Chapter 10: Computational Tools

Graduate Macroeconomics Slides

Orhan Torul

Boğaziçi University

January 23, 2026

Outline

- Introduction
- Motivation
- Approximating a Function
- Root Finding
- Optimization (1-D)
- Discretising an AR(1) Process
- Value Function Iteration
- Solving a Linear Rational Expectations (LRE) Model

Why Computational Tools?

- (Lucas, 1980) famously wrote "Our task as I see it...is to write a FORTRAN program that will accept specific economic policy rules as 'input' and will generate as 'output' statistics describing the operating characteristics of time series we care about, which are predicted to result from these policies." $\Rightarrow$ we need numerical building blocks.
- Two essential blocks in this chapter:
  1. Approximate an unknown function $f(x)$ on $[\underline{x}, \bar{x}]$.
  2. Solve $f(x) = 0$ quickly and reliably.

Problem Statement

Store (or evaluate) $f(x)$ for *any* $x \in [\underline{x}, \bar{x}]$ but keep memory/computation finite.

Two families of approximations

1. **Interpolation on a grid** – uses *local* data.
2. **Basis-function expansion** – uses *global* data.

Trade-off: local methods are robust but low-order; global methods are compact but one region's error can spill over to the rest.

Linear Interpolation

- Equally—or adaptively—spaced grid $\{x_i\}_{i=1}^n$, store $\{f(x_i)\}$.
- For $x \in [x_i, x_{i+1}]$,

$$\hat{f}(x) = f(x_i) + \frac{f(x_{i+1}) - f(x_i)}{x_{i+1} - x_i}(x - x_i).$$

- Pros: simplest; uses only neighbouring points.
- Cons: not differentiable at knots; large error if f highly curved.

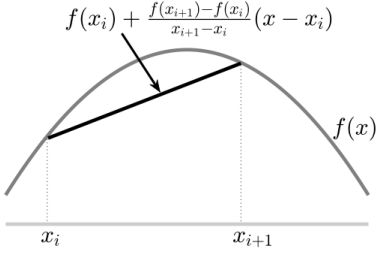

Figure 10.1 Linear interpolation

Cubic Spline vs. Chebyshev Basis

Cubic-spline interpolation

- Fit a cubic on each interval, enforce C^2 continuity.
- Needs global solve for second-derivative conditions.
- Much smaller error; differentiable policy/value functions.

Basis-function (Chebyshev) approximation

$$\hat{f}(x) = \sum_{i=1}^{n} a_i T_i(x), \qquad \text{solve } a_i \text{ via collocation or regression.}$$

- Orthogonal polynomials $\{T_i\}$ capture global shape efficiently.
- Good for high accuracy with small n; but each coefficient depends on all data $\Rightarrow$ local shock affects entire domain.

Bisection Method (bracketing)

1. Pick $a < b$ with $\operatorname{sign} f(a) \neq \operatorname{sign} f(b)$.
2. Halve the interval: $c = (a + b)/2$.
3. Keep the sub-interval where the sign changes.
4. Repeat until $|b - a| < \varepsilon$.

Guarantees convergence; no derivatives required; log-linear speed.

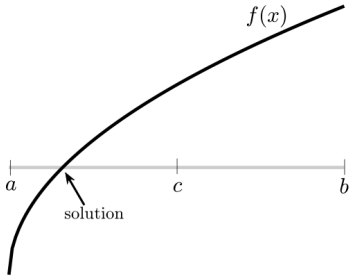

Figure 10.2 Bisection

Newton–Raphson Method

Linearise around x_k:

$$f(x) \approx f(x_k)+f'(x_k)(x-x_k) \quad \Longrightarrow \quad x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$$

Properties

- Quadratic convergence if $f'(x^\star) \neq 0$ and x_0 close.
- Requires derivative (analytic or finite-difference).
- May diverge if f poorly behaved or guess far away.

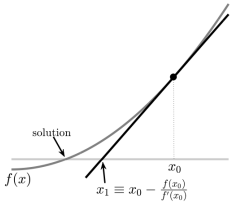

Figure 10.3 Newton step

- Choose interpolation order to balance speed vs. smoothness.
- Chebyshev + Clenshaw is standard in modern DSGE toolkits.
- Root-finding: start with safe bracket (bisection), then switch to Newton for speed.
- These primitives underpin value-function iteration, Euler-equation solvers, and perturbation methods covered in Section 10.6.

Why Do We Optimize?

- Goal: maximize a scalar objective $F(x)$ on $x \in [a, c]$.
- Grid search is fail-safe but $\mathcal{O}(n)$ evaluations.
- Two fast 1-D methods:
  1. Golden-section search (bracketing, derivative-free).
  2. Newton method for maximization (uses F', F'').

Golden-Section Search

Algorithm

1. Find $a < b < c$ with $F(a) < F(b)$ and $F(c) < F(b)$.
2. Compute $\omega = \frac{3-\sqrt{5}}{2} \approx 0.382$.
3. If $(c - b) > (b - a)$ set $x = b - \omega(c - a)$ else $x = a + \omega(c - a)$.
4. Compare $F(x)$ with middle point; drop worst sub-interval.
5. Repeat until $|c - a| < \varepsilon$.

Pros: derivative-free, guaranteed convergence, removes the fraction ω each iteration $\Rightarrow$ length after n steps = $(1 - \omega)^n(c - a)$.

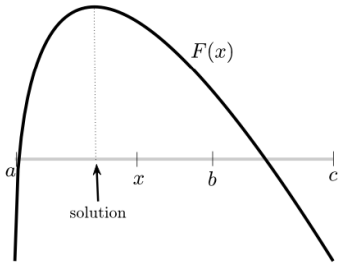

Figure 10.4 Golden-section bracket

Newton Method for Maximization

Approximate F with a quadratic around x_0:

$$F(x) \approx F(x_0) + F'(x_0)(x - x_0) + \tfrac{1}{2}F''(x_0)(x - x_0)^2.$$

Set derivative of RHS to zero:

$$\hat{x} = x_0 - \frac{F'(x_0)}{F''(x_0)}.$$

Iterate $x_{k+1} = x_k - \dfrac{F'(x_k)}{F''(x_k)}$ until $|x_{k+1} - x_k| < \varepsilon$.

Quadratic convergence if $F''(x^\star) < 0$ and start close.

Caveats: needs F', F''; may diverge if $F'' > 0$ or poor start.

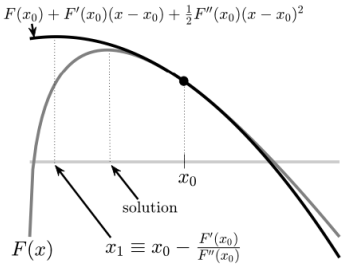

Figure 10.5 Newton step (max)

Root Finding $\leftrightarrow$ Optimization

- The FOC for maximization is $F'(x) = 0$.
- **Newton's method for max** = Newton–Raphson on $f(x) = F'(x)$.
- Golden-section search differs: never uses derivatives, removes $\frac{1}{3}$ (not $\frac{1}{2}$) of interval $\Rightarrow$ slower than bisection but keeps bracketing the *maximum* rather than a root.

Practical rule: if derivatives are cheap and a good initial guess is available $\Rightarrow$ Newton; else start with a bracketing method.

Example: Labor–Leisure Choice (static)

Problem (params $\alpha = \frac{1}{3},\ x = 0.5,\ \omega = 1,\ \gamma = 2$)

$$\max_{c,\ell}\ u(c,\ell) = \ln c - \frac{\omega}{\gamma}\ell^{\gamma} \quad \text{s.t.}\ \ c = f(\ell) + x,\ \ f(\ell) = \ell^{\alpha}.$$

First-order condition

$$\frac{\alpha \ell^{\alpha-1}}{\ell^{\alpha} + x} - \omega \ell^{\gamma-1} = 0. \tag{10.5}$$

Discretizing an AR(1) process

- Continuous AR(1):

$$z_{t+1} = \rho z_t + \varepsilon_{t+1}, \qquad \varepsilon_{t+1} \sim \mathcal{N}(0, \sigma_\varepsilon^2),\ 0 < \rho < 1.$$

- Numerical dynamic-programming packages require *finite* state spaces $\Rightarrow$ approximate with a Markov chain $(\tilde{z}_t, \Pi)$.
- Goal: match unconditional variance $\sigma_z^2 = \sigma_\varepsilon^2/(1-\rho^2)$ *and* persistence.

Tauchen (1986) Method

(Tauchen, 1986) Equally–spaced grid $\{\bar{z}^1, \ldots, \bar{z}^N\}$ with step size ω.

Steps

1. Choose coverage parameter m and set $\bar{z}^1 = -m\sigma_z,\ \bar{z}^N = +m\sigma_z,\ \omega = \dfrac{2m\sigma_z}{N-1}$.
2. Transition probabilities ($i = 1, \ldots, N$):

$$\pi_{ij} = \begin{cases} F\Big(\dfrac{\bar{z}^1 - \rho\bar{z}^i + \omega/2}{\sigma_\varepsilon}\Big) & j = 1, \\ F\Big(\dfrac{\bar{z}^j - \rho\bar{z}^i + \omega/2}{\sigma_\varepsilon}\Big) - F\Big(\dfrac{\bar{z}^j - \rho\bar{z}^i - \omega/2}{\sigma_\varepsilon}\Big) & j = 2, \ldots, N-1, \\ 1 - F\Big(\dfrac{\bar{z}^N - \rho\bar{z}^i - \omega/2}{\sigma_\varepsilon}\Big) & j = N, \end{cases}$$

where F is the standard-normal CDF.

Features: simple; accurate for moderate ρ; error grows as $\rho \to 1$.

Rouwenhorst (1995) Method

(Rouwenhorst, 1995) Designed for highly persistent processes.

Recursive construction

$$\Pi_2 = \begin{bmatrix} p & 1-p \\ 1-p & p \end{bmatrix}, \qquad \Pi_N = p \begin{bmatrix} \Pi_{N-1} & \mathbf{0} \\ \mathbf{0}' & 0 \end{bmatrix} + (1-p) \begin{bmatrix} 0 & \mathbf{0} \\ \mathbf{0}' & \Pi_{N-1} \end{bmatrix} + p \begin{bmatrix} 0 & \Pi_{N-1} \\ \mathbf{0}' & 0 \end{bmatrix} + (1-p)$$

then rescale rows to sum to 1. Grid points $\{\bar{z}^1, \ldots, \bar{z}^N\}$ equally spaced with $m = \sqrt{N-1}$, $p = (1+\rho)/2$.

Properties

- Matches mean and variance *exactly*.
- Converges weakly to a Gaussian as $N \to \infty$.
- Superior to Tauchen when $\rho \approx 1$.

Bellman Equation and VFI

Deterministic Bellman equation:

$$V(k) = \max_{k'} \; F(k, k') + \beta V(k').$$

Contraction: $TV = \max_{k'}[F(k, k') + \beta V(k')]$ satisfies $\|TV - TW\| \leq \beta \|V - W\|$.

Value-Function Iteration

1. Fix grid $\{k^1, \ldots, k^N\}$.
2. Initialise $V_0(k^j) = 0$.
3. For each k^j compute $V_{i+1}(k^j) = \max_{k'}[F(k^j, k') + \beta V_i(k')]$.
4. Stop when $\max_j |V_{i+1}(k^j) - V_i(k^j)| < \varepsilon$.

Guaranteed to converge, though slow when $\beta \to 1$.

Example 3: Cake-Eating (Deterministic)

Utility $\sum_{t=0}^{\infty} \beta^t \sqrt{c_t}$, cake stock evolves $a_{t+1} = a_t - c_t$.

Algorithm

1. Grid for $a \in [0, a_0]$.
2. Initialise $V_0(a) = 0$.
3. At each a maximise $\sqrt{a - a'} + \beta V_i(a')$ over $a' \in$ grid (or via golden-section + interpolation).

Example 4: Cake-Eating (Stochastic)

Shock $z_t \in \{z_L, z_H\}$ with transition matrix Π.
Bellman:

$$V(a, z) = \max_{a'} \sqrt{a - a' + z} + \beta \left[\pi_{zz_H} V(a', H) + \pi_{zz_L} V(a', L)\right].$$

Same VFI loop, but store and update two value vectors $V(\cdot, L)$, $V(\cdot, H)$.

Key change: natural upper grid bound rises above initial a_0 (because shocks add cake).

Linear Rational Expectations Model

- Many DSGE models have no global closed-form $\Rightarrow$ linear (or log-linear) around steady state.
- Local accuracy suffices for "small" business-cycle deviations (RBC, NK baseline).
- Software such as `Dynare` automates the steps we derive next.

$$z_{t+1} = \rho z_t + \varepsilon_{t+1}, \qquad 0 < \rho < 1,\ \varepsilon_{t+1} \sim \mathcal{N}(0, \sigma^2)$$

is our exogenous shock process (recall Section 10.4).

Predetermined vs. Jump Variables

Definitions

- **Predetermined (state) variables**: known at t – e.g. capital K_t.
- **Non-predetermined (jump) variables**: chosen within t – e.g. consumption C_t.

Goal: express future endogenous variables as *linear functions* of states and shocks.
Example context: stochastic neoclassical growth with utility

$$U = \mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t \Big[(1-\phi) \ln C_t + \phi \ln(1 - H_t) \Big],$$

resource constraint (10.9) and technology shock (10.10).

First-Order Conditions and Log-Linearisation

FOCs $\Rightarrow$

$$\frac{1}{C_t} = \beta\,\mathbb{E}_t\Big[(1+\alpha\exp z_{t+1})K_{t+1}^{\alpha-1}H_{t+1}^{1-\alpha} - \delta\Big]\frac{1}{C_{t+1}} \tag{10.13}$$

$$\frac{1-\phi}{C_t}(1-\alpha)\exp(z_t)K_t^{\alpha}H_t^{-\alpha} = \frac{\phi}{1-H_t} \tag{10.14}$$

Log-deviate around steady state ($\tilde{x}_t = \ln(X_t/\bar{X})$) and use $\mathbb{E}_t z_{t+1} = \rho z_t$ to obtain the *linear* system (10.15)–(10.17):

$$\tilde{k}_{t+1} = a_{11}\tilde{k}_t + a_{12}z_t + a_{13}c_t, \quad c_t = -\beta a_{21}\mathbb{E}_t c_{t+1} + a_{22}\tilde{k}_t + a_{23}z_t.$$

Coefficients a_{ij} collect $\bar{K}, \bar{H}, \alpha, \delta, \theta, \beta$.

Blanchard–Kahn (1980) Condition

(Blanchard and Kahn, 1980)

Scalar example with one jump variable:

$$y_t = \phi \mathbb{E}_t y_{t+1} \quad \Longrightarrow \quad y_{t+1} = \lambda y_t, \ \lambda = 1/\phi.$$

- Stability requirement $\lim_{T \to \infty} y_{t+T}$ finite.
- Uniqueness $\Longleftrightarrow$ $|\lambda| > 1$ (otherwise *indeterminacy*)

Multi-variate Rule

For linear system $B[x_{t+1}; \mathbb{E}_t y_{t+1}] = A[x_t; y_t] + Ea_t \Rightarrow$ canonical form $[x_{t+1}; \mathbb{E}_t y_{t+1}] = F[x_t; y_t] + Ga_t$.

$$\text{If } F \text{ has } m \text{ eigenvalues } |\lambda_i| > 1 \Longleftrightarrow m \text{ jump variables } (y_t)$$

then the equilibrium is *unique*. Otherwise: either explosive or indeterminate.

Guess–and–Verify (Undetermined Coefficients)

When BK guarantees uniqueness, postulate

$$y_t = A\, x_t, \qquad A \text{ constant.}$$

Insert into $y_t = \phi \mathbb{E}_t y_{t+1} + z_t \;\Rightarrow\; A = -\dfrac{1}{\lambda - \rho},\ \lambda = 1/\phi$. Technique generalises to matrix case via partitioning $H^{-1} F H = J,\ J = \operatorname{diag}(\lambda_1, \dots)$ and solving

$$\tilde{y}_t = \Lambda \rho a_t, \qquad y_t = -\tilde{H}_{22}^{-1} \tilde{H}_{21} x_t + \tilde{H}_{22}^{-1} \Lambda \rho a_t.$$

References

Blanchard, O. J. and Kahn, C. M. (1980). The solution of linear difference models under rational expectations. *Econometrica*, 48(5):1305–1311.

Lucas, R. E. (1980). Methods and problems in business cycle theory. *Journal of Money, Credit and banking*, 12(4):696–715.

Rouwenhorst, K. G. (1995). *Asset Pricing Implications of Equilibrium Business Cycle Models*. In T. F. Cooley (Ed.), Frontiers of Business Cycle Research, Princeton University Press, Princeton, NJ.

Tauchen, G. (1986). Finite state markov-chain approximations to univariate and vector autoregressions. *Economics Letters*, 20:177–181.

Thank you!

Questions or comments?

orhan.torul@bogazici.edu.tr