Neoclassical Growth Model: Closed-Form and Numerical Methods

1 Model Setup

We consider the neoclassical growth model in two forms: discrete time and continuous time. In both cases, there is a single good, capital k, consumption c, and production $y = f(k) = k^\alpha$ with $\alpha \in (0,1)$. Capital depreciates at rate $\delta \in [0,1]$. The household has CRRA utility

$$u(c) = \frac{c^{1-\gamma}}{1-\gamma}, \quad \gamma > 0,\ \gamma \neq 1; \qquad u(c) = \ln c \quad \text{if } \gamma = 1. \tag{1}$$

1.1 Discrete Time

The resource constraint is

$$c_t + k_{t+1} = (1-\delta)k_t + k_t^\alpha. \tag{2}$$

The household maximizes $\sum_{t=0}^{\infty} \beta^t u(c_t)$ subject to this constraint and k_0 given.

1.2 Continuous Time

Capital evolves as $\dot{k} = f(k) - \delta k - c$. The household maximizes $\int_0^\infty e^{-\rho t} u(c(t))\, dt$.

2 (i) Closed-Form: First-Order Conditions

2.1 Discrete time: Euler equation

The Lagrangian is

$$\mathcal{L} = \sum_{t=0}^{\infty} \beta^t \left\{ u(c_t) + \lambda_t \big[(1-\delta)k_t + k_t^\alpha - c_t - k_{t+1} \big] \right\}.$$

First-order conditions (with respect to c_t and k_{t+1}) give

$$u'(c_t) = \lambda_t, \tag{3}$$

$$\lambda_t = \beta \lambda_{t+1} \big[1 - \delta + \alpha k_{t+1}^{\alpha-1} \big]. \tag{4}$$

Hence the Euler equation:

$$\boxed{u'(c_t) = \beta\, u'(c_{t+1}) \big[1 - \delta + \alpha k_{t+1}^{\alpha-1} \big].} \tag{5}$$

For CRRA, $u'(c) = c^{-\gamma}$, so

$$\frac{c_{t+1}^\gamma}{c_t^\gamma} = \beta \big[1 - \delta + \alpha k_{t+1}^{\alpha-1} \big]. \tag{6}$$

2.2 Steady state (closed form)

In steady state, $k_{t+1} = k_t = k^*$ and $c_{t+1} = c_t = c^*$. From the Euler equation,

$$1 = \beta\big[1 - \delta + \alpha(k^*)^{\alpha-1}\big],$$

so

$$\boxed{(k^*)^{\alpha-1} = \frac{1/\beta - 1 + \delta}{\alpha} \quad \Rightarrow \quad k^* = \left(\frac{\alpha}{1/\beta - 1 + \delta}\right)^{\frac{1}{1-\alpha}}.} \tag{7}$$

From the resource constraint at steady state, $c^* + k^* = (1-\delta)k^* + (k^*)^\alpha$, so

$$\boxed{c^* = (k^*)^\alpha - \delta k^*.} \tag{8}$$

2.3 Continuous time: FOC and steady state

The Hamiltonian (current-value) is $H = u(c) + \mu[f(k) - \delta k - c]$. The FOCs give $u'(c) = \mu$ and $\dot{\mu}/\mu = \rho - (f'(k) - \delta)$. So

$$\boxed{\frac{\dot{c}}{c} = \frac{1}{\gamma}\big[f'(k) - \delta - \rho\big] = \frac{1}{\gamma}\big[\alpha k^{\alpha-1} - \delta - \rho\big].} \tag{9}$$

Steady state $\dot{k} = \dot{c} = 0$ implies $f'(k^*) = \delta + \rho$, i.e. $\alpha(k^*)^{\alpha-1} = \delta + \rho$, so

$$\boxed{k^* = \left(\frac{\alpha}{\delta + \rho}\right)^{\frac{1}{1-\alpha}}, \qquad c^* = (k^*)^\alpha - \delta k^*.} \tag{10}$$

2.4 No closed-form policy function in general

The Euler equation (discrete or continuous) and the resource constraint define the equilibrium, but they do *not* yield an explicit formula for $c(k)$ or $k'(k)$ as a function of k when $\gamma \neq 1$ and $\delta < 1$. So:

> *The neoclassical growth model does not admit a closed-form solution for the policy function $c(k)$ (or $k'(k)$) in the general case.*

Numerical methods (value function iteration, projection, neural networks, etc.) are used to approximate the policy.

Special case: log utility and full depreciation. If $\gamma = 1$ (log utility) and $\delta = 1$, the discrete-time model has a closed-form solution: $c_t = (1 - \beta\alpha)y_t$ and $k_{t+1} = \beta\alpha y_t$, so $c(k) = (1 - \beta\alpha)k^\alpha$ and $k'(k) = \beta\alpha k^\alpha$. This is the only common special case with an explicit policy.

3 (ii) Bellman Equations (Pen-and-Paper)

3.1 Discrete time

The Bellman equation is

$$\boxed{V(k) = \max_{k'} \left\{u\big((1-\delta)k + k^\alpha - k'\big) + \beta V(k')\right\}, \quad \text{s.t. } k' \in [k_{\min},\ (1-\delta)k + k^\alpha].} \tag{11}$$

Let $c = (1-\delta)k + k^\alpha - k'$. The first-order condition in k' is

$$-u'(c) + \beta V'(k') = 0 \quad \Rightarrow \quad u'(c) = \beta V'(k').$$

The envelope condition (differentiate the Bellman with respect to k at the optimal k') gives

$$V'(k) = u'(c)\big[1 - \delta + \alpha k^{\alpha-1}\big].$$

Combining: $V'(k) = u'(c)[1 - \delta + \alpha k^{\alpha-1}]$ and $u'(c) = \beta V'(k')$. This is the same Euler equation as before. So the Bellman formulation yields the same equilibrium conditions; we still do not get a closed-form $V(k)$ or $c(k)$.

3.2 Steady state from the Bellman

At steady state $k' = k = k^*$, the FOC gives $u'(c^*) = \beta V'(k^*)$ and the envelope gives $V'(k^*) = u'(c^*)[1 - \delta + \alpha(k^*)^{\alpha-1}]$. Substituting, $1 = \beta[1 - \delta + \alpha(k^*)^{\alpha-1}]$, which is the same steady-state condition as in (i). So k^* and c^* are again given by the closed-form expressions above.

3.3 Continuous time: HJB

The Hamilton–Jacobi–Bellman equation is

$$\boxed{\rho V(k) = \max_c \left\{ u(c) + V'(k)\big[f(k) - \delta k - c\big] \right\}.} \tag{12}$$

Maximizing in c gives $u'(c) = V'(k)$, so $c = (V'(k))^{-1/\gamma}$ (for CRRA). Substituting back into the HJB yields a nonlinear ODE in $V(k)$. This ODE has no closed-form solution for $V(k)$ or $c(k)$ in general; steady state is again $f'(k^*) = \delta + \rho$, with k^* and c^* as above.

4 (iii) Numeric Policy Function Iteration

The state space (capital k) is discretized on a grid $k_1 < \cdots < k_n$ over $[k_{\min}, k_{\max}]$. Two standard approaches are used.

Value function iteration (VFI). Start with an initial guess $V_0(k)$ (e.g. zero). Iterate the Bellman operator:

$$V_{j+1}(k_i) = \max_{k' \in \mathcal{F}(k_i)} \left\{ u\big((1-\delta)k_i + k_i^\alpha - k'\big) + \beta V_j(k') \right\}, \tag{13}$$

where $\mathcal{F}(k_i)$ is the set of feasible next-period capital values (on the grid). The maximum is taken over grid points. $V_j(k')$ for k' off the grid is obtained by interpolation (e.g. linear). Iterate until $\max_i |V_{j+1}(k_i) - V_j(k_i)| < \epsilon$. The policy $k'(k)$ is the maximizer at each grid point; $c(k) = (1-\delta)k + k^\alpha - k'(k)$.

Howard policy iteration. (1) Given a policy $k'(k)$ (e.g. from one VFI step), solve for the value function that satisfies $V(k) = u(c(k)) + \beta V(k'(k))$ (policy evaluation: a linear system or fixed-point iteration). (2) Update the policy by one-step maximization: $k'_{\text{new}}(k) = \arg\max_{k'} \{u((1-\delta)k + k^\alpha - k') + \beta V(k')\}$ (policy improvement). Repeat until the policy is unchanged. Typically converges in fewer outer iterations than VFI.

Steady state is either computed analytically from the formulas in Section (i) or found numerically as the point where $k'(k^*) = k^*$.

5 (iv) Polynomial Policy Function Iteration

The value function (or policy) is approximated by a finite linear combination of basis functions. Coefficients are chosen so that the model equations hold at a set of nodes.

Chebyshev collocation. Approximate $V(k) \approx \sum_{j=0}^{N} c_j T_j(x(k))$, where T_j are Chebyshev polynomials and $x(k) \in [-1, 1]$ is a linear map of $k \in [k_{\min}, k_{\max}]$. Choose collocation nodes (e.g. Chebyshev nodes) $k_1, \ldots, k_{N+1}$. Impose that the HJB residual is zero at these nodes: for each i,

$$\rho \widehat{V}(k_i) = u(\widehat{c}(k_i)) + \widehat{V}'(k_i)\big[f(k_i) - \delta k_i - \widehat{c}(k_i)\big],$$

where $\widehat{c}(k) = (V'(k))^{-1/\gamma}$ from the FOC. This gives $N+1$ equations in the $N+1$ coefficients. Solve (e.g. with a nonlinear solver) for the coefficients. The result is a smooth approximation to $V(k)$ and $c(k)$.

Sparse grid. In higher dimensions, a full tensor grid is expensive. A sparse grid (e.g. Smolyak) uses a hierarchical set of points and basis functions so that the number of nodes grows more slowly with dimension. In 1D, a level-ℓ sparse grid may use nested Clenshaw–Curtis points. The same collocation idea applies: set the residual to zero at the sparse grid points and solve for the coefficients.

6 (v) One-Layer Policy Function

The policy (e.g. consumption $c(k)$ or next-period capital $k'(k)$) is approximated by a *single* hidden layer:

$$c(k) \approx \psi\big(w_0 + \textstyle\sum_{j=1}^{H} w_j \, \phi(a_j k + b_j)\big),$$

where ϕ is an activation function (e.g. tanh), H is the number of hidden units, and ψ maps to positive consumption. The weights (w_j, a_j, b_j) are chosen by minimizing a loss, e.g. the squared Bellman residual or the squared Euler residual, over a sample of states k. This is a shallow network; capacity is limited compared to deep nets.

7 (vi) Multilayer Policy Function

The value function or policy is approximated by a *multi-layer* (deep) neural network. For example, $V(k) \approx \mathrm{Net}(k; \theta)$ with several hidden layers (e.g. $k \to$ Linear $\to$ tanh $\to$ Linear $\to$ tanh $\to \cdots \to$ Linear $\to V$). The optimal consumption is then obtained from the FOC: $c(k) = (V'(k))^{-1/\gamma}$, where $V'(k)$ is computed by automatic differentiation. Training minimizes a loss such as the mean squared HJB residual over sampled k. Deeper and wider networks can approximate more complex policies at the cost of more parameters and data.

8 (vii) Policy with Closed-Form Parameters and Gradients in Weights

Here the *analytical structure* of the model is fed directly into the training of the approximator. For example:

- The loss uses the *closed-form* FOC: $c = (V'(k))^{-1/\gamma}$, so consumption is not a free output of the network but is derived from the value function gradient. The HJB residual $\rho V - u(c) - V'(k)(f(k) - \delta k - c)$ is then a function of k and the network weights only.
- The loss may include terms that involve $u'(c)$, $V'(k)$, or the Euler residual, all computed from the model formulas (and autograd for $V'(k)$).
- Model parameters (α, δ, γ, ρ) appear explicitly in the loss; the optimizer updates only the weight matrix of the network so that the HJB (or Euler) equation is satisfied as well as possible.

This way the neural network is constrained by the economic first-order conditions rather than learning a black-box mapping; often this improves accuracy and interpretability.

9 Summary

- **(i) Closed-form (FOC):** Euler equation and steady-state k^*, c^* are explicit; the policy $c(k)$ is not available in closed form (except for $\gamma = 1$, $\delta = 1$ in discrete time).
- **(ii) Bellman:** Bellman/HJB equations and FOC/envelope conditions are written in closed form; they imply the same Euler and steady state. $V(k)$ and $c(k)$ have no general closed-form solution.
- **(iii) Numeric PFI:** Discretize the state on a grid; iterate the Bellman operator (VFI) or alternate policy evaluation and improvement (Howard) to get numerical $V(k)$, $c(k)$, $k'(k)$ and steady state.
- **(iv) Polynomial PFI:** Approximate $V(k)$ with Chebyshev (or other) polynomials; set the HJB residual to zero at collocation nodes (or use a sparse grid in higher dimensions). Yields smooth approximations.
- **(v) One-layer:** Approximate the policy with a single hidden layer; train by minimizing Bellman or Euler residual over states.
- **(vi) Multilayer:** Approximate $V(k)$ or $c(k)$ with a deep network; train by minimizing HJB residual; derive c from V' via FOC.
- **(vii) Closed-form in weights:** Use model FOCs and parameters explicitly in the loss (e.g. $c = (V'(k))^{-1/\gamma}$, HJB residual); optimize only the network weights so the model equations are satisfied.