# Computational Economics

## Lecture 4: Practical Dynamic Programming

Etienne Briand

Concordia University

Winter 2026

## Motivation

In the previous lectures, we established a number of powerful results about dynamic programming.

Most importantly, solving the FE delivers the solution to a continuum of corresponding SPs (i.e., one problem for each initial state $s_0$) in both deterministic and stochastic environments.

However, with very few exceptions, the FE cannot be solved analytically. Thus, numerical methods are required.

## Overview

In this lecture, we will go over some of the main computational methods used to solve the Bellman equation.

We begin with value function iteration, which is a direct application of the mathematical results presented in the previous lectures.

We then introduce alternative methods that (i) reduce computation time, and/or (ii) improve numerical accuracy.

## Value Function Iteration

## VFI

Value function iteration directly exploits the fixed-point properties of the Bellman operator.

It constructs a sequence of value functions and associated policy rules by iterating on the Bellman equation.

Starting from an initial guess $V_0$, the iteration proceeds until convergence according to a chosen distance criterion:

$$V_{j+1}(s) = \max_{x \in \Gamma(s)} \left\{ F(x,s) + \beta E\left[ V_j(s') \right] \right\} \tag{1}$$

where the state evolves according to the transition function $s' = \psi(x, \cdot)$.

## VFI (cont'd)

In practice, VFI is typically implemented by discretizing the $n$-dimensional state space $S$ on a grid and proceeding as follows:

0. Initialize a guess $V_0$ defined on the discretized state space.
1. Given $V_j$, compute the optimal policy $g_j(s)$ at each grid point by solving Equation (1).
2. Construct the updated value function $V_{j+1}$ using $g_j$ and $V_j$.
3. Compute a distance measure $d$ between $V_{j+1}$ and $V_j$.
4. If $d > \epsilon$, return to step 1 using $V_{j+1}$ as the updated guess. Otherwise, convergence is achieved and the pair $(V_j, g_j)$ is the solution.

## Policy Function Iteration

## PFI

In practice, VFI may converge slowly. A well-known acceleration method due to Howard is policy function iteration.

The key difference relative to VFI is that, at each iteration, the value function is computed while holding the policy fixed.

0. Initialize a policy function $g_0$ on the discretized state space.
1. Given $g_j$, compute the associated value function $V_j$ by:
    * iterating on the Bellman equation with the policy fixed, or
    * solving the implied system of linear equations.
2. Update the policy by solving for $g_{j+1}(s)$ that maximizes Equation (1) given $V_j$.
3. Compute a distance measure $d$ between $g_{j+1}$ and $g_j$.
4. If $d > \epsilon$, return to step 1 and use $g_{j+1}$ as the updated guess. Otherwise, convergence is achieved.

## Euler Equation Methods

## Time Iteration

While VFI and PFI are methods with broad applicability, we can often solve for problems with higher efficiency by explointing their structures (especially concavity and smoothness properties).

We know that the optimal policy that maximizes Equation (1) is unique, continuous and strictly increasing, and that the value function is concave and continuously differentiable
(we previouly used those properties to characterize the optimality conditions).

## Time Iteration (cont'd)

Analogous to the Bellman operator, the Coleman operator $K$ acts on policy functions. It can be used to solve for the optimal policy directly from the FOC.

$$Ku'(c) = \beta E \left\{ u'[g(c)] r_{t+1} \right\} \tag{2}$$

where $r_{t+1}$ denotes the return on an additional unit of saving chosen at time $t$.

The Coleman operator characterizes the optimal policy $g^*(c)$ to be used today, given that future consumption is determined by the policy $g(c)$.

## Time Iteration (cont'd)

The fixed point of the Coleman operator coincides with the solution to the FE. In theory, we should be indifferent between solving for the optimal policy using VFI or time iteration.

In practice, however, solving for the optimal policy using the Coleman operator is both faster and more accurate

## Time Iteration (cont'd)

Time iteration can be further improved using the endogenous grid method (EGM).

Implementing EGM only requires that $u'(\cdot)$ be invertible, which is typically satisfied. As a result, finding $g(c)$ is straightforward.

Given $g(c)$, EGM constructs an endogenous grid for the state variable. The policy function is then interpolated back onto the fixed state grid, and the procedure is repeated until convergence.

The key advantage of EGM is that the optimal policy is obtained "for free," whereas standard time iteration requires a numerical optimizer.

## Functional Approximations

## Functional Approximation

The methods discussed so far relied on discretizing the state space and exploiting the contraction properties of the Bellman operator.

An alternative approach is to approximate the value function directly using a parametric function, rather than computing it pointwise on a grid.

This shifts the problem from solving for function values to solving for a finite set of parameters.

## Collocation Method

The collocation method assumes that the value function can be approximated as

$$V(s) \approx \hat{V}(s) = \sum_{j=1}^{n} c_j, \phi_j(s), \tag{3}$$

where:

- $\phi_j(s)_{j=1}^{n}$ are linearly independent basis functions,
- $c_1, \ldots, c_n$ are unknown coefficients.

A common choice for $\phi_j$ are Chebyshev polynomials evaluated at Chebyshev nodes, due to their strong approximation properties.

## Collocation Method (cont'd)

Substituting $\hat{V}(s)$ into Equation (1) yields

$$
\begin{aligned}
0 &= -\sum_{j=1}^{n} c_j\,\phi_j(s) + \max_{x\in\Gamma(s)}\{F(x,s) + \beta E[V_j(s')]\} \\
&= -\sum_{j=1}^{n} c_j\,\phi_j(s) + \max_{x\in\Gamma(s)}\{F(x,s) + \beta E\Big[\sum_{j=1}^{n} c_j\,\phi_j(s')\Big]\} \qquad (4)
\end{aligned}
$$

This condition is imposed exactly at $n$ collocation nodes, generating a system of $n$ nonlinear equations in $n$ unknowns.

Standard root-finding algorithms (e.g. Newton or quasi-Newton methods) can be used to solve for the coefficient vector $c = \{c_1, c_2, ..., c_n\}$.

## ANNs

The same logic underlying collocation applies more generally to flexible function approximators.

Artificial Neural Networks (ANNs) are a natural alternative because they:

- approximate a very rich class of functions,
- scale well to high-dimensional state spaces,
- can be trained efficiently using gradient-based algorithms.

We approximate the value function as

$$V(s) \approx \hat{V}(s) = ANN(s \mid \theta), \tag{5}$$

where $\theta$ is a vector of parameters.

## ANNs (cont'd)

Substituting the ANN approximation into Equation (1) implies

$$\begin{aligned} 0 &= -ANN(s|\theta) + \max_{x\in\Gamma(s)} \{F(x,s) + \beta E[V_j(s')]\} \\ &= -ANN(s|\theta) + \max_{x\in\Gamma(s)} \{F(x,s) + \beta E\left[ANN(s'|\theta)\right]\} \end{aligned} \tag{6}$$

The parameter vector $\theta$ is chosen to minimize the mean squared Bellman residual implied by Equation (6), using points sampled from the state space $S$.