

Dynamic Programming Treatment of the Solow and Ramsey–Cass–Koopmans Growth Models

Ahmad Y. A. Thabaine

An-Najah National University

Adnan Daraghme

An-Najah National University

Alvaro H. Salas

ahsalass@unal.edu.co

Universidad Nacional de Colombia

Research Article

Keywords: Solow–Swan model, Ramsey–Cass–Koopmans model, dynamic programming, Bellman equation, value function iteration, fitted VFI, Howard policy iteration, optimal growth

Posted Date: October 9th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11108637/v1>

License:   This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Dynamic Programming Treatment of the Solow and Ramsey–Cass–Koopmans Growth Models

Ahmad Y. A. Thabaine^{*1}, Adnan Daraghme^{†1}, and Alvaro H. Salas^{‡2}

¹Department of Physics and Mathematics, An-Najah National University, Nablus, Palestine

²Department of Mathematics and Statistics, FIZMAKO Research Group, Universidad Nacional de Colombia, Sede Manizales, Manizales, Colombia

Abstract

The Solow–Swan and Ramsey–Cass–Koopmans models provide complementary descriptions of long-run growth: the former imposes an exogenous saving rate, whereas the latter determines saving from an intertemporal optimization problem. This paper develops a self-contained bridge from these growth models to the Bellman equation and then uses the Cobb–Douglas/log-utility benchmark, whose infinite-horizon value function is available in closed form, to study fitted value function iteration (VFI). The numerical focus is the effect of the initial value-function guess on computational effort. For $A = 1$, $\alpha = 0.65$, $\beta = 0.95$, and a 150-point capital grid, the initial guesses $w_A(k) = 5 \log k - 25$ and $w_B(k) = 2k^{0.25} - 35$ require 37 and 28 Bellman sweeps, respectively, to reach an interior sup-norm error below one, a reduction of about 24%. Both sequences exhibit essentially the same geometric decay rate, close to the theoretical discount factor, so the advantage of w_B is primarily a starting-distance/transient effect rather than a different asymptotic rate. We also separate iteration error from interpolation error through grid refinement, show that a coarse grid produces a large boundary discretization error near the logarithmic singularity, compare value- and policy-function convergence, and benchmark initialization gains against partial Howard policy iteration. The accompanying deterministic code reproduces every numerical table and figure.

Keywords: Solow–Swan model, Ramsey–Cass–Koopmans model, dynamic programming, Bellman equation, value function iteration, fitted VFI, Howard policy iteration, optimal growth

1 Introduction

The Solow–Swan model [Solow, 1956, Swan, 1956] and the Ramsey–Cass–Koopmans model [Ramsey, 1928, Cass, 1965, Koopmans, 1965] are canonical frameworks for long-run growth. Their central difference is the treatment of saving. In the Solow model the saving rate is exogenous, whereas in the Ramsey model consumption and saving arise from an intertemporal optimization problem. This distinction makes the Ramsey framework a natural entry point to recursive methods and the Bellman equation [Stokey et al., 1989].

The paper has two purposes. The first is expository: Sections 2–6 give a self-contained derivation of the Solow and Ramsey models, the discrete-time Bellman formulation, the contraction principle

^{*}E-mail: ahmad.thabaine@najah.edu

[†]E-mail: adn.daraghme@najah.edu

[‡]E-mail: ahsalass@unal.edu.co

under standard bounded-return assumptions, qualitative shape results for the Bellman operator, and the closed-form Cobb–Douglas/log-utility value function. The second is computational: Sections 7–9 use that closed form as an exact benchmark for fitted VFI and quantify how initialization, grid resolution, and algorithmic acceleration affect computational effort.

Our numerical question is deliberately narrow. Contraction theory explains why admissible starting values converge to the same fixed point under appropriate hypotheses, but it does not make the number of Bellman sweeps independent of the starting guess. Using the deterministic benchmark, we measure the error to the analytic value function at every iteration, compare two initial guesses used in earlier work [Thabaineh, 2014], extend the comparison to a broader family of initializations, and contrast the resulting gains with partial Howard policy iteration [Howard, 1960]. We also distinguish value-function convergence from policy-function convergence and separate finite-iteration error from the discretization floor induced by piecewise-linear interpolation.

The paper does not propose a new dynamic-programming algorithm. Its contribution is a reproducible numerical analysis of an often-overlooked implementation choice—the initial value-function guess—together with a careful statement of the assumptions under which the surrounding contraction arguments are valid. This distinction is important for log utility: the exact value function is unbounded below as $k \downarrow 0$, and on a coarse grid the first interpolation interval generates a large boundary discretization error. We therefore state explicitly which error norm and state-space region are used in each comparison.

The organization is as follows. Section 2 presents the Solow model, Section 3 the Ramsey–Cass–Koopmans model, and Section 4 their main conceptual differences. Section 5 develops the discrete-time Bellman formulation and supporting dynamic-programming results. Section 6 derives the closed-form Cobb–Douglas/log-utility benchmark. Section 7 describes fitted VFI, Section 8 reports the numerical experiments, Section 9 discusses implications and limitations, and Section 10 concludes. Appendix A records the algebra for the closed-form coefficients, while Appendix B provides the complete reproducibility code used for the numerical results.

2 The Solow Growth Model

2.1 Accounting identities and exogenous saving

The Solow–Swan model [Solow, 1956, Swan, 1956] describes a closed economy, with no international trade and no government, producing a single good using capital and labor. Time is discrete and indexed by t . Five variables are endogenous and time-varying:

- Y_t : output (income);
- K_t : capital;
- L_t : labor;
- I_t : investment;
- C_t : consumption.

Two parameters are exogenous and constant:

- $s \in (0, 1)$: the saving rate;
- $\delta \in (0, 1)$: the depreciation rate of capital.

The defining feature of the Solow model — the feature that the Ramsey model in Section 3 will relax — is that s is simply posited, not derived from any optimization problem. In the good market, output is either consumed or invested,

$$Y_t = C_t + I_t, \tag{1}$$

with $I_t = sY_t$ and $C_t = (1 - s)Y_t$ by definition of the saving rate.

Labor grows at a constant exogenous rate $n \geq 0$: if $L_0 = 1$,

$$L_t = (1 + n)^t. \quad (2)$$

Existing capital depreciates at the fixed rate $\delta \in [0, 1]$ each period, so that the capital stock at the start of next period equals the undepreciated portion of this period's capital plus this period's investment:

$$K_{t+1} = (1 - \delta)K_t + I_t. \quad (3)$$

2.2 The production function and its intensive form

Output is produced from capital, labor, and the “effectiveness of labor” A_t (also called labor-augmenting technology) via

$$Y_t = F(K_t, A_t L_t), \quad (4)$$

where A_t grows exogenously at rate g and is treated as a free, non-excludable, non-rival input available to all firms. The model's central technical assumption on F is *constant returns to scale* in capital and effective labor:

$$F(cK, cAL) = cF(K, AL) \quad \forall c \geq 0. \quad (5)$$

Constant returns lets us work in *intensive form*. Setting $c = 1/(AL)$ in (5) and defining $k = K/(AL)$ (capital per unit of effective labor) and $y = Y/(AL)$ (output per unit of effective labor) gives

$$y = f(k), \quad f(k) \equiv F(k, 1). \quad (6)$$

We impose the standard neoclassical assumptions: positive but diminishing marginal products, essential inputs, and Inada-type boundary behavior. In particular, $F_K > 0$, $F_{KK} < 0$, $\lim_{K \downarrow 0} F_K = \infty$, and $\lim_{K \rightarrow \infty} F_K = 0$, with analogous conditions for effective labor.

The workhorse example, used throughout both the Solow and Ramsey analyses below, is the **Cobb–Douglas production function**,

$$F(K, AL) = K^\alpha (AL)^{1-\alpha}, \quad 0 < \alpha < 1. \quad (7)$$

It is straightforward to verify constant returns: $F(cK, cAL) = cF(K, AL)$. Dividing by effective labor gives the intensive production function

$$y_t = k_t^\alpha, \quad k_t = \frac{K_t}{A_t L_t}, \quad y_t = \frac{Y_t}{A_t L_t}. \quad (8)$$

The discrete-time numerical benchmark introduced later allows an independent productivity scale $A > 0$ and writes $f(k) = Ak^\alpha$. This scalar A should not be confused with the time-varying labor-augmenting technology A_t used in the continuous-time growth model.

2.3 The capital accumulation equation

Combining $I_t = sY_t$ with (3) gives the **capital accumulation equation**,

$$K_{t+1} = (1 - \delta)K_t + sY_t \iff \Delta K_t \equiv K_{t+1} - K_t = sY_t - \delta K_t. \quad (9)$$

In continuous time, with a dot denoting the time derivative, the analogous law of motion for capital per unit of effective labor is

$$\dot{k}(t) = sf(k(t)) - \delta k(t) - (n + g)k(t), \quad (10)$$

where the additional $-(n + g)k$ term captures the “dilution” of capital per effective worker caused by growth in the effective labor force; the sum $n + g + \delta$ is assumed positive.

2.4 The steady state

A balanced-growth steady state is most naturally defined in effective-labor units by $\dot{k} = 0$. With the Cobb–Douglas intensive technology $f(k) = k^\alpha$, equation (10) gives

$$0 = s(k^*)^\alpha - (n + g + \delta)k^* \implies k^* = \left(\frac{s}{n + g + \delta} \right)^{\frac{1}{1-\alpha}}. \quad (11)$$

If labor and technology are instead held fixed and output is written directly in levels as $Y = AK^\alpha$, the corresponding special case is $K^* = (sA/\delta)^{1/(1-\alpha)}$. The principal qualitative conclusion of the Solow model is that the accumulation of physical capital alone, with a fixed saving rate, cannot by itself account for sustained long-run growth in output per person; along the balanced growth path, output per worker grows only at the exogenous rate of technological progress g . The saving rate s affects the *level* of steady-state capital per effective worker (equation (11)) but not its long-run *growth rate*. This is precisely the limitation that motivates asking whether the saving rate should instead be *derived* from optimizing behavior — the question answered by the Ramsey–Cass–Koopmans model.

3 The Ramsey–Cass–Koopmans Model

The Ramsey model [Ramsey, 1928], later developed by Cass [1965] and Koopmans [1965], replaces the Solow model’s exogenous saving rate with the solution to an explicit dynamic optimization problem: a representative, infinitely-lived household selects a consumption path to maximize discounted lifetime utility. This decision is made within a budget set by consumption, labor income, and return on capital.

3.1 Firms

There are many identical competitive firms, each of which has access to the constant-returns production function $Y = F(K, AL)$ of Section 2.2, hence the same assumptions as in the Solow model. Firms hire labor and rent capital in competitive factor markets, taking the labor-effectiveness process A_t (growing at exogenous rate g) as given, and earn zero profit in equilibrium because F has constant returns to scale and markets are competitive. Capital depreciates at the same rate $\delta \in [0, 1)$, as in the Solow model; competitive renting means the household’s net return is the marginal product of capital net of depreciation, so that, using the intensive-form production function $f(k) = F(k, 1)$, at time t , the real interest rate is

$$r(t) = f'(k_t) - \delta, \quad (12)$$

and the wage per unit of effective labor (unaffected by δ , since depreciation falls on the capital return, not the labor return) is

$$w(t) = f(k(t)) - k(t)f'(k(t)). \quad (13)$$

We shall use δ as a free parameter for all of this section to be consistent with the continuous-time Solow model above (Section 2.3–2.4) and with our Hamiltonian derivation of Section 3.5; the special case $\delta = 0$ recovers the simpler “no depreciation” formulation used in some textbooks and is not assumed here.

3.2 Households and preferences

There are many identical households, each of size growing at rate n , and each member has to provide one unit of labor at each point in time, and the household is renting its capital assets to firms. The lifetime utility functional of a household is

$$U = \int_{t=0}^{\infty} e^{-\rho t} u(C(t)) \frac{L(t)}{H} dt, \quad (14)$$

where $C(t)$ is consumption per member of the household, $L(t)/H$ is the number of members of the household (and H is the number of households), and ρ is the pure rate of time preference. For $\theta \neq 1$ we use the constant-relative-risk-aversion (CRRA) representation

$$u(C(t)) = \frac{C(t)^{1-\theta}}{1-\theta}, \quad \theta > 0, \quad \rho - n - (1-\theta)g > 0, \quad (15)$$

and for $\theta = 1$ we use $u(C) = \log C$. The name CRRA is because the coefficient of relative risk aversion, $-Cu''(C)/u'(C)$, equals the constant θ independently of C . Let us differentiate twice,

$$u'(C) = C^{-\theta}, \quad u''(C) = -\theta C^{-\theta-1}, \quad -\frac{C u''(C)}{u'(C)} = \theta. \quad (16)$$

The representation in (15) differs from the continuous-at- $\theta = 1$ normalization $(C^{1-\theta} - 1)/(1 - \theta)$ only by an additive constant, so both generate the same choices. The latter has the pointwise limit $\log C$ as $\theta \rightarrow 1$. The condition $\rho - n - (1 - \theta)g > 0$ is a standard integrability restriction for the normalized objective; without a suitable restriction of this type, discounted lifetime utility may fail to be finite.

3.3 The household's budget constraint

Normalizing by effective labor (defining $c(t) = C(t)/A(t)$ as consumption per unit of effective labor, so that $C(t) = A(t)c(t)$, and similarly $w(t)$ as the wage per unit of effective labor), we can show that the household's instantaneous utility can be written as

$$\frac{C(t)^{1-\theta}}{1-\theta} = A(0)^{1-\theta} e^{(1-\theta)gt} \frac{c(t)^{1-\theta}}{1-\theta}, \quad (17)$$

using $A(t) = A(0)e^{gt}$. Substituting this and $L(t) = L(0)e^{nt}$ into the utility functional and collecting constants gives

$$U = B \int_{t=0}^{\infty} e^{-\varrho t} \frac{c(t)^{1-\theta}}{1-\theta} dt, \quad (18)$$

where $B = A(0)^{1-\theta} \frac{L(0)}{H}$, and $\varrho = \rho - n - (1-\theta)g > 0$. Thus, maximizing U is equivalent to maximizing the normalized objective with effective discount rate ϱ . The household's flow budget constraint, expressed in present-value form and normalized by effective labor (using $R(t) = \int_{\tau=0}^t r(\tau) d\tau$ for the cumulative discount factor implied by the sequence of interest rates), is

$$\int_{t=0}^{\infty} e^{-R(t)} c(t) e^{(n+g)t} dt \leq k(0) + \int_{t=0}^{\infty} e^{-R(t)} w(t) e^{(n+g)t} dt. \quad (19)$$

3.4 The household's optimization problem and the Euler equation

Forming the Lagrangian for the household's problem of maximizing (18) subject to (19), with multiplier λ ,

$$\mathcal{L} = B \int_0^\infty e^{-\varrho t} \frac{c(t)^{1-\theta}}{1-\theta} dt + \lambda \left[k(0) + \int_0^\infty e^{-R(t)} e^{(n+g)t} w(t) dt - \int_0^\infty e^{-R(t)} e^{(n+g)t} c(t) dt \right], \quad (20)$$

the first-order condition for consumption at each date t is

$$B e^{-\varrho t} c(t)^{-\theta} = \lambda e^{-R(t)} e^{(n+g)t}. \quad (21)$$

Taking logs of this,

$$\ln B - \varrho t - \theta \ln c(t) = \ln \lambda - R(t) + (n+g)t = \ln \lambda - \int_{\tau=0}^t r(\tau) d\tau + (n+g)t, \quad (22)$$

and differentiating both sides with respect to t (using $\dot{R}(t) = r(t)$) gives

$$-\varrho - \theta \frac{\dot{c}(t)}{c(t)} = -r(t) + (n+g) \implies \frac{\dot{c}(t)}{c(t)} = \frac{r(t) - n - g - \varrho}{\theta} = \frac{r(t) - \rho - \theta g}{\theta}, \quad (23)$$

where the last equality uses $\varrho = \rho - n - (1-\theta)g$. Equation (23) is the continuous-time **consumption Euler equation**: the growth rate of consumption per effective worker is proportional to the gap between the real interest rate and the sum of the discount rate and the utility-weighted growth rate of technology. When $r(t) > \rho + \theta g$, the household finds it optimal to postpone consumption (save more), and vice versa.

3.5 Hamiltonian derivation and the Ramsey system

The same Euler equation follows from the normalized current-value Hamiltonian

$$\mathcal{H} = u(c) + \lambda [f(k) - c - (n+g+\delta)k], \quad (24)$$

with effective discount rate $\varrho = \rho - n - (1-\theta)g$. The first-order condition is $\lambda = u'(c) = c^{-\theta}$, and the current-value costate equation is

$$\dot{\lambda} = \varrho \lambda - \frac{\partial \mathcal{H}}{\partial k} = \lambda [\varrho - f'(k) + n + g + \delta]. \quad (25)$$

Because $\dot{\lambda}/\lambda = -\theta \dot{c}/c$, we obtain

$$\frac{\dot{c}}{c} = \frac{f'(k) - \delta - \rho - \theta g}{\theta} = \frac{r - \rho - \theta g}{\theta}, \quad (26)$$

which is identical to (23).

For variables measured per unit of effective labor, the Ramsey system is therefore

$$\dot{k} = f(k) - c - (n+g+\delta)k, \quad (27)$$

$$\frac{\dot{c}}{c} = \frac{f'(k) - \delta - \rho - \theta g}{\theta}. \quad (28)$$

The first equation is the resource constraint and the second is the Euler equation. Under the usual neoclassical assumptions the phase diagram has a saddle path leading to the interior steady state. The remainder of the paper turns to a discrete-time benchmark in which the same intertemporal trade-off is expressed recursively.

4 Solow versus Ramsey: Main Differences

The two frameworks share the same basic production technology and capital-accounting logic, but they differ in how the saving/consumption split is determined. In the Solow model the saving rate s is exogenous and constant, whereas in the Ramsey model saving is endogenous and generally time-varying because it is implied by household optimization. The Solow model contains no optimizing decision maker and hence no preference or welfare criterion; the Ramsey model introduces preferences, an intertemporal planning horizon, and a welfare objective. Consequently, capital accumulation in the Solow model is mechanical conditional on s , while in the Ramsey model it is part of a competitive equilibrium generated jointly by household saving and firm factor demand.

Both models retain the central conclusion that sustained long-run growth in output per person requires exogenous technological progress in the baseline specification. The Ramsey model is nevertheless richer because it permits welfare comparisons and provides the recursive optimization problem that motivates the dynamic-programming analysis below.

5 Dynamic Programming Formulation

5.1 Setup and Bellman equation

We now turn from the continuous-time, Hamiltonian-based derivation of Section 3.5 to the discrete-time, Bellman-equation-based formulation of essentially the same optimal growth problem, which is the natural starting point for numerical solutions.

Let an agent (or social planner) have capital $k_t \in \mathbb{R}_+$ at each date t , produces output $f(k_t) \in \mathbb{R}_+$, and divides that output between consumption c_t and capital saved for the next period,

$$k_{t+1} = f(k_t) - c_t. \quad (29)$$

The planner chooses a nonnegative, feasible consumption sequence $\{c_t\}_{t=0}^{\infty}$ so that the implied capital path $\{k_t\}$ is nonnegative to maximize

$$\sum_{t=0}^{\infty} \beta^t u(c_t), \quad 0 < \beta < 1. \quad (30)$$

A remark on depreciation. Equation (29) implicitly sets the depreciation rate to $\delta = 1$ (full depreciation in each period): all the capital in a period is lost in the next period except what is saved from output. This is a genuine, deliberate simplification of the $\delta \in (0, 1)$ we have used in the continuous-time Solow and Ramsey sections above (eqs. (9), (27)) - in line with the usual convention in, e.g., Sargent and Stachurski [2013] - because it is in fact the case for which the Cobb-Douglas/log-utility problem is solved in closed form (Section 6), and therefore our model is an exact numerical benchmark in Sections 7 and 8. The discrete time law of motion is $k_{t+1} = f(k_t) + (1 - \delta)k_t - c_t$ and is achieved by taking $\delta = 1$: In the discrete time case, $\delta = 1$ and so all the dynamic programming theory in the rest of Section 5 (the Bellman equation, the contraction mapping theorem, the shape-preservation results) goes through unchanged for any fixed $\delta \in (0, 1]$ with $f(k_t)$ in (29) replaced by $f(k_t) + (1 - \delta)k_t$ in all cases. The reason is that we take $\delta = 1$ so that the closed form derivation in Section 6 is precisely correct and not because it is harder in general.

Under the usual continuity, feasibility, and compactness conditions, the infinite-horizon problem admits an optimal stationary Markov policy σ , so that $c_t = \sigma(k_t)$ and $k_{t+1} = f(k_t) - \sigma(k_t)$. The planner's problem can then be written recursively via the **Bellman equation**.

The core idea of dynamic programming is to collapse the infinite-horizon sequence problem into a sequence of two-period problems in the recursive form

$$\begin{aligned} V(k_t) &= \max_{c_t, k_{t+1}} \sum_{i=0}^{\infty} \beta^i u(c_{t+i}) \\ &= \max_{c_t, k_{t+1}} \left\{ u(c_t) + \beta \sum_{i=0}^{\infty} \beta^i u(c_{t+i+1}) \right\} \\ &= \max_{c_t, k_{t+1}} \{ u(c_t) + \beta V(k_{t+1}) \} \quad \text{s.t. } k_{t+1} = f(k_t) - c_t. \end{aligned} \quad (31)$$

Equation (31) is the Bellman equation. It is instructive to build up its finite-horizon analogues explicitly, both because this is how the recursive structure is most transparently verified, and because it clarifies exactly what the “infinite horizon” limit in Section 5.3 is a limit *of*.

5.2 Finite-horizon derivation

To make the algebra explicit we specialize, following the same functional forms used throughout the rest of the paper, to

$$f(k) = Ak^\alpha, \quad u(c) = \log(c). \quad (32)$$

Horizon $T = 0$. With zero periods remaining after the current one (lifetime utility after “death” is normalized to zero), the problem is

$$v_0(k_0) = \max_{k_1 \geq 0} \{ u(f(k_0) - k_1) \}. \quad (33)$$

Since u is strictly increasing, the optimum sets $k_1 = 0$ (consume everything) and $c_0 = Ak_0^\alpha$, so

$$v_0(k_0) = \log(Ak_0^\alpha) = \log A + \alpha \log(k_0). \quad (34)$$

Horizon $T = 1$. With one additional period,

$$v_1(k_0) = \max_{k_1, k_2} \left\{ \log(Ak_0^\alpha - k_1) + \beta \log(Ak_1^\alpha - k_2) \right\}, \quad (35)$$

subject to $k_2 \geq 0$; setting $k_2 = 0$, the first-order condition with respect to k_1 is

$$\frac{\partial v_1}{\partial k_1} = \frac{-1}{Ak_0^\alpha - k_1} + \frac{\beta \alpha Ak_1^{\alpha-1}}{Ak_1^\alpha} = 0 \implies \frac{1}{Ak_0^\alpha - k_1} = \frac{\beta \alpha}{k_1}, \quad (36)$$

which rearranges to $k_1 = \beta \alpha Ak_0^\alpha - \beta \alpha k_1$, i.e.

$$k_1 = \frac{\alpha \beta}{1 + \alpha \beta} Ak_0^\alpha. \quad (37)$$

Substituting (37) back into (35) and simplifying (using $\log(x^p) = p \log x$ repeatedly on the resulting expressions) gives

$$v_1(k_0) = \log\left(\frac{A}{1 + \alpha \beta}\right) + \alpha \log(k_0) + \beta \log\left(A \left(\frac{\alpha \beta}{1 + \alpha \beta}\right)^\alpha\right) + \beta \alpha^2 \log(k_0), \quad (38)$$

which can be reorganized, using (34), into the recursive form

$$v_1(k_0) = v_0(k_0) + \log\left(\frac{1}{1 + \alpha\beta}\right) + \alpha\beta \log\left(\frac{\alpha\beta}{1 + \alpha\beta}\right) + (1 + \alpha)\beta \log A + \beta\alpha^2 \log k_0, \quad (39)$$

which is, by construction, exactly the value delivered by one application of the Bellman operator to v_0 , so that

$$v_1(k_0) = \max_{k_1} \left\{ \log(Ak_0^\alpha - k_1) + \beta v_0(k_1) \right\}, \quad (40)$$

i.e. the two-period value function is obtained by applying the Bellman operator once to the zero-period value function, exactly as (31) asserts.

Horizon $T = 2$. With $k_3 = 0$, the problem

$$v_2(k_0) = \max_{k_1, k_2, k_3} \left\{ \log(Ak_0^\alpha - k_1) + \beta \log(Ak_1^\alpha - k_2) + \beta^2 \log(Ak_2^\alpha - k_3) \right\} \quad (41)$$

has first-order conditions for k_2 and k_1 that, after the same algebra as above, yield

$$k_2 = \frac{A\beta\alpha k_1^\alpha}{1 + \beta\alpha}, \quad k_1 = \frac{(\alpha\beta + (\alpha\beta)^2)}{(1 + \alpha\beta + (\alpha\beta)^2)} Ak_0^\alpha, \quad (42)$$

and substituting back gives $v_2(k_0)$ as a function of α, β, A, k_0 satisfying, exactly as in the $T = 1$ case,

$$v_2(k_0) = \max_{k_1} \left\{ \log(Ak_0^\alpha - k_1) + \beta v_1(k_1) \right\}. \quad (43)$$

General T . The pattern in (37) and (42) is the same; the optimal first-period capital choice for a horizon- T problem is

$$k_1 = \frac{\alpha\beta + (\alpha\beta)^2 + \dots + (\alpha\beta)^T}{1 + \alpha\beta + (\alpha\beta)^2 + \dots + (\alpha\beta)^T} Ak_0^\alpha. \quad (44)$$

The numerator and denominator are (shifted) finite geometric series; using $\sum_{t=0}^{T-1} r^t = (1 - r^T)/(1 - r)$ for $|r| < 1$, the formula becomes

$$k_1 = \alpha\beta \left[\frac{1 - (\alpha\beta)^T}{1 - (\alpha\beta)^{T+1}} \right] Ak_0^\alpha. \quad (45)$$

5.3 Infinite-horizon limit

Now let $T \rightarrow \infty$ in (45), and noting $0 < \alpha\beta < 1$ so that $(\alpha\beta)^T \rightarrow 0$, gives the optimal policy in the infinite horizon

$$k_{t+1} = \alpha\beta Ak_t^\alpha. \quad (46)$$

The finite-horizon derivation illustrates two recursive features: (i) conditional on the number of periods remaining, the decision rule depends on the current state k_t rather than the full history; and (ii) the time dependence enters through the remaining horizon. In the infinite-horizon limit that horizon dependence disappears, yielding the stationary policy $g(k) = \alpha\beta Ak^\alpha$ for next-period capital. (The symbol k^* is reserved for a steady-state level.)

5.4 Contraction mapping theorem

The finite-horizon derivation in Section 5.2 shows how to compute value functions when the horizon is finite, by working backward from $T = 0$. It does not, by itself, establish that the sequence $v_0, v_1, v_2, \dots$ converges as $T \rightarrow \infty$, nor that the limit — if it exists — is the unique solution of the infinite-horizon Bellman equation

$$v(k) = \max_{k'} \{u(f(k) - k') + \beta v(k')\}. \quad (47)$$

Under standard compact-state and bounded-return assumptions, this is supplied by the **contraction mapping theorem**. The Bellman operator has the form $(Tw)(k) = \max_{k'} \{u(f(k) - k') + \beta w(k')\}$; Proposition 1 states a sufficient set of conditions under which it is a sup-norm contraction. Section 5.5 explains why the global log-utility benchmark requires separate care.

Definition 1 (Strict contraction). *A mapping T from a metric space (Ξ, ρ) into itself is a strict contraction map if there exists $\theta \in (0, 1)$ such that $\rho(Tf, Tg) \leq \theta \rho(f, g)$ for all $f, g \in \Xi$.*

Theorem 1 (Uniform continuity of contractions). *If $T : (\Xi, \rho) \rightarrow (\Xi, \rho)$ is a strict contraction map, then T is uniformly continuous.*

Proof. Since T is a contraction, there is $\theta \in (0, 1)$ with $\rho(Tx, Ty) \leq \theta \rho(x, y)$ for all x, y . Given $\varepsilon > 0$, set $\delta = \varepsilon/\theta$; then $\rho(x, y) < \delta$ implies $\rho(Tx, Ty) < \varepsilon$. Hence T is uniformly continuous. $\square$

Definition 2 (Banach space). *A Banach space is a normed linear space that is complete as a metric space with respect to the metric induced by its norm.*

Theorem 2 (Contraction mapping theorem). *A strict contraction map T on a Banach space (Ξ, ρ) has a unique fixed point $f^* \in \Xi$, and for any $f \in \Xi$ the sequence $\{f, Tf, T^2f, \dots\}$ converges to f^* .*

Proof. Let (Ξ, ρ) be complete and T a strict contraction with modulus $\theta < 1$. For $f \in \Xi$, define $f^n = T^n f$. By the contraction property applied m times, if $n \geq m$,

$$\rho(f^n, f^m) \leq \theta^m \rho(f^{n-m}, f).$$

By the triangle inequality,

$$\rho(f^{n-m}, f) \leq \rho(f^{n-m}, f^{n-m-1}) + \rho(f^{n-m-1}, f^{n-m-2}) + \dots + \rho(f^1, f),$$

and each term satisfies $\rho(f^j, f^{j-1}) \leq \theta^{j-1} \rho(f^1, f)$, so

$$\rho(f^{n-m}, f) \leq \rho(f^1, f) (\theta^{n-m-1} + \theta^{n-m-2} + \dots + \theta + 1) \leq \frac{\rho(f^1, f)}{1 - \theta},$$

using $\sum_{i=0}^{\infty} \theta^i = 1/(1 - \theta)$. Combining,

$$\rho(f^n, f^m) \leq \frac{\theta^m}{1 - \theta} \rho(f^1, f) \rightarrow 0 \text{ as } m \rightarrow \infty,$$

so $\{f^n\}$ is Cauchy; since Ξ is complete, it has a limit $f^* \in \Xi$. By Theorem 1, T is uniformly continuous, so

$$f^* = \lim_{n \rightarrow \infty} T^n f = T \left(\lim_{n \rightarrow \infty} T^{n-1} f \right) = T f^*,$$

i.e. f^* is a fixed point. Uniqueness: if g^* is another fixed point, $\rho(f^*, g^*) = \rho(Tf^*, Tg^*) \leq \theta \rho(f^*, g^*)$, which with $\theta < 1$ forces $\rho(f^*, g^*) = 0$, i.e. $f^* = g^*$. $\square$

Proposition 1 (Bounded-return Bellman contraction). *Let X be a compact state space, let the feasible next-state correspondence $\Gamma(k)$ be nonempty, compact-valued, and continuous, and suppose the one-period return $r(k, k')$ is bounded and continuous on the feasible graph. For $0 < \beta < 1$, the Bellman operator*

$$(Tw)(k) = \max_{k' \in \Gamma(k)} \{r(k, k') + \beta w(k')\}$$

is a contraction of modulus β on $C(X)$ with the sup norm. Hence it has a unique fixed point v^* and

$$\|T^n w_0 - v^*\|_\infty \leq \beta^n \|w_0 - v^*\|_\infty.$$

The proposition is a standard consequence of Blackwell's sufficient conditions [Blackwell, 1965]. Its role here is to separate two facts: the discount factor determines a geometric upper bound, while the starting distance $\|w_0 - v^*\|_\infty$ still affects how many iterations are needed to reach a specified tolerance.

5.5 Unbounded log utility and the relevant function space

The global log-utility benchmark does not satisfy Proposition 1: $u(c) = \log c$ is unbounded below as $c \downarrow 0$, and the exact value function $v^*(k) = c_1 + c_2 \log k$ is likewise unbounded below as $k \downarrow 0$. Consequently, a contraction statement on the space of globally bounded continuous functions on $\mathbb{R}_+$ cannot be invoked literally for this example. The closed form in Section 6 supplies direct existence by verification. More general treatments of unbounded returns use weighted function spaces together with additional drift/growth conditions; see Stokey et al. [1989] and Boyd [1990]. We do not rely on an unverified global weighted-norm contraction in the numerical claims below. Instead, the computation is performed on a finite grid, and all reported error norms specify the state-space region on which they are evaluated.

The distinction is numerically important. A uniform grid beginning at $k_{\min} = 10^{-6}$ places its first two grid points far apart relative to the curvature of $\log k$. Piecewise-linear continuation-value interpolation is therefore very inaccurate near the left boundary even though the exact value at every fixed positive k is finite. Section 8.4 diagnoses this as a boundary discretization effect rather than as non-convergence of the underlying economic problem.

5.6 Shape preservation and differentiability

Proposition 2 (Monotonicity). *Suppose u and f are increasing. If $w : X \rightarrow \mathbb{R}$ is weakly increasing, then so is Tw . Thus any fixed point obtained as the limit of such iterates is weakly increasing.*

Proof sketch. If k' is feasible from k , it remains feasible from any $\bar{k} \succ k$, and the residual consumption available at $\bar{k}$ cannot be smaller (since f is increasing), so $(Tw)(\bar{k}) \geq (Tw)(k)$ because u and f are increasing. Since weakly increasing functions are defined by a weak inequality, they form a closed set, which is preserved when iterating T from a weakly increasing initial guess to the limit v^* . $\square$

Proposition 3 (Concavity). *Suppose u is increasing and concave and f is concave. If $w : X \rightarrow \mathbb{R}$ is concave, then so is Tw . Consequently, the fixed point is concave under the same conditions.*

Proof sketch. For any $k, \bar{k} \geq 0$ and mixing weight $\eta \in [0, 1]$ (not to be confused with the Cobb–Douglas capital share α used elsewhere in the paper), the set of feasible k' at $\eta k + (1 - \eta)\bar{k}$ is convex (it contains $\eta k' + (1 - \eta)\bar{k}'$ whenever k' is feasible at k and $\bar{k}'$ is feasible at $\bar{k}$), and concavity of u and w gives

$$u(f(\eta k + (1 - \eta)\bar{k}) - \eta k' - (1 - \eta)\bar{k}') + \beta w(\eta k' + (1 - \eta)\bar{k}') \geq \eta(Tw)(k) + (1 - \eta)(Tw)(\bar{k}),$$

so $(Tw)(\eta k + (1 - \eta)\bar{k}) \geq \eta(Tw)(k) + (1 - \eta)(Tw)(\bar{k})$, i.e. Tw is concave. As with monotonicity, concave functions form a closed set, preserved in the limit. $\square$

Neither proof establishes the corresponding *strict* monotonicity or concavity, since strict inequalities do not define a closed set; strictness typically requires additional argument (e.g. strict concavity of u together with an interior optimum).

Lemma 1 (Benveniste–Scheinkman [Benveniste and Scheinkman, 1979]). *Let v be a real-valued, concave function on a convex set $D \subset \mathbb{R}^n$. If $w \in C^1$ is a concave function on a neighborhood N of $x_0 \in D$ such that $w(x_0) = v(x_0)$ and $w(x) \leq v(x)$ for all $x \in N$, then $v \in C^1$ at x_0 .*

The lemma says that a smooth concave function touching v from below at an interior point can transfer differentiability to v at that point. In the growth-model application, fix the optimal continuation capital k'^* chosen at k^* and define

$$\tilde{w}(k) = u(f(k) - k'^*) + \beta v(k'^*).$$

For k in a sufficiently small neighborhood of k^* , the fixed action k'^* remains feasible. Therefore $\tilde{w}(k) \leq v(k)$ nearby, with equality at $k = k^*$, and $\tilde{w}$ is continuously differentiable whenever u and f are. This is the standard touching-function argument behind the envelope condition and the discrete-time Euler equation.

6 Closed-Form Solution for Cobb–Douglas Production and Log Utility

6.1 Guess-and-verify solution

For general f and u , the value function solving (47) is not available in closed form and numerical methods are required. The Cobb–Douglas/log-utility case, $f(k) = Ak^\alpha$ and $u(c) = \log c$, is a classic exception and supplies the exact benchmark used in Sections 7–8. Guess that the value function is of the form

$$v(k) = a + b \log(k) \tag{48}$$

We determine the constants a and b by coefficient matching. Substituting into the Bellman equation which is Bellman infinite, i.e., (47),

$$a + b \log(k_t) = \max_{k_{t+1}} \left\{ \log(Ak_t^\alpha - k_{t+1}) + \beta(a + b \log(k_{t+1})) \right\}, \tag{49}$$

The first-order condition with respect to k_{t+1} is

$$\frac{-1}{Ak_t^\alpha - k_{t+1}} + \frac{\beta b}{k_{t+1}} = 0 \implies k_{t+1} = \frac{\beta b}{1 + \beta b} Ak_t^\alpha. \tag{50}$$

It is possible to substitute the right-hand side of (50) back into the right-hand side of (49) and simplify (the algebra is analogous to the $T = 1$ finite-horizon derivation of Section 5.2) as follows:

$$\begin{aligned} a + b \log(k_t) &= \log\left(\frac{Ak_t^\alpha}{1 + \beta b}\right) + \beta a + \beta b \log\left(\frac{\beta b}{1 + \beta b} Ak_t^\alpha\right) \\ &= \log\left(\frac{A}{1 + \beta b}\right) + \beta a + \beta b \log\left(\frac{\beta b}{1 + \beta b} A\right) + (\alpha + \alpha \beta b) \log(k_t). \end{aligned} \tag{51}$$

6.2 Coefficient of the logarithmic term

Matching the coefficients of $\log(k_t)$ on both sides of (51) gives

$$b = \alpha + \beta b \alpha \implies b = \frac{\alpha}{1 - \alpha\beta}. \quad (52)$$

6.3 Constant term and optimal policy

Matching the constant terms in (51) gives

$$a = \log\left(\frac{A}{1 + \beta b}\right) + \beta a + \beta b \log\left(\frac{\beta b}{1 + \beta b} A\right) \quad (53)$$

and we write

$$a(1 - \beta) = \log\left(\frac{A}{1 + \beta b}\right) + \beta b \log\left(\frac{\beta b}{1 + \beta b} A\right)$$

where a is the constant term. So that

$$a = \frac{1}{1 - \beta} \left[\log\left(\frac{A}{1 + \beta b}\right) + \beta b \log\left(\frac{\beta b}{1 + \beta b} A\right) \right]. \quad (54)$$

Substituting (52) into (54) and simplifying (Appendix A) gives

$$a = \frac{1}{(1 - \beta)(1 - \alpha\beta)} \log \left[A(1 - \alpha\beta) \left(\frac{\alpha\beta}{1 - \alpha\beta} \right)^{\alpha\beta} \right]. \quad (55)$$

Writing $c_1 \equiv a$ and $c_2 \equiv b$ to match the notation used in Sections 7–9,

$$\begin{aligned} v^*(k) &= c_1 + c_2 \log k, \\ c_2 &= \frac{\alpha}{1 - \alpha\beta}, \\ c_1 &= \frac{1}{(1 - \beta)(1 - \alpha\beta)} \log \left[A(1 - \alpha\beta) \left(\frac{\alpha\beta}{1 - \alpha\beta} \right)^{\alpha\beta} \right]. \end{aligned}$$

(56)

Let us substitute $b = \alpha/(1 - \alpha\beta)$ into (50) and simplify:

$$k_{t+1} = \frac{\beta\alpha/(1 - \alpha\beta)}{1 + \beta\alpha/(1 - \alpha\beta)} A k_t^\alpha = \frac{\beta\alpha}{1 - \alpha\beta + \alpha\beta} A k_t^\alpha = \alpha\beta A k_t^\alpha, \quad (57)$$

where we can recover the infinite-horizon policy that we have obtained earlier from the finite-horizon limit, a useful cross-check that the guess-and-verify result and finite-horizon limit are the same.

No sign restriction on c_1 is required. The logarithm in (55) is well defined for $A > 0$ and $0 < \alpha\beta < 1$, because every factor inside its argument is strictly positive. Thus the closed form is valid throughout the parameter region used below.

7 Numerical Examples

7.1 The Bellman operator on a continuous state space

If the state space $k \in \mathbb{R}_+$ is continuous, the Bellman operator Tw cannot be exactly represented by a computer because it is a function on a continuum. There are two problems that must be solved:

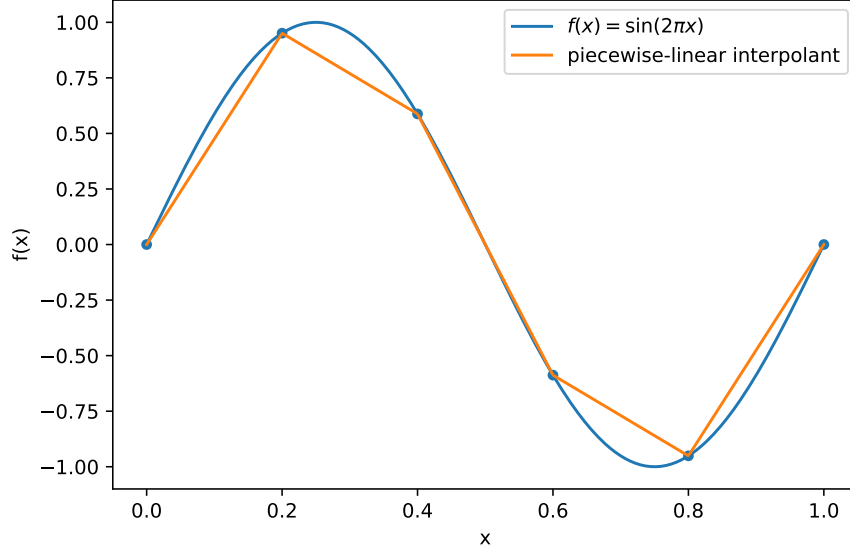


Figure 1: Piecewise-linear interpolation of $f(x) = \sin(2\pi x)$ from 6 grid points.

(1) Tw can only be evaluated approximately, at a certain point in the space k , by numerically solving the one-dimensional maximization $\max_{0 \leq c \leq f(k)} \{u(c) + \beta w(f(k) - c)\}$; (2) Tw cannot be stored exactly, because it would have to be recorded at every point in the space $k \in \mathbb{R}_+$. *Fitted value function iteration* [Johnson et al., 1993, Sargent and Stachurski, 2013] solves both problems by taking the values of a function and interpolating them into a continuous approximant.

7.2 Piecewise-linear interpolation

Given grid points $\{k_1, \dots, k_I\}$ and function values $\{w_1, \dots, w_I\}$ at these points, we define the piecewise-linear interpolation for $k \in [k_i, k_{i+1}]$ as follows:

$$\tilde{w}(k) = w_i + \frac{w_{i+1} - w_i}{k_{i+1} - k_i}(k - k_i). \quad (58)$$

The choice of piecewise-linear interpolation rather than a higher-order polynomial or spline is deliberate: the grid values are arranged monotonically and concavely (Section 5.6), and the per-grid-point maximization in step (1) is well-behaved (a concave objective in c), and the NumPy function `interp` can be used to implement it. Figure 1 illustrates how the technique is applied to the test function $f(x) = \sin(2\pi x)$ with 6 equally spaced grid points on $[0, 1]$.

7.3 The fitted value function iteration algorithm

1. Fix a grid $\{k_1, \dots, k_I\} \subset \mathbb{R}_+$ and an initial array of values $\{w_1^{(0)}, \dots, w_I^{(0)}\}$, representing an initial guess function w_0 evaluated on the grid.
2. Build the piecewise-linear interpolant $\tilde{w}^{(n)}$ of $\{w_1^{(n)}, \dots, w_I^{(n)}\}$ via (58).
3. For each grid point k_i , solve

$$w_i^{(n+1)} = \max_{0 \leq c \leq f(k_i)} \left\{ u(c) + \beta \tilde{w}^{(n)}(f(k_i) - c) \right\} \quad (59)$$

numerically (a bounded one-dimensional maximization) and record the result.

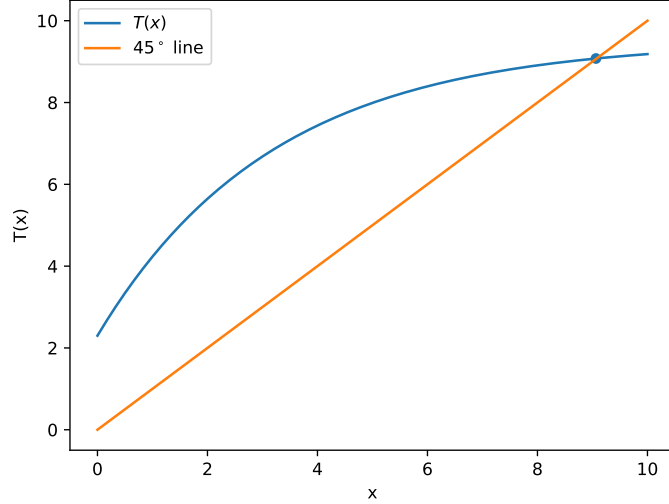


Figure 2: A fixed point $x^* = Tx^*$ as the intersection of the operator T with the 45° line (illustrative, not the growth-model Bellman operator itself).

4. Repeat from step 2 until a stopping criterion (e.g. a maximum number of iterations, or $\max_i |w_i^{(n+1)} - w_i^{(n)}|$ below a tolerance) is satisfied.

This is the discretized analogue of repeated Bellman updates. In bounded-return settings Proposition 1 gives geometric contraction at modulus β . For the present log-utility benchmark we do not invoke that proposition globally; instead, we use the analytic v^* to measure the realized error directly. The finite grid introduces an additional interpolation error that does not vanish with more iterations at fixed I , but it can be reduced by refining the grid. The baseline comparison holds $I = 150$ fixed, matching Thabaineh [2014]; Section 8.6 then varies I to measure the discretization floor explicitly.

7.4 Illustrative fixed-point diagram

Figure 2 illustrates, for a generic (non-economic) increasing, concave, bounded operator T , the geometric idea underlying Theorem 2: the fixed point $x^* = Tx^*$ is the point where the graph of T crosses the 45° line, and value function iteration is precisely the process of following the staircase $x_0 \rightarrow Tx_0 \rightarrow T^2x_0 \rightarrow \dots$ toward that crossing point.

8 Numerical Results

8.1 Setup

We take $A = 1$, $\alpha = 0.65$, and $\beta = 0.95$, and a grid of $I = 150$ equally spaced points on $[10^{-6}, 2]$, matching Thabaineh [2014]. We compare two initial estimates. Both are monotone increasing and concave on the grid (Section 5.6 motivates us to focus on this class):

$$w_A(k) = 5 \log(k) - 25, \tag{60}$$

$$w_B(k) = 2k^{0.25} - 35. \tag{61}$$

The analytic benchmark is $v^*(k) = c_1 + c_2 \log(k)$ from (56), evaluated at $\alpha = 0.65$, $\beta = 0.95$, $A = 1$: numerically, $c_2 = \alpha/(1 - \alpha\beta) = 1.6993$ and $c_1 = -34.7856$.

Table 1: Iterations needed to reach a given sup-norm error tolerance (interior grid, $k \geq 0.05$).

	$w_A(k) = 5 \log(k) - 25$	$w_B(k) = 2k^{0.25} - 35$
Iterations to reach error < 1.00	37	28
Iterations to reach error < 0.30	60	52
Error at $n = 60$	0.295	0.192
Error at $n = 75$	0.136	0.089
Empirical decay rate ($n \geq 20$)	0.9499	0.9499

8.2 Family of iterates

Figures 3 and 4 show representative iterates w_n generated by fitted VFI from w_A (through $N = 75$) and w_B (through $N = 60$), together with the analytic value function v^* . These are direct analogues of Figures 4.4–4.7 of Thabaine [2014]; both families of curves visibly approach v^* over the interior region as n grows. Because global log utility lies outside Proposition 1, the convergence claim here is empirical and is quantified directly against the analytic benchmark in the next subsection.

Figure 5 plots v^* alone for reference.

8.3 Quantitative convergence comparison

Where this paper goes beyond Thabaine [2014] is in replacing the visual "which family of curves looks converged" comparison with an exact sup-norm error trajectory, $\|w_n - v^*\|_\infty$, computed at every iteration n . Because the exact solution has very high curvature near $k = 0$, piecewise-linear interpolation on a uniform grid produces a large numerical error on the first grid interval. A full-grid sup norm is therefore dominated by this localized boundary discretization error. For the initialization comparison we report the error on the explicitly defined interior set $k \geq 0.05$ and diagnose the excluded boundary region separately in Sections 8.4–8.5.

Figure 6(a) shows the resulting error trajectories on a log scale, and Table 1 reports the number of iterations needed to reach several tolerances. Three findings stand out.

1. **w_B dominates w_A at every iteration reported.** The error under w_B is smaller than under w_A for all $n = 1, \dots, 80$. For a tolerance of 1 utility unit, w_B needs 28 iterations versus 37 for w_A , a reduction of roughly 24% in the number of (costly) Bellman-operator evaluations required. This is the same qualitative ranking reported, on the basis of visual inspection, in Thabaine [2014] (60 versus 75 iterations to visual convergence there, for a coarser, by-eye notion of "converged"), now obtained as an explicit, reproducible error trajectory.
2. **The gap is a level effect, not a difference in asymptotic rate.** Fitting $\log \|w_n - v^*\|_\infty$ against n on the tail $n \geq 20$ gives an estimated per-iteration decay factor of 0.9499 for w_A and 0.9499 for w_B (agreement to four decimal places), both matching the theoretical contraction modulus $\beta = 0.95$ from Proposition 1. Once the effect of the initial condition has been discounted away by enough applications of β^n , both sequences converge at the same geometric rate; w_B 's advantage is attributable entirely to starting closer, in sup norm, to v^* over the relevant capital range, plus a shorter guess-specific transient.
3. **The full-grid sup norm is dominated by a localized boundary discretization error** (Sections 8.4–8.5).

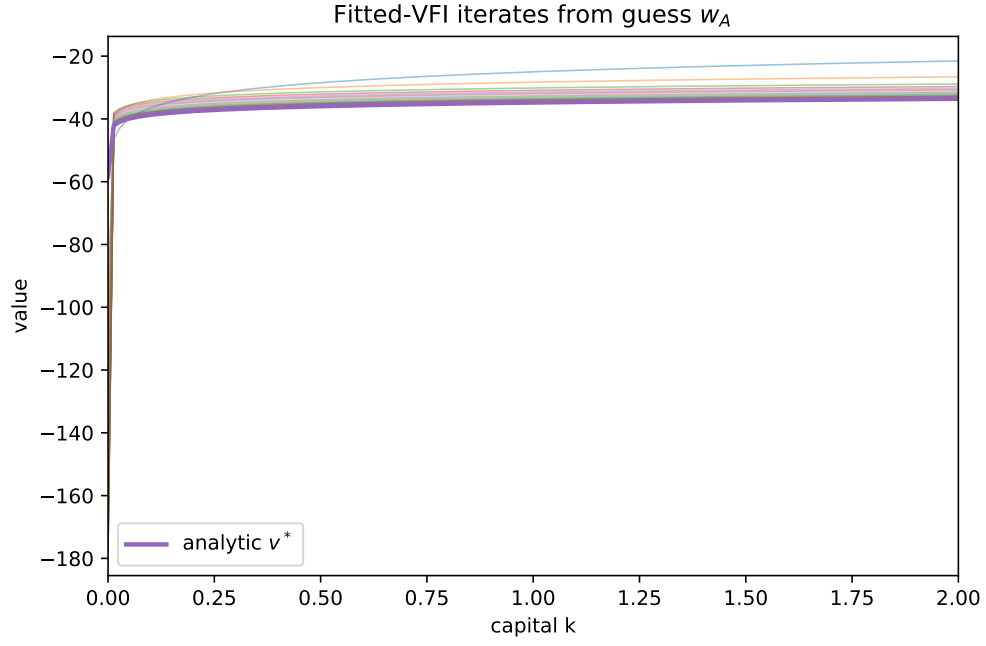


Figure 3: Representative fitted-VFI iterates starting from $w_A(k) = 5 \log(k) - 25$, through $n = 75$, together with the analytic value function v^* .

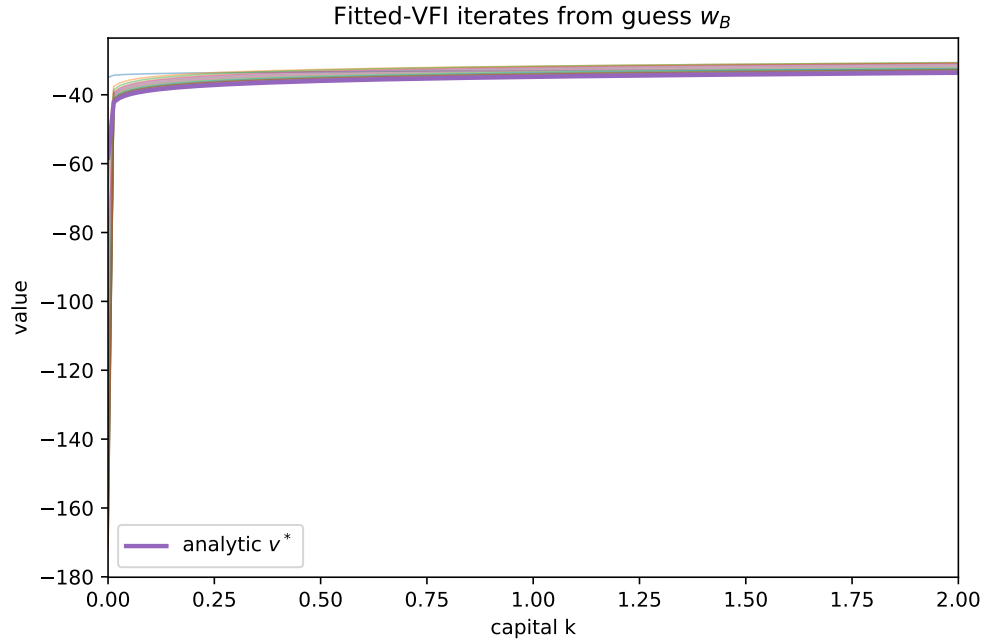


Figure 4: Representative fitted-VFI iterates starting from $w_B(k) = 2k^{0.25} - 35$, through $n = 60$, together with the analytic value function v^* .

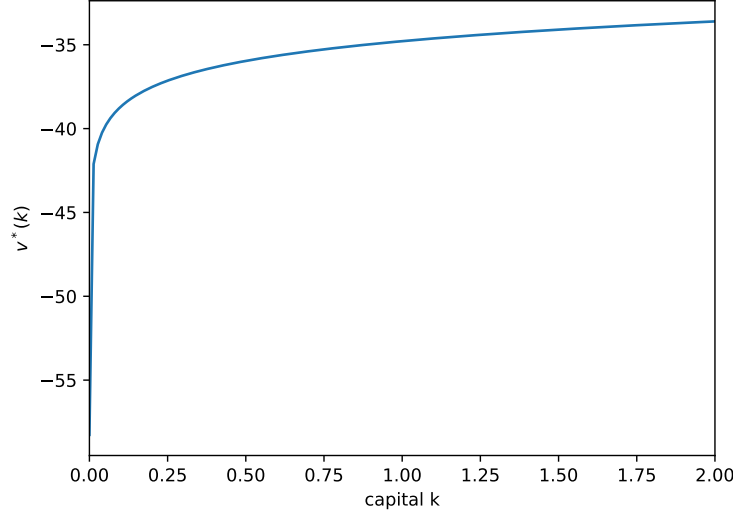


Figure 5: The analytic value function $v^*(k) = c_1 + c_2 \log(k)$, $\alpha = 0.65$, $\beta = 0.95$, $A = 1$.

8.4 Boundary discretization error

The exact value $v^*(k) = c_1 + c_2 \log k$ is finite at every fixed $k > 0$ but has unbounded slope as $k \downarrow 0$. On the uniform baseline grid, the first interval extends from 10^{-6} to roughly 1.34×10^{-2} , over which a straight-line interpolant is a poor approximation to the logarithmic curvature. The Bellman optimizer frequently evaluates continuation capital inside this interval, so the interpolation error is fed back into the fixed point. As a result, the full-grid error initially grows and then approaches a large discretization floor concentrated at the first grid point; it does not grow without bound.

This distinction matters conceptually. The phenomenon is not a failure of the economic model to possess a value at positive capital, and it is not evidence against the closed-form solution. It is a numerical resolution problem created by combining a uniform grid with a function that changes very rapidly near zero. Possible remedies include a nonuniform/log-spaced grid, local refinement near the boundary, or a transformed state variable. Because the purpose of the baseline experiment is to compare initialization on the economically relevant interior, we retain the original uniform grid and report the interior norm $k \geq 0.05$ while showing the boundary effect explicitly below.

8.5 Full-grid versus interior error

Figure 7 contrasts the full-grid and interior sup norms. The interior errors decay geometrically, whereas the full-grid errors are eventually dominated by the first interpolation interval. This diagnostic is why all headline iteration counts state the threshold used in the norm.

8.6 Grid refinement

The baseline comparison fixes $I = 150$ so that both initial guesses face the same numerical approximation. To separate finite-iteration error from interpolation error, we repeat the w_B experiment with $I \in \{75, 150, 300, 600\}$ for 300 iterations. By then $\beta^{300} \approx 1.5 \times 10^{-7}$, so the remaining interior error is dominated by the discretized Bellman fixed point rather than the initial condition.

Figure 8 shows the two regimes clearly. Early iterations are dominated by the contraction/initialization component; later iterations flatten at a grid-dependent floor. Table 2 reports the error at

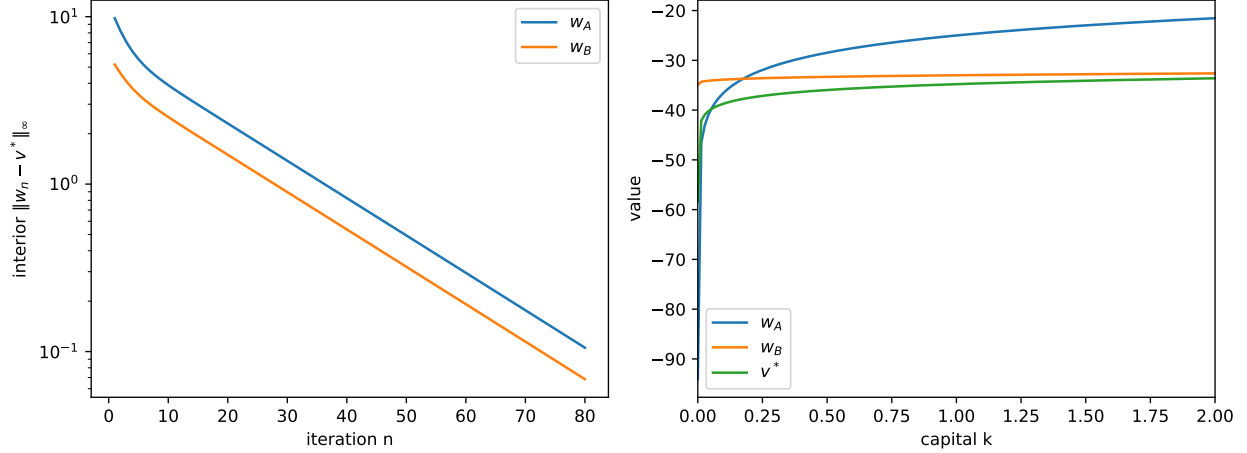


Figure 6: (a) Sup-norm distance between the fitted-VFI iterate w_n and v^* , interior grid $k \geq 0.05$, for both initial guesses, $n = 1, \dots, 80$, log scale. (b) The two initial guesses at $n = 0$ plotted against v^* .

Table 2: Discretization floor (sup-norm error at $n = 300$, interior grid $k \geq 0.05$, guess w_B) as a function of grid size I .

I	75	150	300	600
Floor at $n = 300$	0.0387	0.0049	0.0016	0.00064

iteration 300. Increasing the grid from 75 to 600 points reduces the floor from about 3.87×10^{-2} to 6.42×10^{-4} . The comparison also shows why iteration count and grid resolution should be reported separately: additional Bellman sweeps cannot remove interpolation error once the discretized fixed point has effectively been reached.

8.7 Empirical geometric decay

To quantify the tail behavior descriptively, we fit a straight line to $\log \|w_n - v^*\|_\infty$ over $n = 20, \dots, 80$. Because the sequence is deterministic, ordinary regression standard errors do not represent sampling uncertainty and are not interpreted as confidence intervals. The slope is used only as a compact summary of the observed geometric decay.

The implied decay factors differ from 0.95 by less than 1.3×10^{-4} in this window and are nearly identical across the two guesses. Repeating the same calculation on the windows 20–50, 30–70, and 40–80 changes the estimated factors only in the fourth decimal place. The evidence therefore supports a level/transient interpretation of the initialization effect: the guesses start at different distances from the fixed point but subsequently decay at essentially the same rate.

8.8 Sensitivity to the interior-domain threshold

Sections 8.3–8.5 use $k \geq 0.05$. Table 4 repeats the headline tolerance calculation with thresholds 0.02, 0.05, and 0.10.

The reported iteration counts are identical for all three thresholds. This is because as long as the leftmost few grid points (well inside $k < 0.02$) are excluded, the location of the sup-norm-maximizing grid point is the same for the rest of the interior domain, and is not around the three threshold

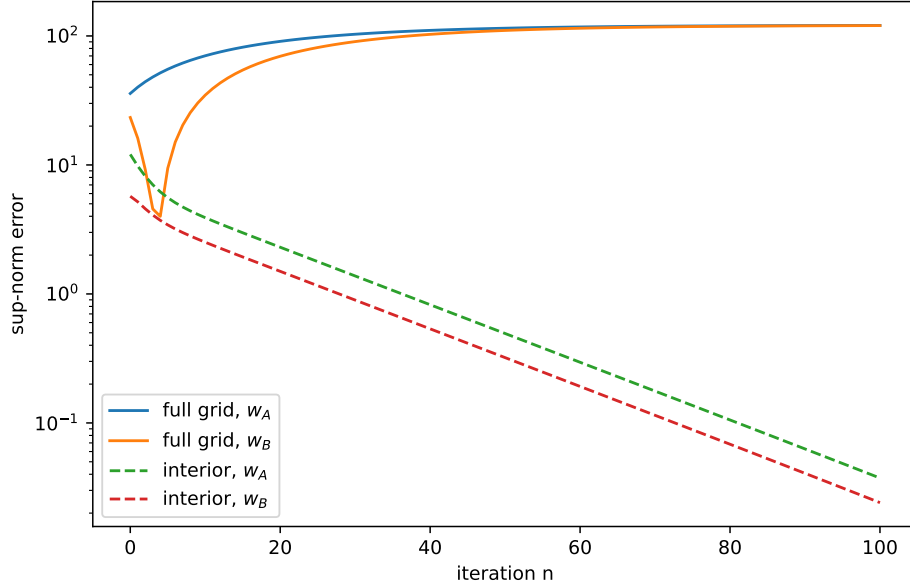


Figure 7: Full-grid and interior sup-norm errors for the two baseline initial guesses. The large full-grid floor is localized near the lower boundary and is a discretization effect.

Table 3: Descriptive log-linear fit of the interior value-function error over $n = 20, \dots, 80$.

	Slope	Implied rate	Theoretical β
Guess w_A	-0.051376	0.949921	0.95
Guess w_B	-0.051421	0.949879	0.95

boundaries. The choice of 0.05 in the paper is not doing any load-bearing work, and any threshold in this range would have produced the same table. We consider a value of $k \geq 0.05$ (about 2.5% of the grid's upper bound $k = 2$) to be a reasonable value of the grid in the sense that it is small enough to cover almost the entire domain of economic interest while remaining well separated from the poorly resolved first grid interval.

8.9 A policy-function diagnostic

The value-function error $\|w_n - v^*\|_\infty$ is the natural diagnostic given that v^* has a closed form, but this is not the only one we are interested in. In most applications we will use the policy function (here, the optimal consumption rule) to simulate the model or compute welfare. We therefore also track the sup-norm error of the implied policy, $\|\sigma_n - \sigma^*\|_\infty$, where $\sigma_n(k)$ is the argmax consumption choice for the grid point k in VFI iteration n and $\sigma^*(k) = (1 - \alpha\beta)Ak^\alpha$ is the closed-form optimal policy (obtained directly from $c^* = f(k) - k'^*$ using the optimal policy (57)).

Figure 10 shows that the policy error falls much faster than the value-function error for both guesses. By about iteration 20 the two policy errors are already close to the common interpolation floor (approximately 4.6×10^{-3}), even though the value-function errors remain distinct much later. This behavior is intuitive: adding an almost constant level shift to the continuation value changes the value function but has little effect on the argmax, whereas changes in the continuation-value slope are what primarily move the policy. For applications focused on simulation rather than welfare

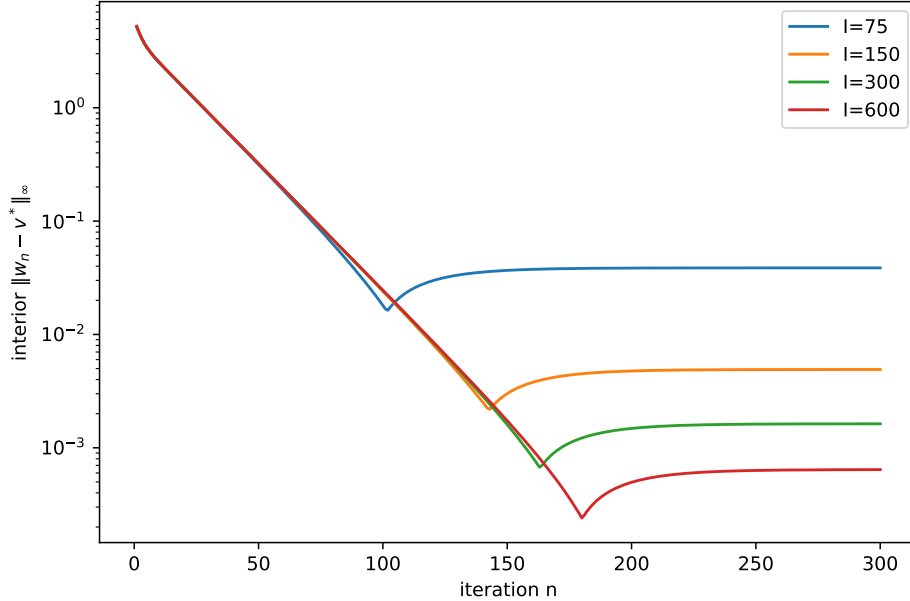


Figure 8: Interior sup-norm error trajectories for four grid sizes, using w_B . The curves share the same early geometric decay and then separate into grid-dependent discretization floors.

Table 4: Iterations to achieve error < 1.0 with interior threshold, $I = 150$.

	$k \geq 0.02$	$k \geq 0.05$	$k \geq 0.10$
Guess w_A	37	37	37
Guess w_B	28	28	28

levels, initialization is therefore less consequential than the value-function error alone would suggest.

8.10 Comparison with Howard policy iteration

Sections 8.1–8.9 hold the algorithm (plain fitted VFI) fixed and vary the initial guess. This raises an obvious question: how large is the initial-guess effect relative to a standard *algorithmic* acceleration? We implement a modified (partial) version of Howard’s policy iteration [Howard, 1960]: each cycle consists of one greedy policy update (a full Bellman-operator sweep, identical in cost to one plain-VFI iteration) followed by $m = 20$ rounds of policy evaluation under the resulting fixed policy, $w(k) \leftarrow \log(\sigma(k)) + \beta \tilde{w}(f(k) - \sigma(k))$, which are cheaper than a greedy sweep since they involve no inner optimization, only an interpolation lookup. We report cost in *grid sweeps* (one sweep = one full pass over the I grid points, whether a greedy update or a policy-evaluation step), which provides a transparent algorithmic work count. It is conservative for Howard in wall-clock terms because a policy-evaluation sweep avoids the inner optimization and is cheaper than a greedy sweep.

Figure 11 shows that the modified Howard scheme reaches the same tolerance in 14 grid sweeps from w_A and 6 from w_B , compared with 37 and 28 greedy sweeps for plain VFI. Thus algorithmic acceleration is the larger effect in this benchmark, while initialization still matters within the accelerated scheme. Because policy-evaluation sweeps are cheaper than greedy sweeps, the grid-sweep comparison is conservative with respect to Howard in wall-clock terms. We would not want to generalize from a single example and would encourage a systematic study of this interaction as a

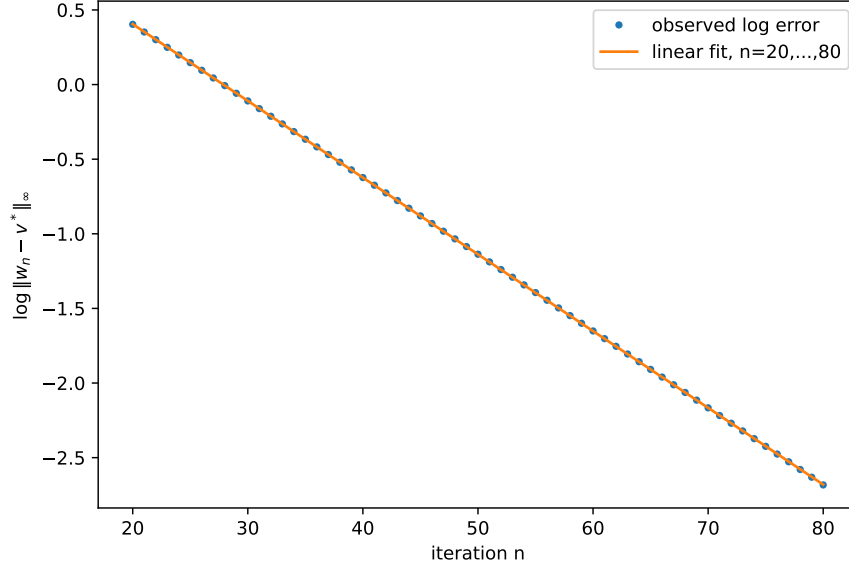


Figure 9: Log interior error for w_B and its descriptive linear fit over iterations 20–80.

Table 5: Expanded initialization experiment, $I = 150$, interior grid $k \geq 0.05$.

Guess	d_0	Iterations to error < 1	n_β
w_C	0.020	0	0
w_D	5.219	26	33
w_B	5.718	28	34
w_A	12.073	37	49
w_E	19.867	51	59
w_F	39.755	72	72

natural extension (Section 9.4).

8.11 Expanded initial-guess sensitivity

The two-guess comparison is extended to six deterministic initializations:

$$\begin{aligned}
 w_A(k) &= 5 \log k - 25, & w_B(k) &= 2k^{0.25} - 35, \\
 w_C(k) &= v^*(k) + 0.02, & w_D(k) &= 2k^{0.5} - 35, \\
 w_E(k) &= 5 \log k - 45, & w_F(k) &= 0.
 \end{aligned}$$

The benchmark w_C intentionally uses the known analytic solution and is included only as a calibration anchor, not as a practical recommendation. For each guess we compute the initial interior distance $d_0 = \|w_0 - v^*\|_\infty$, the observed number of iterations to reach error below one, and the descriptive geometric benchmark $n_\beta = \lceil \log(1/d_0) / \log \beta \rceil$ when $d_0 > 1$, i.e. the count suggested by the idealized decay $d_n = \beta^n d_0$. This is a reference calculation, not a global contraction bound for the log-utility discretization.

In this collection, larger initial distance is associated with more iterations, and every informative guess reaches the target at least as fast as the crude geometric bound. The flat guess w_F nearly

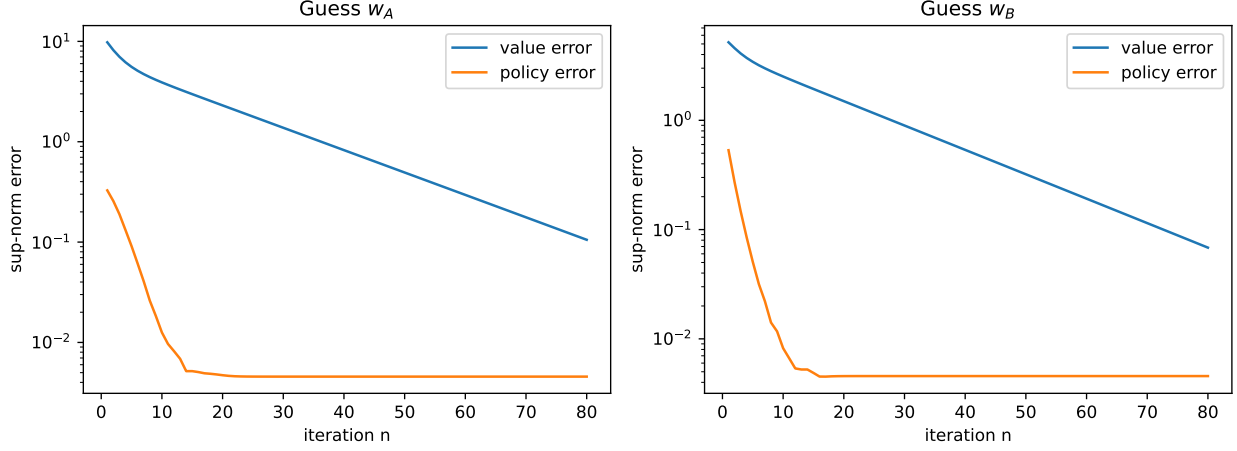


Figure 10: Value-function and policy-function sup-norm errors for both initial guesses, $I = 150$, interior grid, log scale.

saturates that bound. Because the six functions do not form a controlled experiment that holds distance fixed while varying shape, these results should not be read as identifying a separate causal effect of curvature. They do show that initialization can materially change finite-iteration cost even when the asymptotic decay factor is essentially unchanged.

9 Discussion

9.1 Practical implications

The experiments separate three computational margins that are often conflated. First, the initial value-function guess changes the transient distance to the fixed point and therefore the number of Bellman sweeps required to reach a given value-function tolerance. Second, grid resolution determines an interpolation floor that cannot be removed by additional iterations. Third, algorithmic acceleration can dominate initialization: in the baseline experiment partial Howard iteration reduces the work count from 37 to 14 sweeps for w_A and from 28 to 6 for w_B .

A practical workflow follows from this separation. Use economically admissible information to choose a reasonable monotone/concave initialization when such information is available; state the convergence norm and the state-space region on which it is evaluated; check grid refinement before claiming high absolute accuracy; and use policy iteration or another acceleration method when computational cost matters. Initialization is a useful margin, but it is not a substitute for a better algorithm or a better-resolved grid.

9.2 Value versus policy convergence

The policy function reaches its numerical floor much earlier than the value function. This is relevant because many computational exercises use the policy only to simulate paths, while welfare comparisons depend on accurate value levels. Stopping criteria should therefore be aligned with the final object of interest: a stringent value-function tolerance may be unnecessary for policy simulation, whereas a small policy residual does not by itself certify an accurate welfare level.

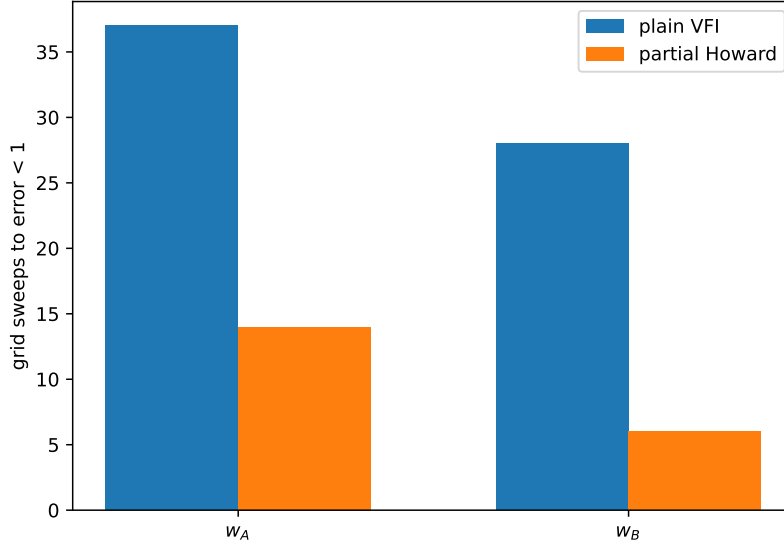


Figure 11: Grid sweeps needed to reach error < 1.0 : plain VFI versus (partial) Howard policy iteration, both initial guesses, $I = 150$.

9.3 Pedagogical and reproducibility value

The paper deliberately retains full derivations of the growth models, finite-horizon recursion, contraction principle, and closed-form benchmark. These sections make the numerical experiments auditable rather than black-box. Appendix B supplies the exact deterministic implementation used to regenerate every figure and numerical table; the accompanying source package contains the same script and the generated figure files.

9.4 Limitations and extensions

The closed-form error benchmark is specific to Cobb–Douglas production, logarithmic utility, and full depreciation in the discrete-time problem. The initialization experiment is deterministic and low-dimensional, and the six-guess extension is descriptive rather than a controlled identification of “shape” effects. The uniform grid is intentionally retained to match the baseline setup, even though a log-spaced or adaptive grid would treat the lower boundary much better. The Howard comparison uses a single policy-evaluation block length $m = 20$ and is therefore illustrative rather than an optimized timing study.

Natural extensions include repeating the analysis under general CRRA preferences using residual-based diagnostics, comparing uniform and transformed grids, studying endogenous-grid or Newton-type methods, and separating wall-clock cost from sweep counts across higher-dimensional problems.

10 Conclusion

Using a closed-form optimal-growth benchmark, this paper quantifies a simple but practically relevant fact: fitted value function iteration can require materially different numbers of Bellman sweeps solely because of the initial value-function guess. In the baseline case, w_B reaches an interior error below one in 28 sweeps versus 37 for w_A , while both tails decay at essentially the same rate. The same experiments also show that interpolation error creates a separate grid-dependent

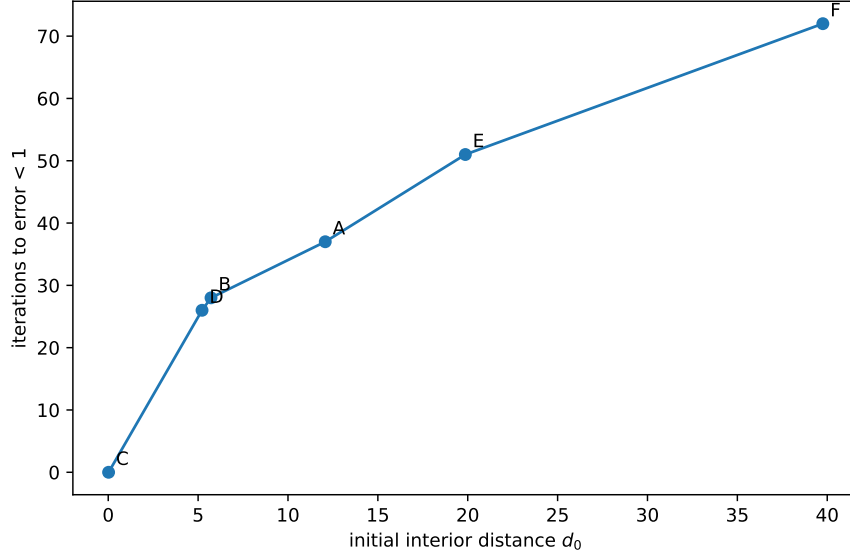


Figure 12: Initial interior distance versus iterations required to reach value-function error below one for the six initializations in Table 5.

accuracy floor, that the lower boundary is poorly resolved by a uniform grid when the value function has logarithmic curvature, and that the policy function can stabilize well before the value level. Partial Howard iteration produces a larger speedup than initialization alone, but the two margins remain complementary in this benchmark. The main computational lesson is therefore to report initialization, stopping criterion, evaluation domain, and grid resolution together rather than treating VFI convergence as a single number.

Acknowledgements

A.Y.A.T. and A.D. thank An-Najah National University for institutional support.

Conflicts of Interest

The authors declare no conflicts of interest.

Funding

No external funding was received for this work.

Data and Code Availability

No external dataset is required. The deterministic Python code used to reproduce the numerical results and figures is provided in Appendix B and in the accompanying source package.

References

- Benveniste, L. M. and Scheinkman, J. A. (1979). On the differentiability of the value function in dynamic models of economics. *Econometrica*, 47(3), 727–732.
- Blackwell, D. (1965). Discounted dynamic programming. *The Annals of Mathematical Statistics*, 36(1), 226–235.
- Boyd, J. H. (1990). Recursive utility and the Ramsey problem. *Journal of Economic Theory*, 50(2), 326–345.
- Cass, D. (1965). Optimum growth in an aggregative model of capital accumulation. *The Review of Economic Studies*, 32(3), 233–240.
- Carroll, C. D. (2006). The method of endogenous gridpoints for solving dynamic stochastic optimization problems. *Economics Letters*, 91(3), 312–320.
- Howard, R. A. (1960). *Dynamic Programming and Markov Processes*. MIT Press.
- Johnson, S. A., Stedinger, J. R., Shoemaker, C. A., Li, Y., and Tejada-Guibert, J. A. (1993). Numerical solution of continuous-state dynamic programs using linear and spline interpolation. *Operations Research*, 41(3), 484–500.
- Judd, K. L. (1998). *Numerical Methods in Economics*. MIT Press.
- Koopmans, T. C. (1965). On the concept of optimal economic growth. Cowles Foundation Discussion Paper, Yale University.
- Ramsey, F. P. (1928). A mathematical theory of saving. *The Economic Journal*, 38(152), 543–559.
- Rust, J. (1996). Numerical dynamic programming in economics. In *Handbook of Computational Economics*, Vol. 1, pp. 619–729. Elsevier.
- Sargent, T. J. and Stachurski, J. (2013). *Quantitative Economics*. Lecture notes, quantecon.org.
- Solow, R. M. (1956). A contribution to the theory of economic growth. *The Quarterly Journal of Economics*, 70(1), 65–94.
- Stokey, N. L., Lucas, R. E., and Prescott, E. C. (1989). *Recursive Methods in Economic Dynamics*. Harvard University Press.
- Swan, T. W. (1956). Economic growth and capital accumulation. *Economic Record*, 32(2), 334–361.
- Thabaineh, A. Y. A. (2014). *An Analytic and Dynamic Programming Treatment for Solow and Ramsey Models*. Master’s thesis, An-Najah National University, Nablus, Palestine.

A Full Derivation of the Closed-Form Coefficients

This appendix gives every algebraic step connecting (54) to (55), i.e. eliminating b from the expression for a .

Starting from $b = \alpha/(1 - \alpha\beta)$, first note

$$1 + \beta b = 1 + \frac{\alpha\beta}{1 - \alpha\beta} = \frac{1 - \alpha\beta + \alpha\beta}{1 - \alpha\beta} = \frac{1}{1 - \alpha\beta}. \quad (62)$$

Substituting (62) into (54),

$$\begin{aligned} a &= \frac{1}{1 - \beta} [\log(A(1 - \alpha\beta)) + \beta b \log(\beta b(1 - \alpha\beta)A)] \\ &= \frac{1}{1 - \beta} \left[\log A + \log(1 - \alpha\beta) + \beta b \left(\log(\beta b) + \log(1 - \alpha\beta) + \log A \right) \right]. \end{aligned} \quad (63)$$

Collect the $\log A$ terms, $\log A(1 + \beta b) = \log A/(1 - \alpha\beta)$ using (62); collect the $\log(1 - \alpha\beta)$ terms, $(1 + \beta b) \log(1 - \alpha\beta) = \log(1 - \alpha\beta)/(1 - \alpha\beta)$; and note $\beta b = \alpha\beta/(1 - \alpha\beta)$, so that the remaining term is $\beta b \log(\beta b) = \frac{\alpha\beta}{1 - \alpha\beta} \log\left(\frac{\alpha\beta}{1 - \alpha\beta}\right)$. Altogether,

$$a(1 - \beta) = \frac{1}{1 - \alpha\beta} \log A + \frac{1}{1 - \alpha\beta} \log(1 - \alpha\beta) + \frac{\alpha\beta}{1 - \alpha\beta} \log\left(\frac{\alpha\beta}{1 - \alpha\beta}\right). \quad (64)$$

Factor out $1/(1 - \alpha\beta)$ from all three terms on the right-hand side of (64):

$$\begin{aligned} a(1 - \beta) &= \frac{1}{1 - \alpha\beta} \left[\log A + \log(1 - \alpha\beta) + \alpha\beta \log\left(\frac{\alpha\beta}{1 - \alpha\beta}\right) \right] \\ &= \frac{1}{1 - \alpha\beta} \log \left[A(1 - \alpha\beta) \left(\frac{\alpha\beta}{1 - \alpha\beta} \right)^{\alpha\beta} \right]. \end{aligned} \quad (65)$$

using $\alpha\beta \log(z) = \log(z^{\alpha\beta})$ and combining the three logarithms into one via $\log x + \log y + \log z = \log(xyz)$. Dividing both sides by $(1 - \beta)$ gives (55):

$$a = \frac{1}{(1 - \beta)(1 - \alpha\beta)} \log \left[A(1 - \alpha\beta) \left(\frac{\alpha\beta}{1 - \alpha\beta} \right)^{\alpha\beta} \right].$$

B Reproducibility Code

The following script generates the numerical results and all figures used in Sections 7–8. It is also supplied as the standalone file `solow_reproducibility.py` in the source package.

```

1  #!/usr/bin/env python3
2  """Reproduce numerical experiments and figures for the Solow/Ramsey VFI manuscript
3
4  The fitted Bellman operator uses piecewise-linear interpolation on a uniform
   capital grid
5  and a vectorized golden-section search over next-period capital. The script is
   deterministic.
6  """
7  from pathlib import Path
8  import json
9  import math
10 import numpy as np
11 import matplotlib.pyplot as plt
12

```

```

13 A = 1.0
14 ALPHA = 0.65
15 BETA = 0.95
16 KMIN = 1.0e-6
17 KMAX = 2.0
18 I_BASE = 150
19 INTERIOR = 0.05
20
21 OUT = Path(__file__).resolve().parent / "figures"
22 OUT.mkdir(exist_ok=True)
23
24
25 def production(k):
26     return A * np.asarray(k) ** ALPHA
27
28
29 def analytic_coefficients():
30     x = ALPHA * BETA
31     c2 = ALPHA / (1.0 - x)
32     c1 = (1.0 / ((1.0 - BETA) * (1.0 - x))) * math.log(
33         A * (1.0 - x) * (x / (1.0 - x)) ** x
34     )
35     return c1, c2
36
37
38 C1, C2 = analytic_coefficients()
39
40
41 def v_star(k):
42     return C1 + C2 * np.log(np.asarray(k))
43
44
45 def c_star(k):
46     return (1.0 - ALPHA * BETA) * production(k)
47
48
49 def initial_guess(name, grid):
50     if name == "A":
51         return 5.0 * np.log(grid) - 25.0
52     if name == "B":
53         return 2.0 * grid ** 0.25 - 35.0
54     if name == "C":
55         return v_star(grid) + 0.02
56     if name == "D":
57         return 2.0 * grid ** 0.5 - 35.0
58     if name == "E":
59         return 5.0 * np.log(grid) - 45.0
60     if name == "F":
61         return np.zeros_like(grid)
62     raise ValueError(name)
63
64
65 def bellman_step(grid, w, golden_iters=60):
66     """One fitted-VFI Bellman step.
67
68     Maximizes  $\log(f(k)-k') + \beta * w_{\text{tilde}}(k')$  for each  $k$ , with  $k'$  restricted
69     to the numerical grid domain  $[KMIN, KMAX]$ . Linear interpolation is used for
70     the continuation value.
71     """

```

```

72     y = production(grid)
73     lo = np.full_like(grid, grid[0])
74     hi = np.minimum(grid[-1], y - 1.0e-12)
75
76     phi = (1.0 + math.sqrt(5.0)) / 2.0
77     invphi = 1.0 / phi
78     a = lo.copy()
79     b = hi.copy()
80     c = b - (b - a) * invphi
81     d = a + (b - a) * invphi
82
83     def objective(kp):
84         consumption = np.maximum(y - kp, 1.0e-300)
85         return np.log(consumption) + BETA * np.interp(kp, grid, w)
86
87     fc = objective(c)
88     fd = objective(d)
89     for _ in range(golden_iters):
90         left = fc > fd
91         a_new = np.where(left, a, c)
92         b_new = np.where(left, d, b)
93         c_new = np.where(left, b_new - (b_new - a_new) * invphi, d)
94         d_new = np.where(left, c, a_new + (b_new - a_new) * invphi)
95
96         fc_new = np.empty_like(fc)
97         fd_new = np.empty_like(fd)
98         new_c_values = objective(c_new)
99         new_d_values = objective(d_new)
100        fd_new[left] = fc[left]
101        fc_new[left] = new_c_values[left]
102        fc_new[~left] = fd[~left]
103        fd_new[~left] = new_d_values[~left]
104        a, b, c, d, fc, fd = a_new, b_new, c_new, d_new, fc_new, fd_new
105
106        choose_c = fc > fd
107        kp = np.where(choose_c, c, d)
108        val = np.where(choose_c, fc, fd)
109        cons = y - kp
110        return val, cons, kp
111
112
113 def run_vfi(I=I_BASE, n_iter=100, guess="B", keep_history=True):
114     grid = np.linspace(KMIN, KMAX, I)
115     w = initial_guess(guess, grid)
116     history = [w.copy()] if keep_history else None
117     policies = [] if keep_history else None
118     for _ in range(n_iter):
119         w, c, kp = bellman_step(grid, w)
120         if keep_history:
121             history.append(w.copy())
122             policies.append(c.copy())
123     if keep_history:
124         return grid, np.asarray(history), np.asarray(policies)
125     return grid, w
126
127
128 def interior_error(grid, history, threshold=INTERIOR):
129     mask = grid >= threshold
130     target = v_star(grid[mask])

```

```

131     return np.max(np.abs(history[:, mask] - target[None, :]), axis=1)
132
133
134 def descriptive_rate(errors, n0=20, n1=80):
135     n = np.arange(n0, n1 + 1)
136     slope, intercept = np.polyfit(n, np.log(errors[n0:n1 + 1]), 1)
137     return slope, math.exp(slope), intercept
138
139
140 def howard_sweeps(guess, tol=1.0, m=20, max_sweeps=200):
141     grid = np.linspace(KMIN, KMAX, I_BASE)
142     mask = grid >= INTERIOR
143     w = initial_guess(guess, grid)
144     target = v_star(grid)
145     errors = []
146     sweeps = 0
147     while sweeps < max_sweeps:
148         w, c, kp = bellman_step(grid, w)
149         sweeps += 1
150         err = float(np.max(np.abs(w[mask] - target[mask])))
151         errors.append(err)
152         if err < tol:
153             return sweeps, errors
154         for _ in range(m):
155             w = np.log(c) + BETA * np.interp(kp, grid, w)
156             sweeps += 1
157             err = float(np.max(np.abs(w[mask] - target[mask])))
158             errors.append(err)
159             if err < tol:
160                 return sweeps, errors
161         raise RuntimeError("Howard iteration did not reach tolerance")
162
163
164 def savefig(name):
165     plt.tight_layout()
166     plt.savefig(OUT / name, bbox_inches="tight")
167     plt.close()
168
169
170 def make_figures_and_tables():
171     grid, hA, pA = run_vfi(n_iter=100, guess="A")
172     _, hB, pB = run_vfi(n_iter=100, guess="B")
173     eA = interior_error(grid, hA)
174     eB = interior_error(grid, hB)
175     mask = grid >= INTERIOR
176
177     # Figure 1: interpolation demo
178     x = np.linspace(0, 1, 400)
179     xg = np.linspace(0, 1, 6)
180     yg = np.sin(2 * np.pi * xg)
181     plt.figure(figsize=(6.3, 4.2))
182     plt.plot(x, np.sin(2 * np.pi * x), label=r"$f(x)=\sin(2\pi x)$")
183     plt.plot(x, np.interp(x, xg, yg), label="piecewise-linear interpolant")
184     plt.scatter(xg, yg, s=20)
185     plt.xlabel("x")
186     plt.ylabel("f(x)")
187     plt.legend()
188     savefig("fig_interp.pdf")
189

```

```

190 # Figure 2: fixed point illustration
191 x = np.linspace(0, 10, 300)
192 T = 2.3 + 7.2 * (1 - np.exp(-x / 3.2))
193 plt.figure(figsize=(5.7, 4.4))
194 plt.plot(x, T, label=r"$T(x)$")
195 plt.plot(x, x, label=r"$45^\circ$ line")
196 idx = np.argmin(np.abs(T - x))
197 plt.scatter([x[idx]], [T[idx]], s=28)
198 plt.xlabel("x")
199 plt.ylabel("T(x)")
200 plt.legend()
201 savefig("fig_fixedpoint.pdf")
202
203 # Iterates A/B
204 for name, H, maxn in [("A", hA, 75), ("B", hB, 60)]:
205     plt.figure(figsize=(7.0, 4.7))
206     indices = np.unique(np.round(np.linspace(0, maxn, 24)).astype(int))
207     for n in indices:
208         plt.plot(grid, H[n], linewidth=0.8, alpha=0.45)
209         plt.plot(grid, v_star(grid), linewidth=2.2, label=r"analytic $v^*$")
210         plt.xlim(0, 2)
211         plt.xlabel("capital k")
212         plt.ylabel("value")
213         plt.title(f"Fitted-VFI iterates from guess $w_{\{name\}}$")
214         plt.legend()
215         savefig(f"fig_iterates_{name}.pdf")
216
217 # analytic value function
218 plt.figure(figsize=(6.0, 4.3))
219 plt.plot(grid, v_star(grid))
220 plt.xlim(0, 2)
221 plt.xlabel("capital k")
222 plt.ylabel(r"$v^*(k)$")
223 savefig("fig_value_function.pdf")
224
225 # convergence + initial guesses in one figure (two panels)
226 fig, axes = plt.subplots(1, 2, figsize=(10.4, 4.0))
227 n = np.arange(1, 81)
228 axes[0].semilogy(n, eA[1:81], label=r"$w_A$")
229 axes[0].semilogy(n, eB[1:81], label=r"$w_B$")
230 axes[0].set_xlabel("iteration n")
231 axes[0].set_ylabel(r"interior $\|w_n - v^*\|_\infty$")
232 axes[0].legend()
233 axes[1].plot(grid, hA[0], label=r"$w_A$")
234 axes[1].plot(grid, hB[0], label=r"$w_B$")
235 axes[1].plot(grid, v_star(grid), label=r"$v^*$")
236 axes[1].set_xlim(0, 2)
237 axes[1].set_xlabel("capital k")
238 axes[1].set_ylabel("value")
239 axes[1].legend()
240 fig.tight_layout()
241 fig.savefig(OUT / "vfi_convergence.pdf", bbox_inches="tight")
242 plt.close(fig)
243
244 # Grid refinement
245 grid_floors = {}
246 plt.figure(figsize=(7.0, 4.7))
247 for I in [75, 150, 300, 600]:
248     g, H, _ = run_vfi(I=I, n_iter=300, guess="B")

```

```

249     err = interior_error(g, H)
250     grid_floors[I] = float(err[-1])
251     plt.semilogy(np.arange(1, 301), err[1:], label=f"I={I}")
252 plt.xlabel("iteration n")
253 plt.ylabel(r"$\log\|w_n - v^*\|_\infty$")
254 plt.legend()
255 savefig("fig_grid_refinement.pdf")
256
257 # Descriptive log-linear rate fit
258 slopeA, rateA, intA = descriptive_rate(eA)
259 slopeB, rateB, intB = descriptive_rate(eB)
260 nfit = np.arange(20, 81)
261 plt.figure(figsize=(6.8, 4.6))
262 plt.plot(nfit, np.log(eB[20:81]), "o", markersize=3, label="observed log error")
263 plt.plot(nfit, intB + slopeB * nfit, label="linear fit, n=20,...,80")
264 plt.xlabel("iteration n")
265 plt.ylabel(r"$\log\|w_n - v^*\|_\infty$")
266 plt.legend()
267 savefig("fig_rate_regression.pdf")
268
269 # Policy vs value
270 ctarget = c_star(grid)
271 peA = np.array([np.max(np.abs(p[mask] - ctarget[mask])) for p in pA])
272 peB = np.array([np.max(np.abs(p[mask] - ctarget[mask])) for p in pB])
273 fig, axes = plt.subplots(1, 2, figsize=(10.4, 4.0))
274 for ax, name, ev, ep in [(axes[0], "A", eA, peA), (axes[1], "B", eB, peB)]:
275     nn = np.arange(1, 81)
276     ax.semilogy(nn, ev[1:81], label="value error")
277     ax.semilogy(nn, ep[:80], label="policy error")
278     ax.set_xlabel("iteration n")
279     ax.set_ylabel("sup-norm error")
280     ax.set_title(f"Guess $w_{\{name\}}$")
281     ax.legend()
282 fig.tight_layout()
283 fig.savefig(OUT / "fig_policy_vs_value.pdf", bbox_inches="tight")
284 plt.close(fig)
285
286 # Howard comparison
287 plainA = int(np.where(eA < 1.0)[0][0])
288 plainB = int(np.where(eB < 1.0)[0][0])
289 howA, _ = howard_sweeps("A")
290 howB, _ = howard_sweeps("B")
291 labels = [r"$w_A$", r"$w_B$"]
292 xloc = np.arange(2)
293 width = 0.34
294 plt.figure(figsize=(6.1, 4.3))
295 plt.bar(xloc - width/2, [plainA, plainB], width, label="plain VFI")
296 plt.bar(xloc + width/2, [howA, howB], width, label="partial Howard")
297 plt.xticks(xloc, labels)
298 plt.ylabel("grid sweeps to error < 1")
299 plt.legend()
300 savefig("fig_howard_comparison.pdf")
301
302 # Expanded guess sensitivity
303 guess_rows = []
304 for name in "ABCDEF":
305     g, H, _ = run_vfi(n_iter=120, guess=name)
306     err = interior_error(g, H)

```

```

307     d0 = float(err[0])
308     indices = np.where(err < 1.0)[0]
309     it = int(indices[0]) if len(indices) else None
310     if d0 < 1.0:
311         benchmark = 0
312     else:
313         benchmark = int(math.ceil(math.log(1.0 / d0) / math.log(BETA)))
314     ratio = None if benchmark == 0 else it / benchmark
315     guess_rows.append({"guess": name, "d0": d0, "iterations": it,
316                       "geometric_benchmark": benchmark, "ratio": ratio})
317 ordered = sorted(guess_rows, key=lambda r: r["d0"])
318 plt.figure(figsize=(6.7, 4.5))
319 plt.plot([r["d0"] for r in ordered], [r["iterations"] for r in ordered], "o-")
320 for r in ordered:
321     plt.annotate(r["guess"], (r["d0"], r["iterations"]), xytext=(4,4),
322                 textcoords="offset points")
323 plt.xlabel(r"initial interior distance $d_0$")
324 plt.ylabel("iterations to error < 1")
325 savefig("fig_initial_guess_sensitivity.pdf")
326
327 # Boundary discretization illustration: full-grid vs interior errors
328 fullA = np.max(np.abs(hA - v_star(grid)[None, :]), axis=1)
329 fullB = np.max(np.abs(hB - v_star(grid)[None, :]), axis=1)
330 plt.figure(figsize=(7.0, 4.6))
331 nn = np.arange(0, 101)
332 plt.semilogy(nn, fullA, label=r"full grid, $w_A$")
333 plt.semilogy(nn, fullB, label=r"full grid, $w_B$")
334 plt.semilogy(nn, eA, linestyle="--", label=r"interior, $w_A$")
335 plt.semilogy(nn, eB, linestyle="--", label=r"interior, $w_B$")
336 plt.xlabel("iteration n")
337 plt.ylabel("sup-norm error")
338 plt.legend()
339 savefig("fig_boundary_error.pdf")
340
341 # Summary data
342 def first_below(e, tol):
343     ix = np.where(e < tol)[0]
344     return int(ix[0]) if len(ix) else None
345
346 rate_windows = {}
347 for n0, n1 in [(20, 50), (20, 80), (30, 70), (40, 80)]:
348     sa, ra, _ = descriptive_rate(eA, n0, n1)
349     sb, rb, _ = descriptive_rate(eB, n0, n1)
350     rate_windows[f"{n0}-{n1}"] = {
351         "A_slope": sa, "A_rate": ra, "B_slope": sb, "B_rate": rb
352     }
353
354 results = {
355     "parameters": {"A": A, "alpha": ALPHA, "beta": BETA,
356                   "kmin": KMIN, "kmax": KMAX, "I": I_BASE,
357                   "interior_threshold": INTERIOR},
358     "analytic": {"c1": C1, "c2": C2},
359     "headline": {
360         "A_iter_error_lt_1": first_below(eA, 1.0),
361         "B_iter_error_lt_1": first_below(eB, 1.0),
362         "A_iter_error_lt_0_3": first_below(eA, 0.3),
363         "B_iter_error_lt_0_3": first_below(eB, 0.3),
364         "A_error_n60": float(eA[60]), "B_error_n60": float(eB[60]),
365         "A_error_n75": float(eA[75]), "B_error_n75": float(eB[75]),

```

```

365         "A_rate_20_80": rateA, "B_rate_20_80": rateB,
366         "A_slope_20_80": slopeA, "B_slope_20_80": slopeB,
367         "policy_floor_A_n100": float(peA[99]),
368         "policy_floor_B_n100": float(peB[99]),
369         "howard_sweeps_A": howA, "howard_sweeps_B": howB,
370         "plain_sweeps_A": plainA, "plain_sweeps_B": plainB,
371     },
372     "grid_floors": grid_floors,
373     "rate_windows": rate_windows,
374     "expanded_guesses": guess_rows,
375     "boundary": {
376         "full_error_A_n100": float(fullA[100]),
377         "full_error_B_n100": float(fullB[100]),
378         "interior_error_A_n100": float(eA[100]),
379         "interior_error_B_n100": float(eB[100])
380     }
381 }
382 with open(Path(__file__).resolve().parent / "numerical_results.json", "w",
383           encoding="utf-8") as fh:
384     json.dump(results, fh, indent=2)
385     return results
386
387 if __name__ == "__main__":
388     result = make_figures_and_tables()
389     print(json.dumps(result, indent=2))

```

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- [SolowAuditedPackage.zip](#)