operatorname{dom}(\tilde{c}_t^j), \end{aligned} \tag{7}$$

where $v_t^k(x) := u\big(c_t^k(x)\big) + \beta v_{t+1}\big(\max\{(1+r)(x - c_{t+1}^k(x)) + y_t,\ \underline{x}\}\big)$.

Remark. The fusion operator F does the job of updating the domains of two policy segments such that any of the two returned policy segment is always preferred to the other in its own updated domain.

Before we discuss how to implement the fusion operator F, one theorem and its corollary that can help to simplify the implementation of F are shown below.

Theorem 3. *For any $t \in \{1, \ldots, T\}$, the optimal end-of-period asset function $a_t^\star(x) := x - c_t^\star(x)$ in nondecreasing in x.*

Proof. See Appendix A. □

Corollary 3.1. *For any $t \in \{2, \ldots, T\}$, given child policy segments $\{c_{t-1}^{k+1}\}_{k=0}^n$ of the parent policy segments in the policy collection $\mathbf{c}_t = \{c_t^{\star k}\}_{k=0}^n$, if the optimal policy $c_{t-1}^\star(x) = c_{t-1}^{k+1}(x)$, then $c_{t-1}^\star(x') \neq c_{t-1}^{j+1}(x')$ or $x' \notin \operatorname{dom}(c_{t-1}^{j+1})$ for all $x' > x$ and $j < k$.*

Proof. See Appendix A. □

Remark. Corollary 3.1 shows if the agent switches to a new policy child once her wealth reaches some level, she will not switch back to the original policy child as her wealth increases. It also shows that if the domains of two policy children overlap, to implement the fusion operator F, we only need to identify the potential unique intersection of two candidate value functions derived from the two policy children. Our implementation of the fusion operator F is shown in Algorithm 4.

Algorithm 4: Fusion Operator F

Input : policy children $c_t^j \equiv (x_1,\ x_2,\ b_0,\ b_1)$, $c_t^k \equiv (x_3,\ x_4,\ d_0,\ d_1)$, where $j < k$;

Output: refined policy segment $\tilde{c}_t^j,\ \tilde{c}_t^k$;

Function $v_{\texttt{diff}}(x)$**:**

  $c_j,\ x'_j = c_t^j(x),\ \max\{(1+r)(x - c_t^j(x)) + y_{t+1}, \underline{x}\}$;

  $c_k,\ x'_k = c_t^k(x),\ \max\{(1+r)(x - c_t^k(x)) + y_{t+1}, \underline{x}\}$;

  $v_j, v_k = u(c_j) + \beta v_{t+1}(x'_j), u(c_k) + \beta v_{t+1}(x'_k)$; // use Algorithm 2 for v_{t+1}

return $v_j - v_k$

if $x_2 \le x_3$ *or* $x_1 \ge x_4$ **then**

  $\tilde{c}_t^j, \tilde{c}_t^k \leftarrow c_t^j, c_t^k$

else

  if $v_{\texttt{diff}}(x_1) \le 0$ *and* $v_{\texttt{diff}}(x_2) \le 0$ **then**

    $\tilde{c}_t^k \leftarrow c_t^k$; // c_t^k dominates c_t^j on $[x_3, x_4]$

    $\tilde{c}_t^j \leftarrow x_1 < x_3\ ?\ (x_1, x_3, b_0, b_1) : \emptyset$; // $\emptyset$ means the domain of $\tilde{c}_t^j$ is empty

  else if $v_{\texttt{diff}}(x_1) \ge 0$ *and* $v_{\texttt{diff}}(x_2) \le 0$ **then**

    $kink \leftarrow$ root of $v_{\texttt{diff}}(x) = 0$;

    $\tilde{c}_t^j, \tilde{c}_t^k \leftarrow (x_1, kink, b_0, b_1), (kink, x_4, d_0, d_1)$;

  else

    $\tilde{c}_t^j \leftarrow c_t^j$; // c_t^j dominates c_t^k on $[x_1, x_2]$

    $\tilde{c}_t^k \leftarrow x_2 < x_4\ ?\ (x_2, x_4, b_0, b_1) : \emptyset$; // $\emptyset$ means the domain of $\tilde{c}_t^k$ is empty

  end

end

return $\tilde{c}_t^j,\ \tilde{c}_t^k$

Notice that in Algorithm 4, we can use the Brent method for root finding. The Brent method is guaranteed to converge at super linear speed in Algorithm 4 since the two value segments can have at most one intersection point in the bracket by Corollary 3.1.[7]

Now we introduce the glue operator G which can refine the set of child policy segments to the policy collection by implementing the fusion operator F repeatedly (See Algorithm 5).

Definition 7. The *glue operator* is a mapping $G : \mathfrak{S}_t \times \mathcal{V}_{t+1} \to \mathfrak{S}_t$ such that:

$$G\Big(\{c_t^{k+1}\}_{k=0}^n; v_{t+1}\Big) = \{c_t^{\star j}\}_{j=0}^m \tag{8}$$

where $\{c_t^{\star j}\}_{j=0}^m$ is the policy collection $\mathbf{c}_t$ refined from set of child policy segments $\{c_t^{k+1}\}_{k=0}^n$.

[7] For the special case $x2 = x4 = \infty$, we can add a while loop to find the bracket that contains the root. See section 9.3 of Press et al. (2007) for an introduction to the Brent method.

Algorithm 5: Glue Operator G

Input : child policy segments $\{c_t^{k+1}\}_{k=0}^{n}$, parameters in DP problem (3);

Output: policy collection $\mathbf{c}_t$;

$\mathbf{c_t} = \{c_t^0\}$; `// initialize the policy collection as saving nothing`

for $k = 0 : n$ **do**

  $c_t^j \leftarrow pop(\mathbf{c_t})$; `// pop the last policy segment from the set` $\mathbf{c_t}$

  $c_t^j, c_t^k \leftarrow F(c_t^j, c_t^k; v_{t+1})$;

  if $c_t^j \neq \emptyset$ *and* $c_t^k \neq \emptyset$ **then**

    append (c_t^j, c_t^k) to the end of $\mathbf{c_t}$;

    append c_t^k to the end of $\mathbf{c_t}$;

  else if $c_t^k \neq \emptyset$ **then**

    while $c_t^j = \emptyset$ **do**

      $c_t^j \leftarrow pop(\mathbf{c_t})$;

      $c_t^j, c_t^k \leftarrow F(c_t^j, c_t^k; v_{t+1})$;

    end

    append (c_t^j, c_t^k) to the end of $\mathbf{c_t}$;

  else

    append c_t^j to the end of $\mathbf{c_t}$;

  end

end

return $\mathbf{c_t}$;

Now the policy function at any period t can be traced through the policy tree by using operator M and G interchangeably. The Policy-Tree Descendant (PTD) algorithm to solve the policy function is shown in Algorithm 6.

Algorithm 6: PTD algorithm for deterministic DP problem

Input : parameters in DP problem (3) ;

Output: the list of policy collections at each period $pl \equiv [\mathbf{c}_T, \mathbf{c}_{T-1}, \cdots, \mathbf{c}_1]$;

$pl = [c_T^0,]$; `// period T policy is consuming all the wealth`

for $t = T - 1 : 1$ **do**

  $\{c_t^{k+1}\}_{k=0}^{\#\mathbf{c}_{t+1}} \leftarrow M(\mathbf{c}_{t+1})$; `// apply` M `to each policy segment in` $\mathbf{c}_{t+1}$

  $\mathbf{c}_t \leftarrow G(\{c_t^{k+1}\}_{k=0}^{\#\mathbf{c}_{t+1}}; v_{t+1})$;

  append $\mathbf{c}_t$ to the end of pl;

end

return pl;

3.4 Performance of PTD solver for the deterministic model

Now we show the accuracy and speed of the PTD algorithm by solving the deterministic DP problem (3) with parameters settings in Section 3.2, i.e., $\beta = 0.98,\ \gamma = 1,\ r = 0.1,\ \underline{x} = 3, y_t = 1$ for all $t \in \{1, \ldots, 50\}$. As a comparison, we also use the DC-EGM algorithm[8] (Iskhakov et al., 2017) to solve the same problem. The reason we choose EGM rather than VFI with grid search as comparison is that the inaccuracy and inefficiency of the latter has been well documented in the EGM literature (Iskhakov et al., 2017; Druedahl and Jørgensen, 2017). As far as we know, the EGM based algorithms are the most accurate and efficient methods to solve the non-convex DP problems in the literature. Our implementation of DC-EGM for solving (3) is described in Appendix B. All the numerical exercises in this paper are coded in Julia 1.10 and run on a laptop with Intel Core i7-13700H CPU.

PTD is a grid-free algorithm, it will generates the global policy function. To implement EGM, we need to set up the exogenous end-of-period asset a grid in advance. Assuming we are interested in the policy function at $t = 1$ for the wealth state x in the region $[0,\ 50]$, The intuitive choice is using a uniform a grid on $[0,\ 50]$, to increase the accuracy, we can also expand the a grid. We show the running time of PTD and EGM under three settings of a grid in Table 1. We also draw the policy functions at $t = 1$ in Figure 5.

As Table 1 shows, PTD is significantly faster than EGM even if we only use 100 grid points in the latter algorithm. This is because for PTD, the descendant operator M can be implemented with ignorable computational cost, the only computational burden comes from implementing the fusion operator F and glue operator G, which can be efficiently controlled since we pre-save the $v_t(\underline{x})$. On contrast, EGM needs to save the policy and value functions on grids, partitioning the non-monotonic endogenous state grids, and loop through all the state grids to construct the upper envelope of value function for each iteration. These non-trivial tasks and related memory allocations make EGM much slower than PTD. doing interpolation and extrapolation To compare the accuracy of PTD and EGM, we run a imaginary "utility contest", that is, we give the initial wealth state x uniformly spaced between $[0,\ 50]$ to 5000 agents in the model with parameters described above, and let them use policy functions derived from PTD and EGM respectively. Thus, we can observe which algorithm brings the agents highest summation of life-time discounted utility. The number of winners column in Table 1 shows the number of agents get highest utility from each algorithm. If several algorithms achieve the highest utility together, then all of them are winners[9]. As we can see from Table 1, for each level of initial

[8] We will abbreviate the "DC-EGM algorithm" as "EGM" in the following text.

[9] In the numerical exercises, we set $1e-12$ as the tolerance level when we compare two Float numbers, i.e., we regard two utility levels are equivalent if their difference is below $1e-12$.

wealth, PTD always brings the agent highest life-time utility. For EGM, the trade-off between accuracy and speed is obvious, which requires the users to "finely tune" the a grid to get the best performance for each specific parameter settings. However, the "out-of-box" PTD algorithm always has the best performance.

Table 1: Performances of PTD and EGM for the deterministic model

Method	**Grids of a**	**Time (ms)**	**# winners**
PTD		1.75	5000
EGM1	range(0.0, 50.0, 100)	5.38	4821
EGM2	range(0.0, 100.0, 200)	10.64	4991
EGM3	range(0.0, 200.0, 400)	21.32	4993

Notes: $range(x, y, n)$ means n points evenly spaced between x and y. The running time is the average of 100 samples. The utility contests are run for 5000 initial wealth levels evenly spaced between 0.1 and 50.

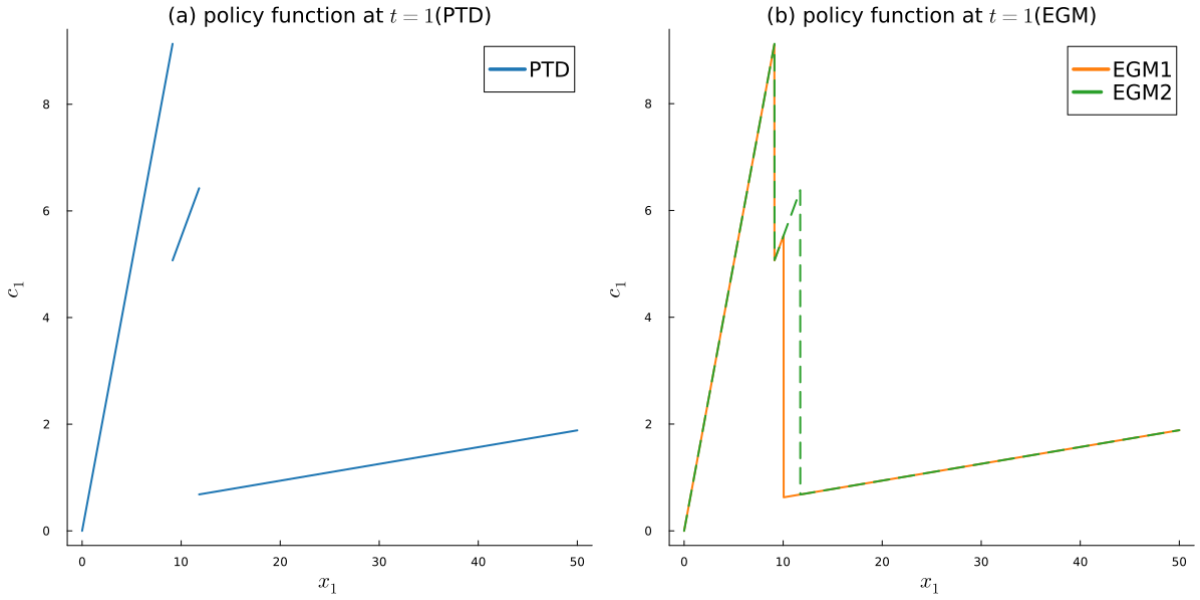

Figure 5: Policy functions derived from PTD and EGM

Notes: We didn't draw the policy function of EGM3 for clarity of the plot since it is very close to EGM2.

4 Policy-Tree Descendant Algorithm for Stochastic Models

4.1 Main Idea of the Simulation Method

After describing the PTD algorithm for the deterministic DP problem, now we can tackle the stochastic problem (2). Before entering the details, we first introduce the distinctive feature of PTD algorithm, that is, **simulation-based solver**. For algorithms like EGM and VFI, to evaluate time t policy function at a state $(x_t, \boldsymbol{s}_t)$, we need to approximate the gloabl policy function on state grids first, then interpolate the policy function at the particular state. PTD

avoids this time-consuming and inaccurate process by simulating the state-policy path starting from state $(x_t, \boldsymbol{s}_t)$ directly.

We show the main idea of the simulation-based solver in Figure 6. Assuming we are interested in the consumption-saving path of the agent described by (2) starting from the state $(x_1, \boldsymbol{s}_1)$ at time $t = 1$, where the actual basic uncertainties sequence is $\{\boldsymbol{\varepsilon}_t\}_{t=2}^T$[10], i.e., the agent observes the realization of $\boldsymbol{\varepsilon}_t$ at the start of time t.

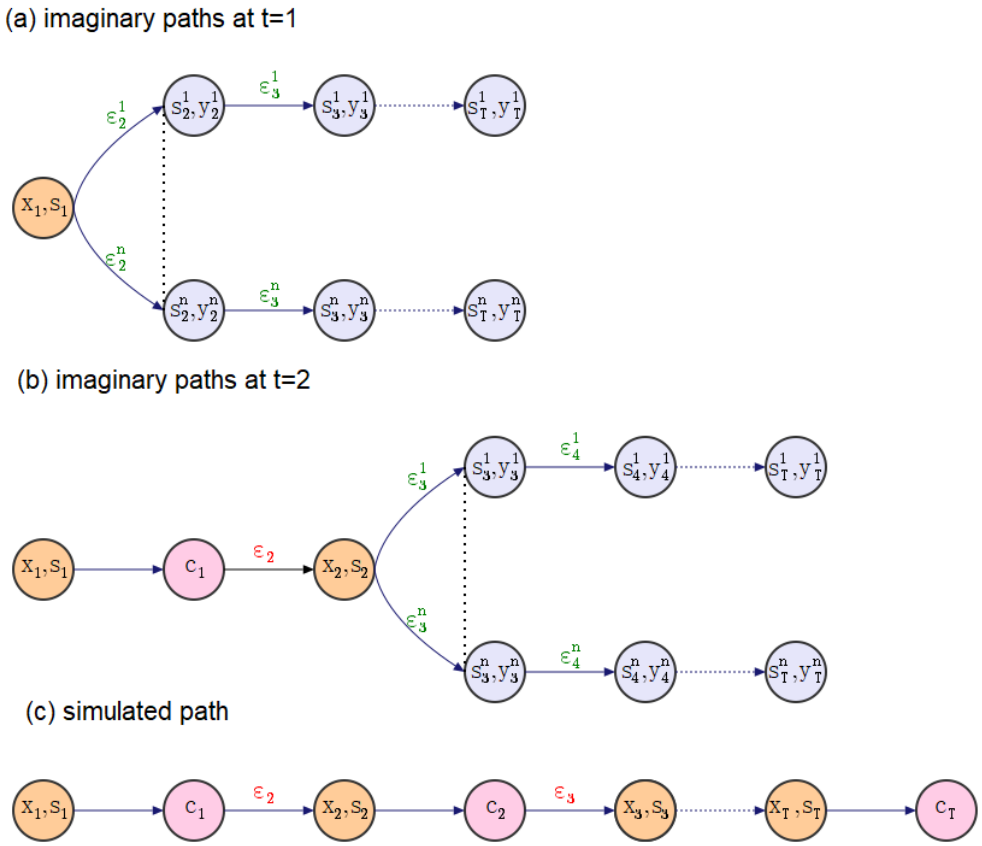

Figure 6: Simulation Algorithm

At $t = 1$, to find the optimal consumption, the agent uses the following steps:

(i.) The agent randomly draw n realizations of the basic uncertainties paths. We denote $\{\boldsymbol{\varepsilon}_t^i\}_{t=2}^T$ as the i-th realization, where $i \in \{1, \ldots, n\}$. For each realization of the basic uncertainties sequence, the agent faces a deterministic DP problem (3) on a "imaginary path"[11] with income sequence $\{y_t^i\}_{t=2}^T$ pinned down by the initial state $(x_1, \boldsymbol{s}_1)$ and shocks $\{\boldsymbol{\varepsilon}_t^i\}_{t=2}^T$. We

[10] With a little abuse of notation, we use $\boldsymbol{\varepsilon}_t$ to denote the realization of uncertainties here rather than random vector itself.

[11] We use the term "imaginary path" to avoid confusion with the "simulated path"generated by the actual shocks $\{\boldsymbol{\varepsilon}_t\}_{t=2}^T$.

draw the two imaginary paths for $i = 1$ and $i = n$ in Figure 6 (a).

(ii.) She solves the deterministic DP problem using the PTD algorithm (Algorithm 6) on each imaginary path, i.e., she has the policy function at $t = 2$ on each imaginary path.

(iii.) She tries to find a consumption level c_1^{int} such that the marginal utility of c_1^{int} is the same as the adjusted "average" marginal utility of the optimal consumption at $t = 2$ on the imaginary paths given her end-of-period asset at $t = 1$ is $x - c_1^{int}$.

(iv.) If c_1^{int} exists, she will compare whether $c_1 = c_1^{int}$ or $c_1 = x_1$ (relying on social insurance at $t = 2$) will bring her higher life-time discounted utility. If c_1^{int} does not exist, she will consume all her wealth at $t = 1$.

With optimal consumption at $t = 1$ found, the agent has state $(x_2, \boldsymbol{s}_2)$ at $t = 2$ after the realization of $\boldsymbol{\varepsilon}_2$. She faces the similar problem shown in Figure 6 (b) at $t = 2$ as at $t = 1$. She also uses steps (i)-(iv), i.e., generating n imaginary paths starting from $(x_2, \boldsymbol{s}_2)$, to find the optimal consumption c_2. The process continues until $t = T$, and we find the whole simulated consumption-saving path of the agent as shown in Figure 6 (c). In the next subsection, we formally describe the PTD algorithm for the stochastic DP problem (2).

4.2 PTD algorithm for the stochastic DP problem

The key ingredient of PTD algorithm for the stochastic DP problem is a special version of stochastic Euler equation, which connects the stochastic DP problem with the deterministic DP problem. This stochastic Euler equation allows us to evaluate the optimal policy function with a manner like time iteration (Coleman, 1990; Li and Stachurski, 2014). We first introduce some notations.

Notations. We denote $\boldsymbol{\varepsilon}_{t:T} := [\boldsymbol{\varepsilon}_t, \boldsymbol{\varepsilon}_{t+1}, \ldots, \boldsymbol{\varepsilon}_T]$, the product measure $\mu_{t:T} := \bigotimes_{i=t}^{T} \mu$. We write $v_t(x; \boldsymbol{y}_{t+1:T})$ as the deterministic value function in (3) with income sequence $\boldsymbol{y}_{t+1:T} := [y_{t+1}, y_{t+2}, \ldots, y_T]$. To save notations, in the following text, we don't distinguish random variables and their realizations if the context is clear. We also write $\mu_{t:T}(\boldsymbol{\varepsilon}_{t:T})$ as $\mu(d\boldsymbol{\varepsilon}_{t:T})$ for brevity if no confusion arises.

To connect the stochastic DP problem with the deterministic DP problem, we use the following simple lemma.

Lemma 1. *In (2), denote* $x_{t+1}(\boldsymbol{\varepsilon}_{t+1}) = \max\big(z_{t+1}(\boldsymbol{\varepsilon}_{t+1}), \underline{x}\big)$*, we have:*

$$\int \omega_{t+1}\Big(x_{t+1}(\boldsymbol{\varepsilon}_{t+1}), \boldsymbol{s}_{t+1}\Big)\mu(d\boldsymbol{\varepsilon}_{t+1}) = \int v_{t+1}\Big(x_{t+1}(\boldsymbol{\varepsilon}_{t+1}); \boldsymbol{y}_{t+1:T}(\boldsymbol{\varepsilon}_{t+1:T})\Big)\mu_{t+1:T}(d\boldsymbol{\varepsilon}_{t+1:T}), \quad (9)$$

where the mapping $\boldsymbol{y}_{t+1:T}(\boldsymbol{\varepsilon}_{t+1:T})$ *means the income sequence* $\boldsymbol{y}_{t+1:T}$ *is determined by the exogenous state* $\boldsymbol{s}_t$ *and the following shocks* $\boldsymbol{\varepsilon}_{t+1:T}$.

Proof. See Appendix A. □

Lemma 1 allow us to reformulate the stochastic DP problem (2) as:

$$\begin{aligned}
\omega_t(x_t, \boldsymbol{s}_t) &= \max_{c_t} \Big\{ u(c_t) + \beta \int v_{t+1}\Big(\max\{z_{t+1}(\boldsymbol{\varepsilon}_{t+1}), \underline{x}\}; \boldsymbol{y}_{t+1:T}(\boldsymbol{\varepsilon}_{t+1:T})\Big) \mu(d\boldsymbol{\varepsilon}_{t+1:T}) \Big\}, \\
\text{s.t.} \quad a_t &= x_t - c_t, \\
z_{t+1} &= (1+r)a_t + y_{t+1}, \\
y_{t+1} &= f(\boldsymbol{s}_{t+1}), \\
\boldsymbol{s}_{t+1} &= g(\boldsymbol{s}_t, \boldsymbol{\varepsilon}_{t+1}), \\
0 &\le c_t \le x_t, \\
\omega_T(x_T, \boldsymbol{s}_T) &= u(x_T) \quad \text{for all } (x_T, \boldsymbol{s}_T) \in \mathbb{R}_+ \times \mathbb{R}^k,
\end{aligned} \tag{10}$$

where $t \in \{1, \dots, T-1\}$, v_{t+1} is defined in (3).

The dynamic optimization problem (10) has a slightly different interpretation from its equivalence (2). The agent with state $(x_t, \boldsymbol{s}_t)$ at time t tries to find the optimal consumption c_t at time t such that the sum of the current period utility $u(c_t)$ and the expected discounted utility of the optimal consumption stream over the distribution of all future shocks is maximized.

We show another simple lemma which helps to establish the stochastic Euler equation.

Lemma 2. *For any time* $t \in \{1, \dots, T\}$, *wealth level* $x > 0$ *and income vector* $\mathbf{y}_{t+1:T}$ *such that the optimal consumption level* $c_t^\star(x; \mathbf{y}_{t:T}) < x$, *then* $v_t(x; \mathbf{y}_{t:T})$ *is differentiable at* x. *Moreover,* $\frac{\partial v_t(x;\ \mathbf{y}_{t:T})}{\partial x} = u'(c_t^\star(x; \mathbf{y}_{t:T}))$.

Proof. See Appendix A. □

Since the policy function $c_t^\star(x; \mathbf{y}_{t:T})$ has only finite number of discontinuities, so does $u'(c_t^\star(x; \mathbf{y}_{t:T}))$, we can differentiate the right hand side of (10) if the optimal consumption $c_t < x_t$, the first order condition (F.O.C.) is:

$$u'(c_t) = \beta(1+r) \int \mathbb{1}\big(z_{t+1}(\boldsymbol{\varepsilon}_{t+1}) > \underline{x}\big) \frac{\partial v'_{t+1}\big(z_{t+1}(\boldsymbol{\varepsilon}_{t+1}); \boldsymbol{y}_{t+1:T}(\boldsymbol{\varepsilon}_{t+1:T})\big)}{\partial z_{t+1}} \mu(d\boldsymbol{\varepsilon}_{t+1:T}). \tag{11}$$

Combining Lemma 2 and (11), we have the stochastic Euler equation.

Theorem 4. *For any $t \in \{1, \dots, T-1\}$ and wealth level x such that $c_t^\star(x_t) < x_t$, the stochastic Euler equation holds:*

$$u'\big(c_t^\star(x_t, \boldsymbol{s}_t)\big) = \beta(1+r) \int \mathbb{1}\big(z_{t+1}(\boldsymbol{\varepsilon}_{t+1}) > \underline{x}\big) u'\Big(c_{t+1}^\star\big(z_{t+1}(\boldsymbol{\varepsilon}_{t+1}); \boldsymbol{y}_{t+1:T}(\boldsymbol{\varepsilon}_{t+1:T})\big)\Big) \mu(d\boldsymbol{\varepsilon}_{t+1:T}) \tag{12}$$

Proof. This is the direct implication of (11) and Lemma 2. □

Remark. Notice that the policy function on the left hand side of (12) is the policy function in the stochastic DP problem (2), which on the right hand side is the policy function on the imaginary path in the deterministic DP problem (3). The intuition behind (12) is that at the optimal consumption level, the utility loss of saving one dollar today equals the expected utility gain of consuming one dollar tomorrow on the imaginary paths such that the agent's wealth being beyond the consumption floor tomorrow.

Since we already have a fast and accurate solver for the deterministic DP problem, for any consumption level c_t at the state $(x_t, \boldsymbol{s}_t)$, the right hand side of (12) can be efficiently calculated by Monte Carlo integration. This is especially true for models like Example 1, Example 2 and Example 3 since $\boldsymbol{\varepsilon}_t$ has normal or lognormal distributions, and the high-dimensional Monte Carlo integration can be highly efficient in these cases (Tang, 2023).

Algorithm 7: Monte Carlo Integration

Input : $x_t, \boldsymbol{s}_t, c_t$, parameters in DP problem (10),
number of samples in the integral region of (12) n;

Output: r.h.s. in the stochastic Euler equation (12);

$rhs = 0,\ i = 0$;
while $i \le n$ **do**
  draw $\boldsymbol{\varepsilon}_{t+1:T}^i$ from the distribution of $\boldsymbol{\varepsilon}_{t+1:T}$;
  $z_{t+1} \leftarrow (1+r)(x_t - c_t) + y_{t+1}(\boldsymbol{\varepsilon}_{t+1})$;
  if $z_{t+1} > \underline{x}$ **then**
    $rhs \leftarrow rhs + \frac{1}{n}\beta(1+r)u'\Big(c_{t+1}^\star\big(z_t + 1; y_{t+1}(\boldsymbol{\varepsilon}_{t+1})\big)\Big)$;
  end
  $i \leftarrow i + 1$;
end
return rhs;

Notice that we use Algorithm 6 to get the policy functions on each imaginary path in Algorithm 7. We also use Monte Carlo integration to evaluate the maximand of the right hand

side of (10). We skip the algorithm description here since it is very similar to Algorithm 7.

Now we show how to solve the stochastic Euler equation (12). Notice that for a particular state $(x_t, \boldsymbol{s}_t)$, (12) may have zero, one or two roots. (12) can have two roots since the r.h.s. of (12) is first increasing then decreasing in c_t. The increasing part is due to the increasing expected marginal utility of c_{t+1} since the end-of-period asset a_t decreases as c_t increases. The decreasing part is due to the the decreasing a_t makes the measure of $\{\boldsymbol{\varepsilon}_{t+1} : z_{t+1}(\boldsymbol{\varepsilon}_{t+1}) > \underline{x}\}$ smaller. Thus, we can use a combination of bisection method and brent method to find the roots as shown in Algorithm 8.

Algorithm 8: Roots Finding for the Stochastic Euler Equation

Input : $x_t, \boldsymbol{s}_t$, parameters in DP problem (10),
number of samples in the integral region of (12) n,
maximum number of brackets in root findings $mMax$;

Output: roots of the stochastic Euler equation (12);

draw n samples $\{\boldsymbol{\varepsilon}^j_{t+1:T}\}_{j=1}^n$ for evaluating r.h.s. of (12) ; `// Algorithm 7`

$roots = [\,],\ m = 2$; `// initialize empty list of roots`

while $m \leq mMax$ ***and*** *roots is empty* **do**
  split $[0, x_t]$ into m brackets $b_1, \ldots, b_m$ evenly;
  for $i = 1 : m$ **do**
    if *root exists on* b_m **then**
      append the root to *roots*;
    end
  end
  $m \leftarrow 2m$;
end
return *roots*;

Notice that we set $mMax = 128$ in all the numerical experiments, which makes the possibility of not finding the roots but the roots exist extremely low. In the root finding process, we draw the shock sequences $\{\boldsymbol{\varepsilon}^j_{t+1:T}\}_{j=1}^n$ in Algorithm 7 once and use them repeatedly to save the computational cost of solving policy functions on the imaginary paths generated by the shock sequences.

After we know how to solve the stochastic Euler equation, we can find the optimal consumption policy for any given state, and generate the simulated wealth-consumption path for any given initial state and shocks sequence as shown in Algorithm 9.

Algorithm 9: PTD simulator for stochastic DP problem

Input : parameters in DP problem (2);

initial state variables $(x_1, \boldsymbol{s}_1)$;

shock sequence $\{\boldsymbol{\varepsilon}_t\}_{t=2}^{T}$;

Output: simulated wealth path $\boldsymbol{x}$, consumption path $\boldsymbol{c}$;

1 $\boldsymbol{x} = [x_1,]$, $\boldsymbol{c} = [\ ]$;

2 **for** $t = 1 : T-1$ **do**

3 $\quad$ *candidates* $\leftarrow$ solutions of (12); // Algorithm 8

4 $\quad$ append x_t to *candidates*;

5 $\quad$ $c_t \leftarrow \underset{c \in candidates}{\text{argmax}} \left\{ u(c_t) + \beta \int v_{t+1}\Big(\max\{z_{t+1}(\boldsymbol{\varepsilon}_{t+1}), \underline{x}\}; \boldsymbol{y}_{t+1:T}(\boldsymbol{\varepsilon}_{t+1:T}) \Big) \mu(d\boldsymbol{\varepsilon}_{t+1:T}) \right\}$;

6 $\quad$ append c_t to $\boldsymbol{c}$;

7 $\quad$ $x_{t+1} \leftarrow \max\{z_t(\boldsymbol{\varepsilon}_t), \underline{x}\}$;

8 $\quad$ append x_{t+1} to $\boldsymbol{x}$;

9 **end**

10 append x_T to $\boldsymbol{c}$; // consuming all the wealth at T

11 **return** $\boldsymbol{x}$, $\boldsymbol{c}$;

Notice that in Line 4, we consider the case that consuming all the wealth might be the optimal policy. In Line 5, we use Monte Carlo integration similar to Algorithm 7 to calculate the maximand value for each candidate consumption level.

4.3 Performance of the PTD Simulator for the Stochastic DP Problem

Now we examine the performance of the PTD simulator in Example 1, Example 2 and Example 3. We also use EGM as the comparison. In all the numerical experiments below, we keep the preference parameters and the consumption floor same as in Section 3, i.e., $T = 50,\ \beta = 0.98,\ \gamma = 1,\ r = 0.1,\ \underline{x} = 3.0$.

Example 1 (IID). We consider $y_t \sim \text{lognormal}(0, \sigma)$, where $\sigma \in \{0.1, 0.5\}$. The settings for PTD and EGM algorithms are show in Table 2.

Table 2: Simulation Time of PTD and EGM for the IID model

Method	Grids of a	n_{mc}	Time (s) for $\sigma = 0.1$	Time (s) for $\sigma = 0.5$
PTD1		100	0.119	0.123
PTD2		200	0.273	0.282
PTD3		1000	1.632	1.784
EGM1	range(0.0, 100.0, 200)		1.218	0.764
EGM2	range(0.0, 200.0, 400)		1.981	1.207
EGM3	range(0.0, 1000.0, 2000)		9.617	4.930

We run simulations for the consumption-saving path for 100 agents with initial wealth evenly spaced between 0 and 50 using PTD algorithm with different number of points in Monte Carlo integration (n_{mc} column in Table 2) and EGM algorithm with different end-of-period asset grids (Grids of a column). The average running times of 5 samples are also recorded in Table 2. Notice that PTD algorithm is much faster than EGM algorithm.

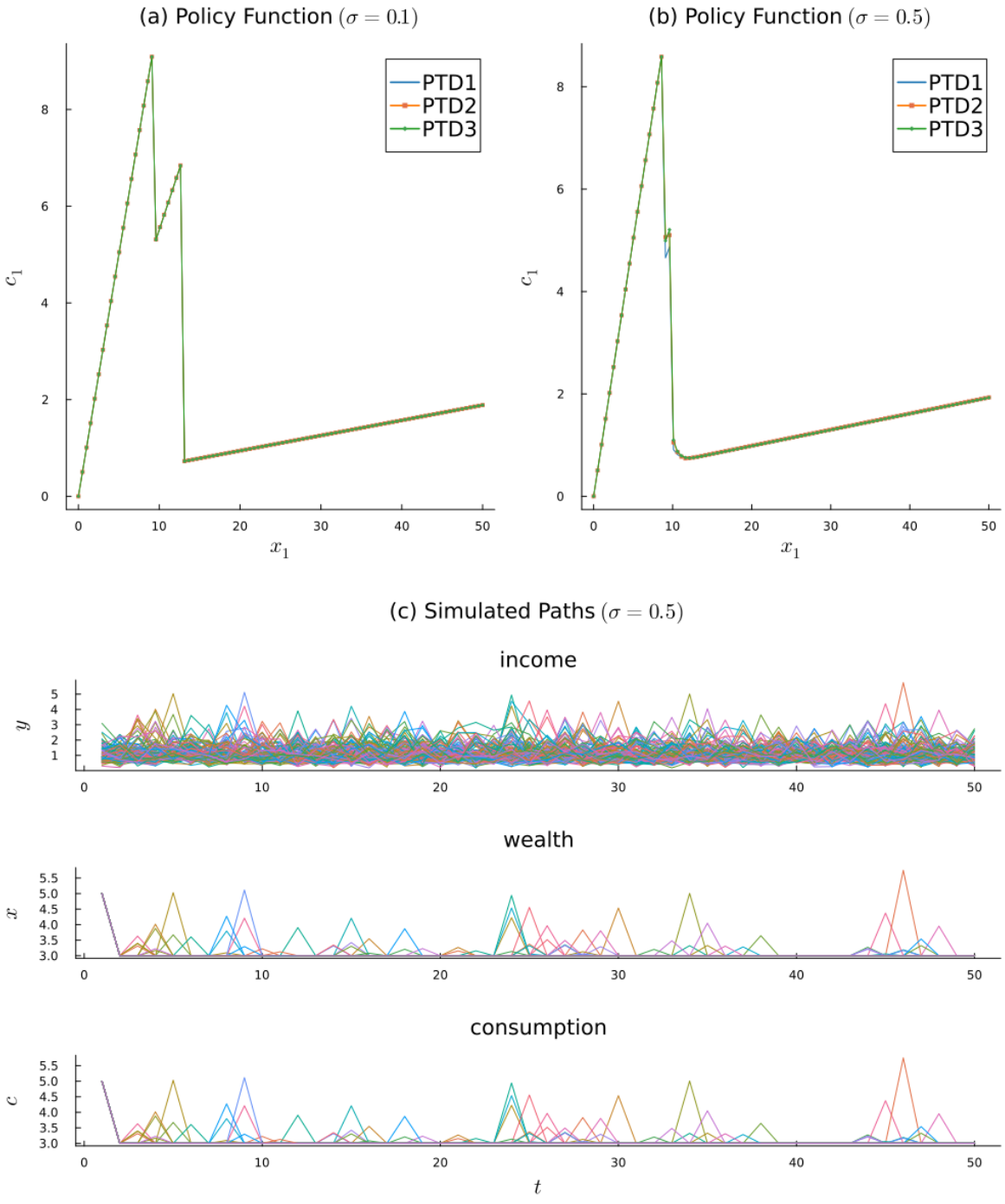

Figure 7: Policy functions and simulation results for IID model (PTD)

Now we analyze the numerical performance of PTD and EGM algorithms in detail. We first

consider the policy function at $t = 1$ and simulation path derived by PTD algorithm as shown in Figure 7. Notice that PTD algorithm is pretty stable for different number of points used in Monte Carlo integration (n_{mc}) though there are slight differences among three implementations at the region near the second kink for $\sigma = 0.5$. This means we can get quite accurate simulation results even if we use 100 points in Monte Carlo integration, which only costs 0.1 second.

The simulated paths for the 100 agents ($\sigma = 0.5, n_{mc} = 100$) are shown in Figure 7 (c). Notice that the standard deviation of transitory income shocks are not high enough to make people start to save. Agents consume almost all the transitory income since they realize the next period income will be closer to 1.0, which makes relying on social insurance tomorrow a better choice than suppressing consumption today and saving for tomorrow.

Next We consider the policy function at $t = 1$ and simulation path derived by PTD algorithm as shown in Figure 8. The policy functions for $\sigma = 0.1$ are shown in Figure 8 (a). Intuitively, the policy function under this case is very close to which of the deterministic model[12], and the finer end-of-period asset grid will make the policy function more accurate. However, on contrary to the intuition, the policy function under the end-of-period asset grid $range(0.0, 200, 400)$ is totally wrong though the coarser end-of-period asset grid $range(0.0, 100, 200)$ generates the correct policy function. The reason is that very low level of end-of-period asset a generates a very small value of the right hand side of the Euler equation (12), which makes value function wrong calculated on the very large endogenous wealth grids. This make the value function miscalculated on grids near $x = 200$, and in turn generates a miscalculated kink in the value function. As we increased the upper bound of the end-of-period asset grid to 1000, the "wrong" values are dropped by the upper envelope calculation, and the policy function behaves correctly again. However, this robust implementation of EGM algorithm with end-of-period asset grid as $range(0.0, 1000, 2000)$ is very slow, which costs almost 10 seconds for $\sigma = 0.1$ and 5 seconds for $\sigma = 0.5$ to simulate the consumption-saving paths for 100 agents.

For $\sigma = 0.5$, there are significant differences for the tail part of policy functions generated by the three EGM implementations as shown in Figure 8 (b). This is due to the interpolation and extrapolation errors, which can be mitigated by using finer end-of-period asset grids at the cost of longer running time.

We also draw the simulated paths for the 100 agents ($\sigma = 0.5$) using the finest end-of-period asset grid (EGM3) in Figure 8 (c). The realizations of shocks are kept as the same as in Figure 7 (c). The simulated paths are almost the same as those generated by PTD algorithm (PTD1) as

[12]Notice that Iskhakov et al. (2017); Druedahl and Jørgensen (2017) choose $\sigma = 0.01$, which makes the stochastic model almost the same as the deterministic model.

shown in Figure 7 (c) though the latter is almost 100 times faster than the former.

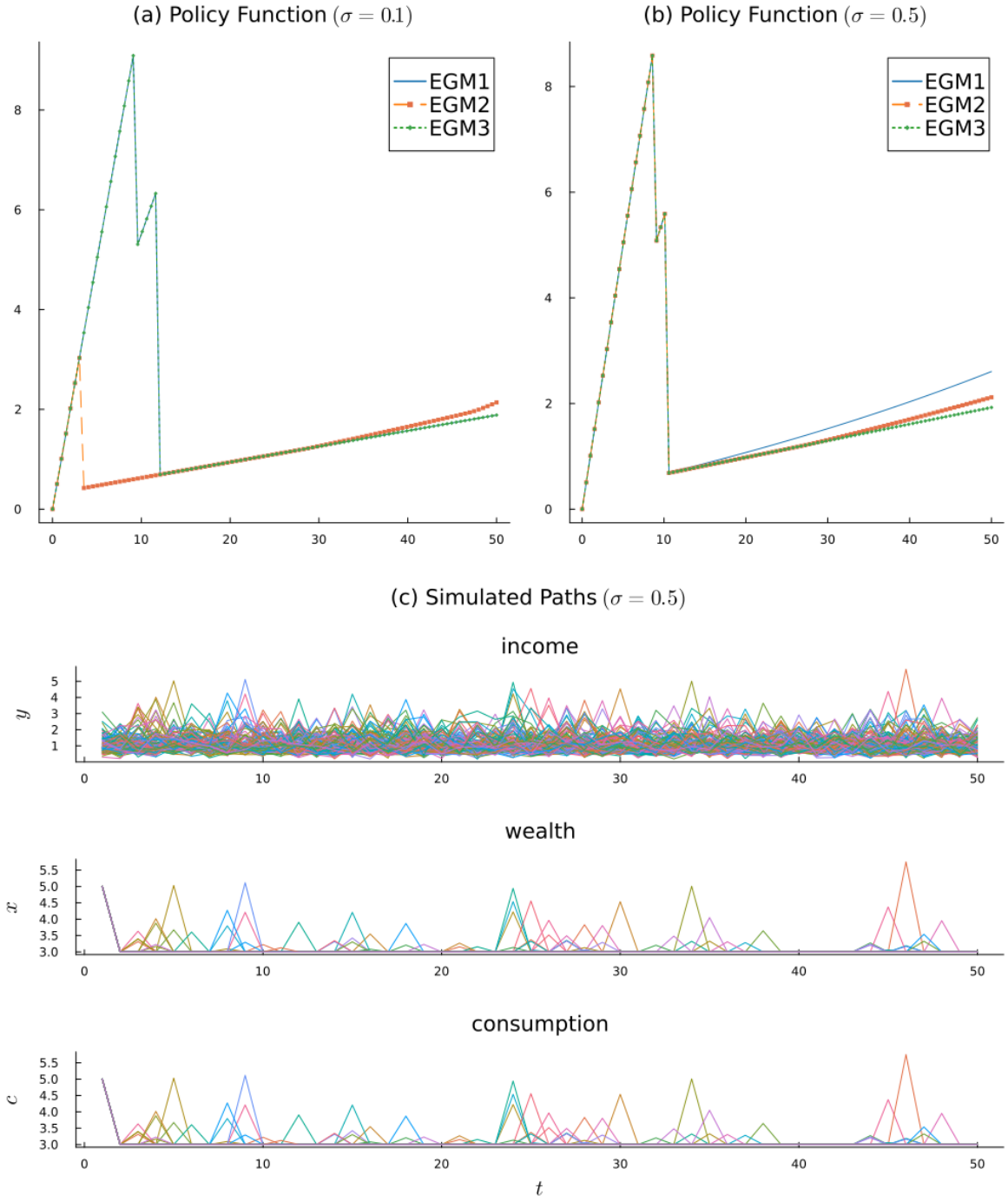

Figure 8: Policy functions and simulation results for IID model (EGM)

To further compare the accuracy of PTD and EGM algorithms, we run a "utility contest" for the two algorithms by letting 1000 agents with wealth evenly spaced on $[0.1, 50]$ using consumption policies derived from the six implementations of the two algorithms. We can observe which algorithm generates the highest expected lifetime discounted utility at $t = 1$. We use 1000 points in Monte Carlo integration to calculate the maximand of r.h.s. of (10), and use it as the

expected utility since the deterministic PTD solver is accurate. For each agent, if an algorithm generates the highest utility, we give it one "gold medal"; if it generates the second-highest utility, we give it one "silver medal"; if it generates the third-highest utility, we give it one "bronze medal".

As Table 3 shows without any surprise, PTD3 ($n_{mc} = 1000$) is the most accurate simulation algorithm, and it is almost 3 times faster than the second most accurate algorithm EGM3.

Table 3: Utility Contest (IID model, $\sigma = 0.5$, 6 methods)

Method	**# Gold**	**# Silver**	**# Bronze**
PTD1	179	10	38
PTD2	186	38	447
PTD3	979	13	3
EGM1	178	20	6
EGM2	179	233	79
EGM3	176	513	252

To further investigate the relative accuracy between PTD with fewer points in Monte Carlo integration and EGM with coarser end-of-period asset grids, we also run a "mini contest" for PTD1, PTD2, EGM1 and EGM2. The results are shown in Table 4. Notice that PTD2 is more accurate than EGM2, and it is 4 times faster than EGM2. PTD1 is also more accurate than EGM1, and it is 6 times faster than EGM1.

Table 4: Utility Contest (IID model, $\sigma = 0.5$, 4 methods)

Method	**# Gold**	**# Silver**	**# Bronze**
PTD1	187	472	319
PTD2	658	321	21
EGM1	194	10	40
EGM2	486	23	446

To sum up our discussion for IID model, out-of-box PTD algorithm is fast, accurate and robust. On contrary, EGM algorithm is slower and less accurate than PTD algorithm in general. And researchers always need to carefully choose the end-of-period asset grids before using EGM algorithm, which is not necessary for PTD algorithm. A final concern for the speed of PTD algorithm is related to large scale simulation, for example, if we need to simulate 10,000 agents' consumption-saving profiles (this is a possible scenario for solving heterogeneous agent models using algorithms in Krusell and Smith (1998)), the speed of PTD algorithm will be slower the EGM due to the accurate simulation-based solver of PTD algorithm. However, this limitation of PTD algorithm can be easily overcome by allowing interpolation in PTD algorithm. We introduce the interpolation mode of PTD algorithm (PTD-I) in Appendix C.

Example 2 (AR1). In the example with AR(1) log income process. We set the autoregressive coefficient $\rho = 0.9$, white noise has a normal distribution $\mathcal{N}(0, 0.5)$ truncated on the support $(-\infty, 3.0]$. The truncated distribution is used to prevent the wealth level in simulation goes to extremely large due to the self-dependence of income process. We also simulate 100 agents' consumption-saving paths with initial wealth 5.0 and initial income 1.0. We use 100 points for Monte Carlo integration in PTD algorithm and three different end-of-period asset grids for EGM algorithm. The simulation time is shown in Table 5.

Table 5: Simulation Time of PTD and EGM for the AR1 model

Method	**Grids of a**	**Grids of y**	n_{mc}	$Time(s)$
PTD1			100	59.872
EGM1	range(0.0, 100.0, 200)	range(0.0, 70.0, 140)		88.410
EGM2	range(0.0, 200.0, 400)	range(0.0, 70.0, 140)		206.274
EGM3	range(0.0, 600.0, 1200)	range(0.0, 70.0, 140)		369.701

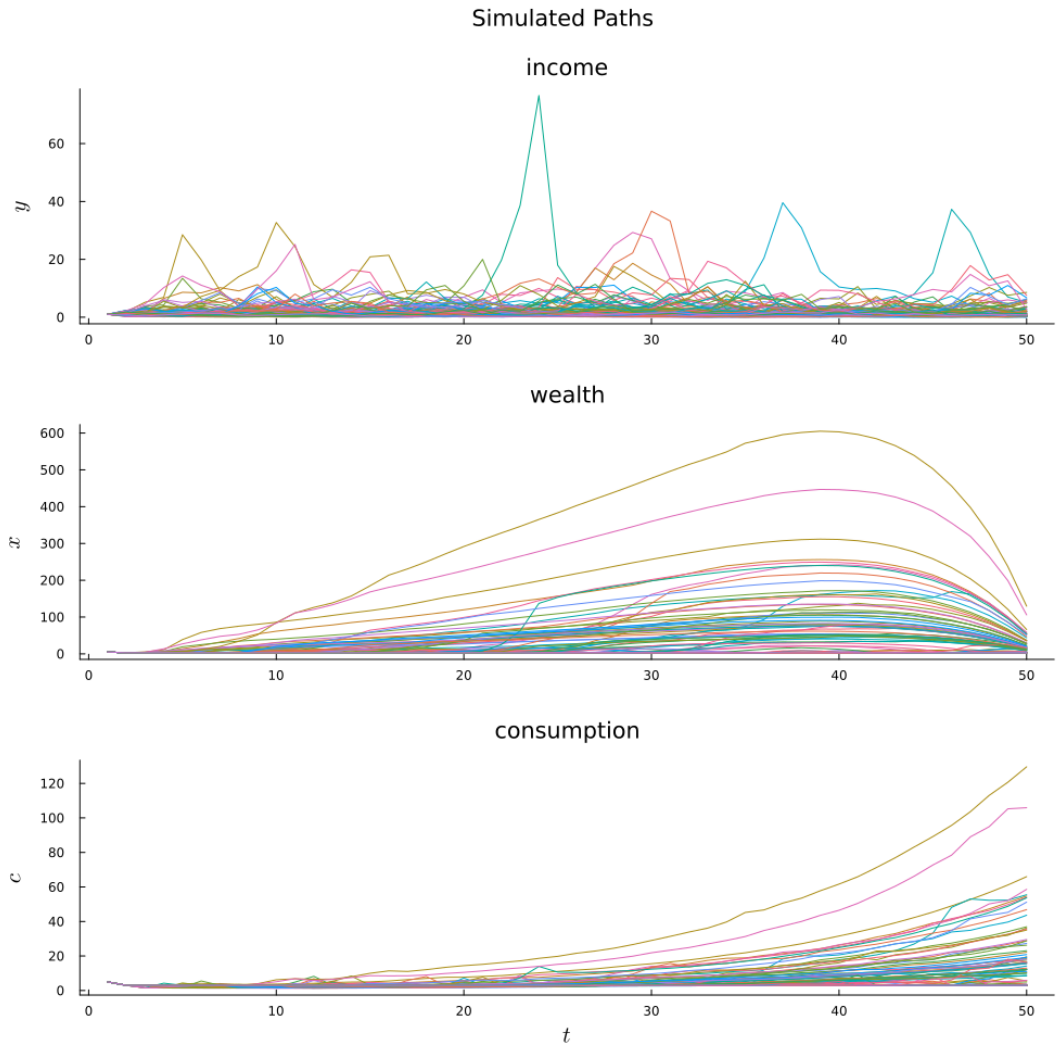

Figure 9: Policy functions and simulation results for AR1 model (PTD1)

The simulation results generated by PTD algorithm is shown in Figure 9. As it shows, most of the people get rid of the poverty trap since the positive income shock is persistent, and they start saving to smooth the consumption as described by the permanent income hypothesis (Hall, 1978).

Now we examine the simulation results generated by EGM algorithm. Figure 10 shows the simulated consumption-saving profiles generated by EGM1, i.e., the end-of-period asset grid is $range(0.0, 100, 200)$. Notice that the paths are drastically different from those generated by PTD algorithm: the wealth levels in life cycle are capped by 160 in Figure 10, while the richest agent in Figure 9 holds wealth more than 600; there are many spurious heavy fluctuations in the wealth levels; and there are many spurious spikes in the consumption paths. These features indicate large interpolation-extrapolation errors in EGM since the upper bound of end-of-period asset grid is too small.

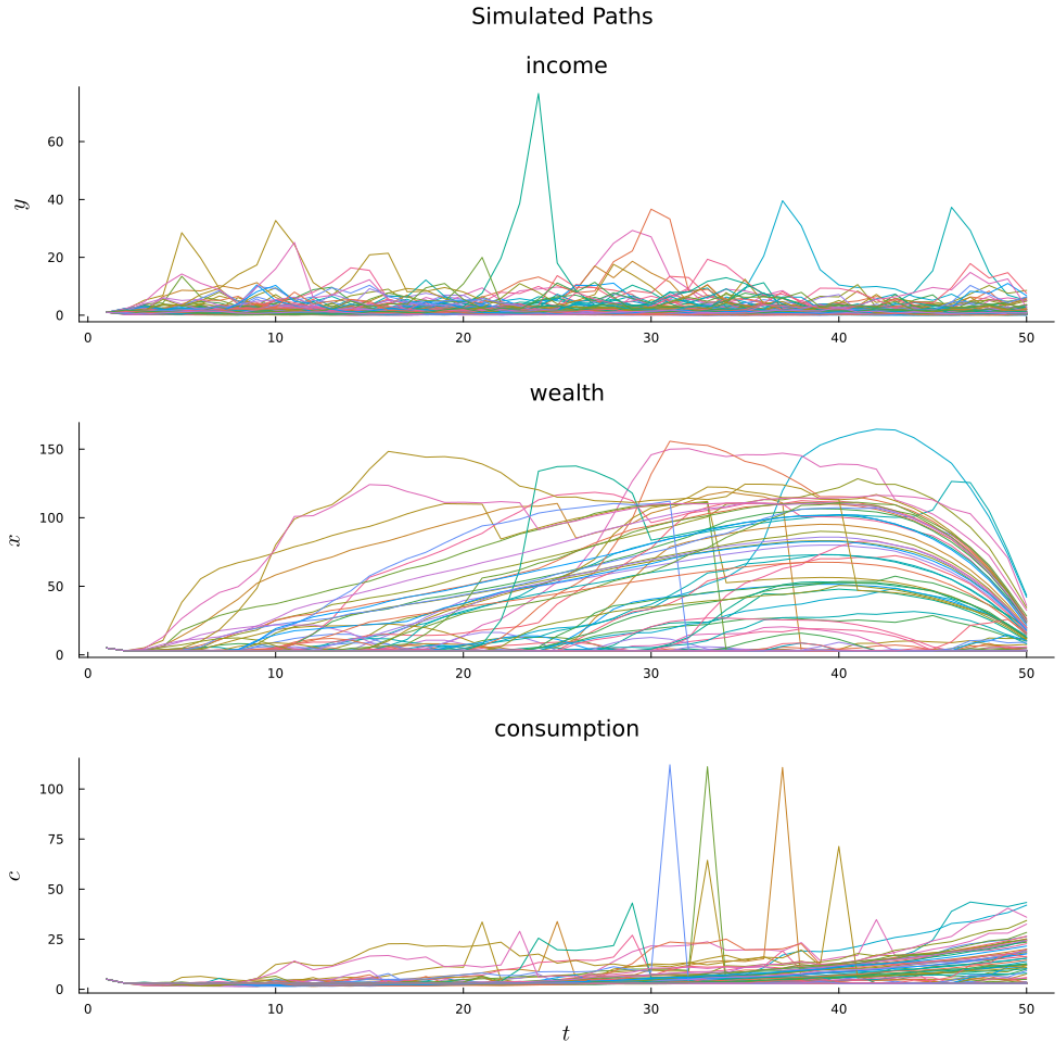

Figure 10: Policy functions and simulation results for AR1 model (EGM1)

If we expand the end-of-period asset grid to $range(0.0, 200, 400)$, there are still some spurious huge fluctuations in wealth paths and some spurious spikes in consumption paths as shown in Figure 11. These features indicate that the interpolation-extrapolation errors are still very large though we have expanded the end-of-period asset grid.

If we expand the end-of-period asset grid to $range(0.0, 600, 1200)$, the simulation results are almost the same as those generated by PTD algorithm (Figure 12 is almost identical to Figure 9), but EGM in this case is 6 times slower than PTD algorithm.

To conclude our discussion for Example 2, the out-of-box PTD algorithm is faster, more accurate and robust than EGM. The EGM algorithm can generate accurate results if we have a correct specified end-of-period asset grid, but which can only be obtained by trails and errors.

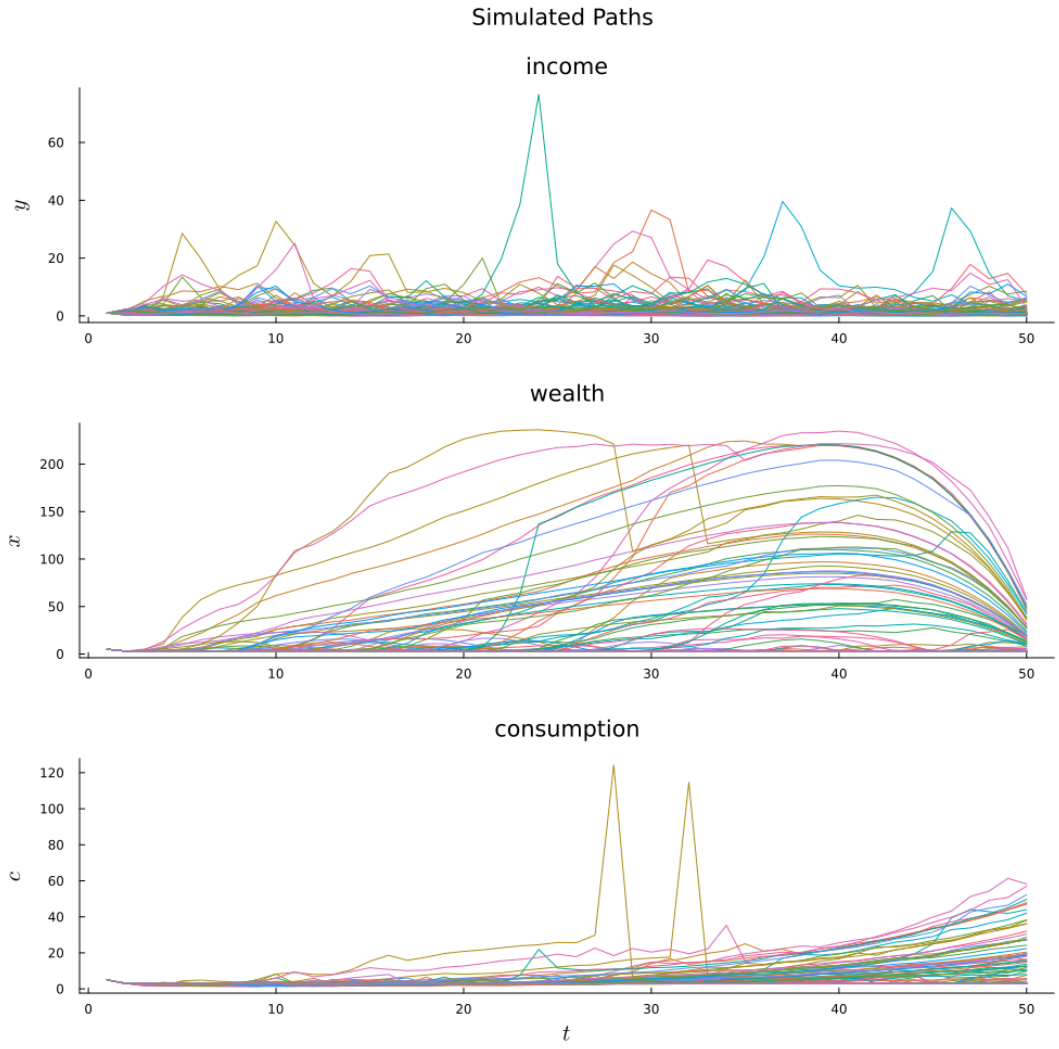

Figure 11: Policy functions and simulation results for AR1 model (EGM2)

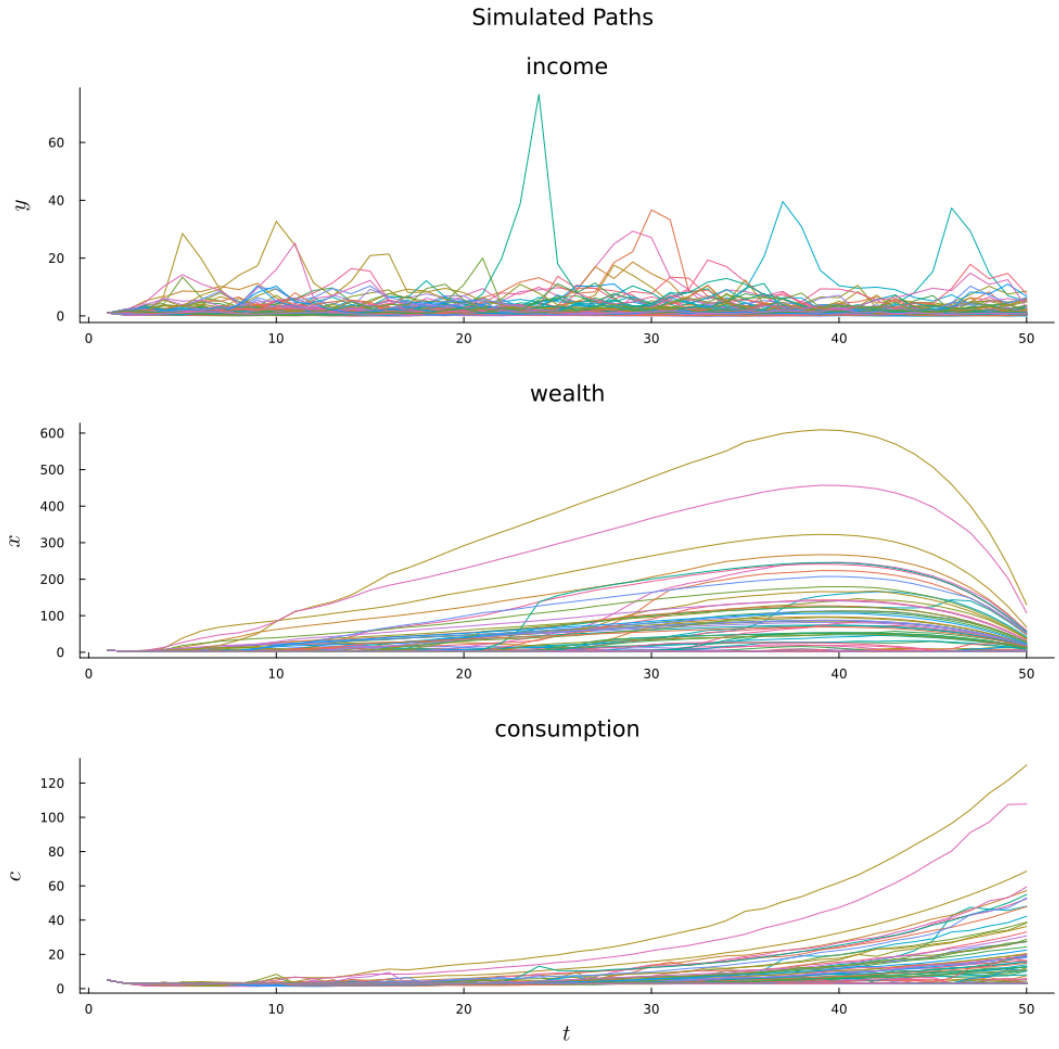

Figure 12: Policy functions and simulation results for AR1 model (EGM3)

Example 3 (HSZ). This example is intimidating for EGM, and it needs more than one hour to get a roughly accurate simulation for 100 agents. But it can still be simulated by PTD algorithm in less than one minute. We set the autoregressive coefficient in log earning process $\rho_e = 0.9$, the autoregressive coefficient in the medical expense process $\rho_m = 0.95$, the white noise ε_t^e in log earning process has a normal distribution $\mathcal{N}(0, 0.5)$ truncated on $(-\infty, 3.0]$, the white noise ε_t^m in log earning process has a normal distribution $\mathcal{N}(0, 0.3)$ truncated on $(-\infty, 3.0]$.

Figure 13 shows the simulated consumption-saving paths generated by PTD algorithm for 100 agents starting from the wealth $x_1 = 5.0$, earnings $e_1 = 1.0$ and medical expense $m_1 = 5.0$. Notice that unlike the AR1 model, most of the agents are trapped by the consumption floor. This is because the self-persistent medical expense shifts the net income process downwards, for people with low wealth and earning levels, they expect the medical expense will make their savings useless for smoothing the consumption, thus, they choose to consume all the wealth and

rely on social insurance in the next period.

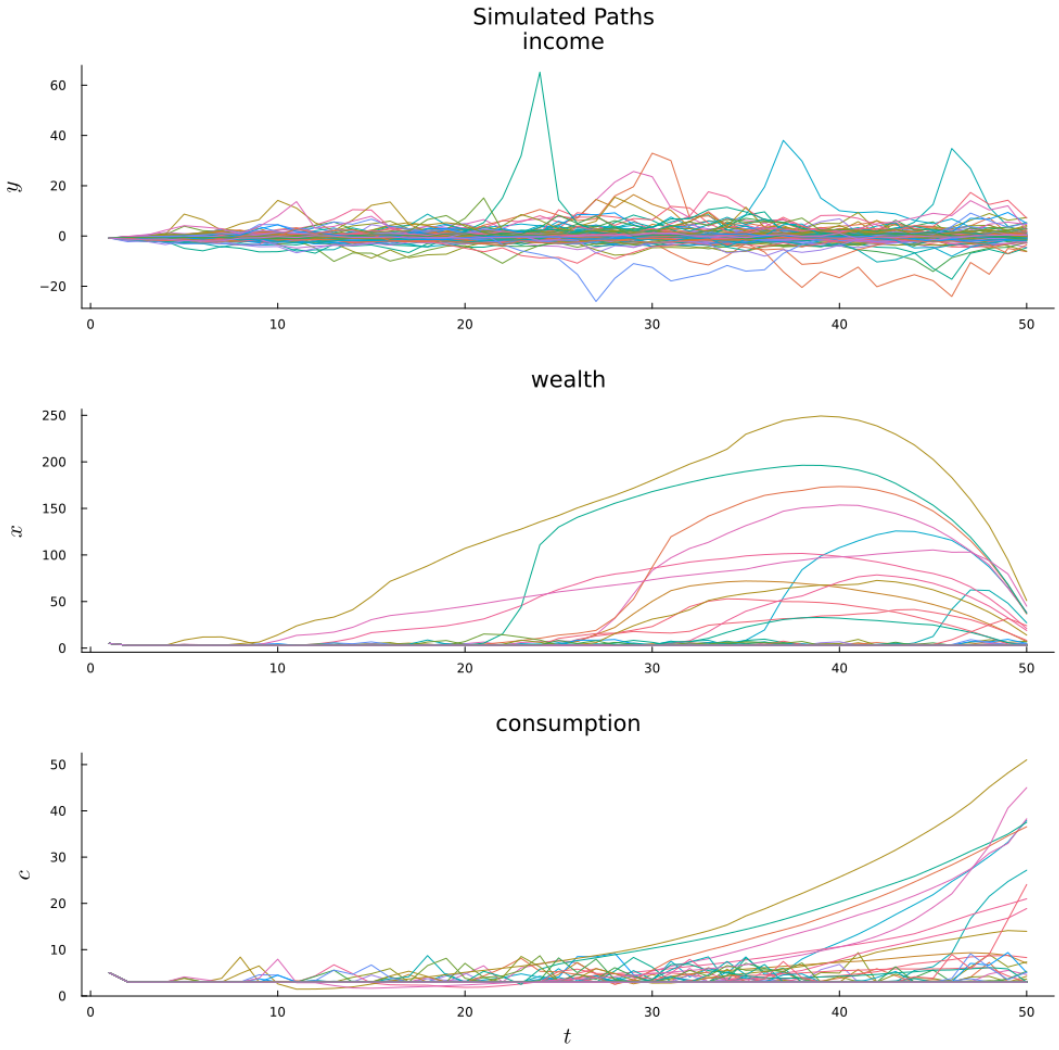

Figure 13: Policy functions and simulation results for HSZ model (PTD)

Notes: $n_{mc} = 100$, 53.341s is used to simulate 100 agents' paths.

5 PTD Recipe for Life-Cycle Models with Non-Convexity

Now we describe the class of the life-cycle model our algorithm can solve. We still denote x_t as wealth at time t, c_t as consumption at time t, a_t as end-of-period asset level, $\boldsymbol{s}_t$ as the vector of exogenous states, d_t as a discretized state variable that has only finite possible values, i.e., $d_t \in D_t \subset D$, $\#D < \infty$. Notice that D_{t+1} can be a function of d_t, for example, agents cannot work after retirement in the model discussed in Iskhakov et al. (2017). We also denote the IID shocks as $\boldsymbol{\varepsilon}_t$. Consider the following Bellman equation of finite horizon DP problems with a consumption-saving choice:

$$
\begin{aligned}
\omega_t(x_t, d_t, \boldsymbol{s}_t) = & \max_{c_t, d_{t+1}} \Big\{ u(c_t, d_t) + \beta \mathbb{E}_{\boldsymbol{\varepsilon}_{t+1}} \Big(\omega_{t+1}\big(x_{t+1}, d_{t+1}, \boldsymbol{s}_{t+1}\big) \mid \boldsymbol{s}_t \Big) \Big\}, \\
\text{s.t.} \quad & a_t = x_t - c_t, \\
& a_t \geq 0, \\
& d_{t+1} \subset D_{t+1}(d_t), \\
& \boldsymbol{s}_{t+1} = \gamma(\boldsymbol{s}_t, \boldsymbol{\varepsilon}_{t+1}), \\
& x_{t+1} = f(a_t, \boldsymbol{s}_{t+1}, d_{t+1}), \\
\omega_T(x_T, d_T, \boldsymbol{s}_T) = & g(x_T, s_T),
\end{aligned}
\tag{13}
$$

where $t \in 1, \cdots, T-1$. Notice that d_t is whether using social insurance or not at t in this paper, and whether retiring or not at t in Iskhakov et al. (2017).

The deterministic DP counterpart of (13) is given by

$$
\begin{aligned}
v_t(x_t, d_t) = & \max_{c_t, d_{t+1}} \Big\{ u(c_t, d_t) + \beta v_{t+1}(x_{t+1}, d_{t+1}) \Big\}, \\
\text{s.t.} \quad & a_t = x_t - c_t, \\
& a_t \geq 0, \\
& d_{t+1} \subset D_{t+1}(d_t), \\
& x_{t+1} = h(a_t, d_{t+1}), \\
\omega_T(x_T, d_T) = & g(x_T, d_T),
\end{aligned}
\tag{14}
$$

where $t \in 1, \cdots, T-1$. Notice that the exogenous state vector $\boldsymbol{s}_t$ has been removed from the Bellman equation since $\{\boldsymbol{\varepsilon}_t\}_{t=1}^T$ has been fixed.

We impose the following assumptions as the structure of the DP problems suitable for the PTD algorithm:

Assumption 1. $g(x_T, d_T)$ is an affine function of x_T for each d_T.

Assumption 2. The Euler equation for (14) is linear in c_t for each d_t.

Assumption 3. The optimal discrete choice function $d_{t+1}^\star$ satisfies the "no switching back property", i.e., for any $x' > x$, if $d_{t+1}^\star(x') \neq d_{t+1}^\star(x)$, then $d_{t+1}^\star(x'') \neq d_{t+1}^\star(x)$ for all $x'' > x'$.

The recipe for simulating the DP problems (13) from the initial state $(x_1, d_1, \boldsymbol{s}_1)$ by PTD algorithm is as follows[13]:

[13] This part is not fully complete, we will update the recipe and provide more numerical examples soon.

PTD Recipe for Life-Cycle Models with Non-Convexity

1. From the starting state (x_1, d_1, s_1), draw the sequences of exogenous states $\{\boldsymbol{\varepsilon}_t\}_{t=1}^T$.
2. For each sequence of the exogenous states, construct Policy Trees for the deterministic DP problem (14). Each policy segment at t corresponding one d_{t+1}.
3. Solve the deterministic problem along the policy tree using Algorithm 6.
4. Solve the optimal (c_1, d_2) using Algorithm 9, update x_2.
5. Repeat steps 1 - 4 to update the whole consumption-saving path.

6 Conclusion

In this paper, we developed a fast and accurate solution approach that targets non-convex dynamic optimization problems, but avoids discretization of not only the choice but also the continuous state variable. Our PTD algorithm builds the solver of the stochastic DP problem based on a fast and accurate solver of its counterpart deterministic DP problem. First, we build a solver of the deterministic model that relies on a tree structure of policy functions in the life cycle. Second, for the stochastic model, we use path simulation and Monte Carlo integration to approximate the expectation of the value function and establish the stochastic Euler equation. Finally, we solve the optimal policy accurately in real-time simulation based on time iteration.

We also show numerical examples of solving the consumption-saving problem with social insurance by the PTD algorithm. As the numerical exercises show, the PTD algorithm can solve this consumption-saving problem with non-convexity both faster and more accurately than the EGM algorithm.

Finally, we proposed a recipe for solving a class of life-cycle models with non-convexity by PTD algorithm. We will complete the recipe and the appendices in the next version of the paper.

Appendix A: Proofs of the Theorems

Proof of Theorem 1. Let $c_{t-1}^{k+1} = M(c_t^k)$. If $\gamma = 1$ in the utility function, (6) becomes $\frac{1}{c_{t-1}^{k+1}(x_{t-1})} = \beta(1+r)\frac{1}{c_t^k(z_t)}$; If $\gamma \neq 1$ in the utility function, (6) becomes $c_{t-1}^{k+1}(x_{t-1})^{-\gamma} = \beta(1+r)c_t^k(z_t)^{-\gamma}$. In both cases, $c_{t-1}^{k+1}(x_{t-1})$ is a linear function of $c_t^k(z_t)$, we also notice that $z_t = (1+r)(x_{t-1} - c_{t-1}^{k+1}(x_{t-1})) + y_t$ is an affine function of $c_t^{k+1}(x_{t-1})$ and x_{t-1}. Since $c_t^k(z_t)$ is an affine function of z_t, we have a linear equation of $c_{t-1}^{k+1}(x_{t-1})$ and x_{t-1}. Thus, $c_{t-1}^{k+1}(x_{t-1})$ is an affine function in x_{t-1}. □

Proof of Theorem 2. If $c_{t-1}^\star(x) = x$, the theorem holds trivially.

If $c_{t-1}^\star(x) < x$, the agent will have a positive saving at the end of $t-1$, thus, she will not rely on social insurance at t, otherwise she will not save at $t-1$ at all. We denote j as the first period she will consume all after t given the wealth level x at $t-1$, thus, the agent will never use social insurance before $j+1$ by the same argument.

From $t-1$ to j, the DP problem can be transformed into the following convex optimization problem:

$$\max\{\sum_{i=t-1}^{j} \beta^{i-t+1}u(c_i)\}, \tag{A.1}$$

$$\text{s.t.} \sum_{i=t-1}^{j}(1+r)^{j-i}c_i = (1+r)^{j-t+1}x + \sum_{i=t}^{j}(1+r)^{j-i}y_i. \tag{A.2}$$

The first two F.O.C.s are:

$$u'(c_{t-1}) = \lambda(1+r)^{j-t+1}$$
$$\beta u'(c_t) = \lambda(1+r)^{j-t},$$

where λ is the Lagrange multiplier.

Combing the two F.O.C.s, we have $u'(c_{t-1}) = \beta(1+r)u'(c_t)$. By the definition of policy collection at t, we know there is a $c_t^{*k}(z_t(x)) = c_t$. Thus, $c_{t-1}^{k+1}(x) = c_{t-1}$, where c_{t-1}^{k+1} is the policy child of c_t^{*k}. □

Remark. In the proof of Theorem 2, we haven't used any property of value function such as the differentiability of v_t at interior local maximizers, which requires finding the lower support function of v_t as described in Clausen and Strub (2020). The data structure in our algorithm simplifies the proof.

Proof of Theorem 3. If $t = T$, the theorem holds since $a_T^\star(x) = 0$.

For $t < T$, assume the theorem doesn't hold. Then there exists $x > x'$ but $a_t^\star(x) < a_t^\star(x')$, we denote $a = a_t^\star(x),\ \ a' = a_t^\star(x'),\ c = c_t^\star(x),\ c' = c_t^\star(x')$. Since $x > x'$ and $a < a'$, we have $c > c'$.

If $a = 0$, then by the definition of the value function, we have $c = x$:

$$v_t(x) = u(x) + \beta v_{t+1}(\underline{x}) \geq u\big(c' + (x - x')\big) + \beta v_{t+1}\big((1+r)a' + y_{t+1}\big), \tag{A.3}$$

$$v_t(x') = u(c') + \beta v_{t+1}\big((1+r)a' + y_{t+1}\big) \geq u(x') + \beta v_{t+1}(\underline{x}). \tag{A.4}$$

Since $u(\cdot)$ is a strictly concave function, we have:

$$u\big(c' + (x - x')\big) - u(c') > u(x) - u(x'). \tag{A.5}$$

Add (A.4) and (A.5), we have

$$u\big(c' + (x - x')\big) + \beta v_{t+1}\big((1+r)a' + y_{t+1}\big) > u(x) + \beta v_{t+1}(\underline{x}).$$

But this contradicts with (A.3), thus, $0 = a < a'$ is impossible.

If $a > 0$, notice that $c - c' > a' - a$ since $x > x'$, then by the definition of the value function, we have

$$v_t(x) = u(c) + \beta v_{t+1}\big((1+r)a + y_{t+1}\big) \geq u\big(c - (a' - a)\big) + \beta v_{t+1}\big((1+r)a' + y_{t+1}\big), \tag{A.6}$$

$$v_t(x') = u(c') + \beta v_{t+1}\big((1+r)a' + y_{t+1}\big) \geq u(c' + (a' - a)) + \beta v_{t+1}\big((1+r)a + y_{t+1}\big). \tag{A.7}$$

Since $u(\cdot)$ is a strictly concave function and $c > c' + (a' - a)$, we have:

$$u(c) - u(c - (a' - a)) < u(c' + (a' - a)) - u(c'). \tag{A.8}$$

But if we add (A.6) and (A.7), we have:

$$u(c) + u(c') \geq u(c - (a' - a)) + u(c' + (a' - a)).$$

But this contradicts with (A.8), thus, $0 < a < a'$ is impossible.

Combing the two cases, we proved that the assumption $a < a'$ doesn't hold, i.e., the theorem holds. □

Proof of Corollary 3.1. This is the direct implication of Theorem 3 and Definition 4. Since $x' > x$, then $z_t\big(a_t^\star(x')\big) > z_t\big(a_t^\star(x)\big) \in \text{dom}(c_t^{\star k})$, i.e., $z_t\big(a_t^\star(x')\big) \notin \text{dom}(c_t^{\star j})$ since $a_t^\star(x') > a_t^\star(x)$. Thus, $c_{t-1}^\star(x') \neq c_{t-1}^{j+1}(x')$ or $x' \notin \text{dom}(c_{t-1}^{j+1})$. □

Porof of Lemma 1. We prove this lemma by induction. The lemma holds for T since $\omega_T(x_T, \boldsymbol{s}_T) = v_T(x_T)$. Now assume the lemma holds for $t+2$, for period $t+1$, we have:

$$
\begin{aligned}
&\int \omega_{t+1}\Big(x_{t+1}(\boldsymbol{\varepsilon}_{t+1}), \boldsymbol{s}_{t+1}\Big)\mu(d\boldsymbol{\varepsilon}_{t+1}) \\
&= \int \max_{c_{t+1}} \Big\{u(c_{t+1}) + \beta \int \omega_{t+2}\Big(x_{t+2}(c_{t+1}, \boldsymbol{\varepsilon}_{t+2}), \boldsymbol{s}_{t+2}\Big)\mu(d\boldsymbol{\varepsilon}_{t+2})\Big\}\mu(d\boldsymbol{\varepsilon}_{t+1}), \\
&= \int \max_{c_{t+1}} \Big\{u(c_{t+1}) + \beta \int v_{t+2}\Big(x_{t+2}(c_{t+1}, \boldsymbol{\varepsilon}_{t+2}), \boldsymbol{y}_{t+2:T}\Big)\mu(d\boldsymbol{\varepsilon}_{t+2:T})\Big\}\mu(d\boldsymbol{\varepsilon}_{t+1}), \\
&= \int \max_{c_{t+1}} \Big(\int \Big\{u(c_{t+1}) + \beta v_{t+2}\Big(x_{t+2}(c_{t+1}, \boldsymbol{\varepsilon}_{t+2}), \boldsymbol{y}_{t+2:T}\Big)\Big\}\mu(d\boldsymbol{\varepsilon}_{t+2:T})\Big)\mu(d\boldsymbol{\varepsilon}_{t+1}), \\
&= \int \Big(\int_{\boldsymbol{\varepsilon}_{t+1}} \max_{c_{t+1}} \Big\{u(c_{t+1}) + \beta v_{t+2}\Big(x_{t+2}(c_{t+1}, \boldsymbol{\varepsilon}_{t+2}), \boldsymbol{y}_{t+2:T}\Big)\Big\}\mu(d\boldsymbol{\varepsilon}_{t+1:T})\Big)\mu(d\boldsymbol{\varepsilon}_{t+1}), \\
&= \int v_{t+1}\Big(x_{t+1}(\boldsymbol{\varepsilon}_{t+1}), \boldsymbol{y}_{t+1:T}\Big)\mu(d\boldsymbol{\varepsilon}_{t+1:T}),
\end{aligned}
$$

where the last but one equality holds since (i) we can change the underlying measures in the derivation since $\{\boldsymbol{\varepsilon}_t\}_{t=1}^T$ is IID process; (ii)the maximizer c_{t+1} is a a measurable function of $\boldsymbol{\varepsilon}_{t+1:T}$ so we can exchange the max operator and the integral. The last equality holds by the law of iterated expectation. □

Proof of Lemma 2. If $t = T$, the lemma holds trivially since $v_T(x_T) = u(x_T)$. If $t < T$, since $c_t^\star(x; \mathbf{y}_{t:T}) < x$, we can construct the Lagrangian of the standard convex optimization problem by changing $t-1$ to t in (A.1) and (A.2), and the envelope theorem shows:

$$
\frac{\partial v_t(x;\ \mathbf{y}_{t:T})}{\partial x} = \lambda(1+r)^{j-t}.
$$

Moreover, the F.O.C. shows that $u'(c_t) = \lambda(1+r)^{j-t}$. Thus, the lemma holds. □

Appendix B: DC-EGM Algorithm for Solving Consumption-Savings Problems with Safety Net

In this section we provide a brief description of how to use the DC-EGM (Iskhakov et al., 2017) algorithm to solve the consumption-savings problem with safety net. We refer readers to Iskhakov et al. (2017) for a detailed introduction to the DC-EGM algorithm.

Appendix C: Using PTD-I Algorithm for Approximating Policy Functions

References

Carroll, C. D. (2006): "The method of endogenous gridpoints for solving dynamic stochastic optimization problems," *Economics Letters*, 91 (3), 312–320.

Clausen, A., and C. Strub (2020): "Reverse calculus and nested optimization," *Journal of Economic Theory*, 187, 105019.

Coleman, W. J. (1990): "Solving the stochastic growth model by policy-function iteration," *Journal of Business & Economic Statistics*, 8 (1), 27–29.

De Nardi, M., E. French, and J. B. Jones (2010): "Why do the elderly save? The role of medical expenses," *Journal of Political Economy*, 118 (1), 39–75.

De Nardi, M., E. French, and J. B. Jones (2009): "Life expectancy and old age savings," *American Economic Review, Papers and Proceedings*, 99 (2), 110–115.

Dobrescu, L., and A. Shanker (2022): "Fast upper-envelope scan for discrete-continuous dynamic programming," *Working Paper*.

Druedahl, J. (2021): "A guide on solving non-convex consumption-saving models," *Computational Economics*, 58 (3), 747–775.

Druedahl, J., and T. H. Jørgensen (2017): "A general endogenous grid method for multi-dimensional models with non-convexities and constraints," *Journal of Economic Dynamics and Control*, 74, 87–107.

Fan, X., A. Seshadri, and C. Taber (2024): "Estimation of a Life-Cycle Model with Human Capital, Labor Supply, and Retirement," *Journal of Political Economy*, 132 (1), 48–95.

Fella, G. (2014): "A generalized endogenous grid method for non-smooth and non-concave problems," *Review of Economic Dynamics*, 17 (2), 329–344.

Hall, R. E. (1978): "Stochastic implications of the life cycle-permanent income hypothesis: theory and evidence," *Journal of Political Economy*, 86 (6), 971–987.

Hubbard, R. G., J. Skinner, and S. P. Zeldes (1995): "Precautionary saving and social insurance," *Journal of Political Economy*, 103 (2), 360–399.

Iskhakov, F., T. H. Jørgensen, J. Rust, and B. Schjerning (2017): "The endogenous grid method for discrete-continuous dynamic choice models with (or without) taste shocks," *Quantitative Economics*, 8 (2), 317–365.

Krusell, P., and A. A. Smith, Jr (1998): "Income and wealth heterogeneity in the macroeconomy," *Journal of political Economy*, 106 (5), 867–896.

Li, H., and J. Stachurski (2014): "Solving the income fluctuation problem with unbounded rewards," *Journal of Economic Dynamics and Control*, 45, 353–365.

Li, W., H. Liu, F. Yang, and R. Yao (2016): "Housing over time and over the life cycle: a structural estimation," *International Economic Review*, 57 (4), 1237–1260.

Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery (2007): *Numerical Recipes: The Art of Scientific Computing*: Cambridge University Press.

Scholz, J. K., A. Seshadri, and S. Khitatrakun (2006): "Are Americans saving "optimally" for retirement?" *Journal of Political Economy*, 114 (4), 607–643.

Tang, Y. (2023): "A Note on Monte Carlo Integration in High Dimensions," *The American Statistician*, 1–7.