Solving Dynamic Models Numerically

Lecture 2

Stephen J. Terry
BU Macro & NBER

June 2023

We've Covered the Basics, But...

There's a whole world of stuff left to learn out there, including information on

- improving the speed of the model solution,
- improving the speed and accuracy of simulation,
- handling equilibrium relationships,
- and adding states and complexity to your model.

Let's explore a bit!

Roadmap

Tweaking VFI: More Speed, Please

Avoiding Monte Carlo: Nonstochastic Simulation

Equilibrium: Inner and Outer Loops

The Curse of Dimensionality

Fun with MATLAB!

Recall the VFI Algorithm

Solving the Bellman equation

$$V(s,x) = \max_{s' \in \Gamma(s,x)} \left[R(s,x,s') + \beta \mathbb{E}_{x'|x} V(s',x') \right].$$

You start the VFI algorithm with $R(s,x,s')$, $F(x'|x)$, $\Gamma(s,x)$, and $0 < \beta < 1$, as well as a guess $V_{(1)}$ and a specified solution tolerance $\varepsilon_{tol} > 0$. For each step of VFI $n = 1, 2, ...,$ do

1. Compute $V_{(n+1)}$ by solving for optimal s' at each (s,x) via

$$V_{(n+1)}(s,x) = \max_{s' \in \Gamma(s,x)} \left[R(s,x,s') + \beta \mathbb{E}_{x'|x} V_{(n)}(s',x') \right].$$

2. Compute the error

$$||V_{(n+1)} - V_{(n)}|| = \max_{s,x} |V_{(n+1)}(s,x) - V_{(n)}(s,x)|$$

3. If $||V_{(n+1)} - V_{(n)}|| < \varepsilon_{tol}$, exit. If $||V_{(n+1)} - V_{(n)}|| \geq \varepsilon_{tol}$, go to next iteration by jumping back to Step 1 with $n = n + 1$.

VFI Tweak: Howard Improvement

VFI is robust and transparent but slow.

- Must compute optimal policies at each step n, but must wait for the **level** of $V_{(n)}$ to converge.
- Equivalent to waiting for a geometric series with ratio β to converge - a long time for high β!

Howard improvement attacks this issue directly. Why not sometimes skip the optimization step within VFI?

- Start with the usual VFI step, i.e., optimization of policies given a guess for $V_{(n)}$.
- Apply the Bellman equation in a number of Howard improvement steps **without optimization** to obtain the next guess $V_{(n+1)}$.

If you choose a large number of Howard steps, this algorithm becomes essentially equivalent to **policy iteration**.

VFI Tweak: Howard Improvement

Start with $R(s,x,s')$, $F(x'|x)$, $\Gamma(s,x)$, and $0<\beta<1$, as well as a guess $V_{(1)}$ and a specified solution tolerance $\varepsilon_{tol}>0$. For each $n=1,2,...,$ do

1. Compute $V_{(n+1)}$.
   - A Solve for optimal policies $s'_{(n)}$ via

$$s'_{(n)}(s,x)=\arg\max_{s'\in\Gamma(s,x)}\left[R(s,x,s')+\beta\mathbb{E}_{x'|x}V_{(n)}(s',x')\right].$$

   - B Set $\tilde{V}^{(1)}_{(n)}=V_{(n)}$. For each Howard improvement step $h=1,...,H-1$, iterate the Bellman equation without optimization via

$$\tilde{V}^{(h+1)}_{(n)}(s,x)=R(s,x,s'_{(n)}(s,x))+\beta\mathbb{E}_{x'|x}\tilde{V}^{(h)}_{(n)}\left(s'_{(n)}(s,x),x'\right).$$

   - C Set $V_{(n+1)}=\tilde{V}^{(H)}_{(n)}$.
2. Compute the error

$$||V_{(n+1)}-V_{(n)}||=\max_{s,x}|V_{(n+1)}(s,x)-V_{(n)}(s,x)|$$

3. If $||V_{(n+1)}-V_{(n)}||<\varepsilon_{tol}$, exit. If $||V_{(n+1)}-V_{(n)}||\geq\varepsilon_{tol}$, go to next iteration by jumping back to Step 1 with $n=n+1$.

VFI Tweak: MQP Bounds

Again, note that the VFI algorithm is slow because you must wait for a geometric series to converge in **levels**. Exploit the useful mathematical result that within VFI step n

$$V_{(n)}(s,x) + \underline{b}_n \leq V(s,x) \leq V_{(n)}(s,x) + \bar{b}_n$$

where

$$\underline{b}_n = \frac{\beta}{1-\beta} \min_{(s,x)} \left(V_{(n)}(s,x) - V_{(n-1)}(s,x) \right)$$

$$\bar{b}_n = \frac{\beta}{1-\beta} \max_{(s,x)} \left(V_{(n)}(s,x) - V_{(n-1)}(s,x) \right)$$

are known as the **MacQueen-Porteus (MQP) Bounds**. A reasonable update of the value function would be

$$V_{(n)}(s,x) + \frac{\underline{b}_n + \bar{b}_n}{2},$$

a choice which is closer to the level of $V(s,x)$.

VFI Tweak: MQP Bounds

Start with $R(s,x,s')$, $F(x'|x)$, $\Gamma(s,x)$, and $0<\beta<1$, as well as a guess $V_{(1)}$ and a specified solution tolerance $\varepsilon_{tol}>0$. For each step $n=1,2,...,$ do

1. Compute $V_{(n+1)}$.
   - A Compute $\tilde{V}_{(n+1)}$ by solving for optimal s' at each (s,x) via

$$\tilde{V}_{(n+1)} = \max_{s'\in\Gamma(s,x)} \left[R(s,x,s') + \beta \mathbb{E}_{x'|x} V_{(n)}(s',x')\right].$$

   - B Compute the two constants

$$\underline{b}_{n+1} = \frac{\beta}{1-\beta} \min_{(s,x)} \left(\tilde{V}_{(n+1)}(s,x) - V_{(n)}(s,x)\right)$$

$$\bar{b}_{n+1} = \frac{\beta}{1-\beta} \max_{(s,x)} \left(\tilde{V}_{(n+1)}(s,x) - V_{(n)}(s,x)\right).$$

   - C Set $V_{(n+1)} = \tilde{V}_{(n+1)} + \frac{\underline{b}_{n+1}+\bar{b}_{n+1}}{2}$.
2. Compute the error

$$||V_{(n+1)} - V_{(n)}|| = \max_{s,x} |V_{(n+1)}(s,x) - V_{(n)}(s,x)|$$

3. If $||V_{(n+1)} - V_{(n)}|| < \varepsilon_{tol}$, exit. If $||V_{(n+1)} - V_{(n)}|| \geq \varepsilon_{tol}$, go to next iteration by jumping back to Step 1 with $n = n+1$.

Roadmap

Tweaking VFI: More Speed, Please

Avoiding Monte Carlo: Nonstochastic Simulation

Equilibrium: Inner and Outer Loops

The Curse of Dimensionality

Fun with MATLAB!

Monaco is Pretty, But...

Part 1 covered a simulation approach based on

- Drawing a large number of random shocks x_t.
- Plugging them into the model solution to simulate s_t.

This can be used to form an approximation to moments via **Monte Carlo integration** which follows the formula

$$\mathbb{E}(g(s,x)) \approx \frac{1}{T}\sum_{t=1}^{T} g(s_t, x_t).$$

Two practical issues present themselves:

- Simulating the model with a large number of random shocks can be time consuming.
- The law of large numbers doesn't perfectly apply with finite T. Sample averages may be noisy (later see SMM vs GMM).

Stationary Distributions

Let's review info on a Markov chain's **stationary** or **ergodic** distribution.

- Such a distribution exists for well behaved Markov chains.
- If you simulate a bunch of firms i for a bunch of periods t to get a panel x_{it}, you would eventually observe the stationary distribution
  - in the cross-section across firms i in some period t, and
  - across a large number of periods t for a single firm i.

A few observations make stationary distributions useful for us.

- The combined vector $(s', x')|(s, x)$ is a Markov chain with its own **joint** stationary distribution in well behaved models.
- Monte Carlo means converge to stationary distribution means, i.e., the object of interest.
- The stationary distribution of $(s', x')|(s, x)$ is often faster to compute than a large panel of simulated data.

Nonstochastic Simulation

Young (2010, JEDC) proposes a method for computing the stationary distribution $\mu(\bar{s}_i, \bar{x}_j)$ of a discretized model with solution already in hand. Start with a guess $\mu^{(1)}$ for the distribution, which is a set of $N_s N_x$ numbers $\mu_{ij}^{(1)} \in [0,1]$ such that $\sum_{i,j} \mu_{ij}^{(1)} = 1$. For each step $n = 1, 2, ...$, do the following:

1. Compute an updated guess $\mu^{(n+1)}$.
   - A Initialize your guess for $\mu^{(n+1)}$ to a $N_s N_x \times 1$ zero vector.
   - B For each combination $(\bar{s}_i, \bar{x}_j)$, extract the optimal policy $s'(\bar{s}_i, \bar{x}_j) \in \{\bar{s}_1, ..., \bar{s}_{N_s}\}$.
   - C For each $k = 1, ..., N_x$, add the weight $\mu^{(n)}(\bar{s}_i, \bar{x}_j)\Pi^x_{jk}$ to the entry in $\mu^{(n+1)}(s'(\bar{s}_i, \bar{x}_j), \bar{x}_k)$, where Π^x is x's transition matrix.
2. Compute the error

$$||\mu^{(n+1)} - \mu^{(n)}|| = \max_{i,j} \left| \mu^{(n+1)}(\bar{s}_i, \bar{x}_j) - \mu^{(n)}(\bar{s}_i, \bar{x}_j) \right|$$

3. If $||\mu^{(n+1)} - \mu^{(n)}|| < \varepsilon_{tol}$, then exit. If $||\mu^{(n+1)} - \mu^{(n)}|| \geq \varepsilon_{tol}$, then set $n = n + 1$ and return to Step 1.

You can then compute moments using the stationary distribution μ via

$$\mathbb{E}(g(s,x)) = \sum_{i,j} g(\bar{s}_i, \bar{x}_j)\mu(\bar{s}_i, \bar{x}_j)$$

Some Notes and Caveats on Nonstochastic Simulation

- Monte Carlo simulation may still be required for some flavors of structural estimation.
- Monte Carlo simulation is also still useful for examining sample paths across short time intervals, while the stationary distribution is a long-run or cross-sectional concept.
- One could compute stationary distributions differently using various linear algebra tricks applied to an appropriately constructed transition matrix for the joint Markov chain $(s', x')|(s, x)$, although the nonstochastic simulation approach is often faster.

Roadmap

Tweaking VFI: More Speed, Please

Avoiding Monte Carlo: Nonstochastic Simulation

Equilibrium: Inner and Outer Loops

The Curse of Dimensionality

Fun with MATLAB!

Equilibrium Interactions

Sometimes the baseline Bellman equation

$$V(s,x) = \max_{s' \in \Gamma(s,x)} \left[R(s,x,s') + \beta \mathbb{E}_{x'|x} V(s',x') \right]$$

isn't computable in isolation but interacts with choices or price determinations made by other agents with other optimization problems. Such **equilibrium** interactions are tricky.

Risky Debt: A Classic Corporate Finance Example

- A firm with debt b faces productivity shocks z and chooses debt issuance $b'(b,z,...)$ and default $df(b,z,...) \in \{0,1\}$ in order to solve a Bellman equation, taking as given a debt price schedule $q(z,b',...)$.
- The debt price schedule $q(z,b',...)$ comes from a zero-profit condition from a risk-neutral lender, which sets the expected return from lending – given firm default choices and recovery outcomes – equal to the risk-free rate.

The tricky issue is that you can't compute firm value $V(b,z,...)$ without knowing the debt price schedule $q(z,b',...)$, and vice-versa.

Inner Loop, Outer Loop

In the risky debt case, an "inner loop, outer loop" approach works:

1. **Outer Loop**: Guess a debt price schedule $q(z, b', ...)$.
    A. **Inner Loop**: Given $q(z, b', ...)$, use some flavor of VFI to solve for firm decisions $b'(b, z, ...)$ and $df(b, z, ...)$.
    B. This of course involves a loop or iteration over value functions, etc...
2. Check whether debt prices are correct given firm default decisions.
3. If so, exit outer loop.
4. If not, update debt prices based on last inner loop's default choices and return to Step 1.

Key Insight

For many problems in corporate finance, macroeconomics, and IO, this inner loop/outer loop alternation works well. But results and stability of the algorithm can vary by context. You should understand the theoretical context of your model's equilibrium to judge whether it makes sense.

Roadmap

Tweaking VFI: More Speed, Please

Avoiding Monte Carlo: Nonstochastic Simulation

Equilibrium: Inner and Outer Loops

The Curse of Dimensionality

Fun with MATLAB!

A Warning

The formulation of the Bellman equation

$$V(s,x) = \max_{s' \in \Gamma(s,x)} \left[R(s,x,s') + \beta \mathbb{E}_{x'|x} V(s',x') \right]$$

is quite general when you allow for the endogenous states s and the shocks x to be vectors.

But take care!

- Let s be K_s dimensional, with N_1^s, N_2^s, ..., $N_{K_s}^s$ grid points each.
- Let x be K_x dimensional, with N_1^x, N_2^x, ..., $N_{K_x}^x$ grid points each.
- The discretized Cartesian product grid for (s,x) has N points:

$$N = \prod_{k=1}^{K_s} N_k^s \prod_{k=1}^{K_x} N_k^x.$$

- That's a lot of places you'll need to solve an optimization problem, for each and every VFI step and for each evaluation of a structural estimation objective function!

One could even call it a **curse of dimensionality**.

Some Strategies to Keep in Mind

There is no magic bullet to deal with the curse of dimensionality. But a few strategies to keep in mind include:

- Parsimonious choice of model states, shocks, and grid densities.
- Wise choice of grids, using sparse projection grid methods.
- Efficient coding:
  - Using liberal vectorization and linear algebra in some languages (MATLAB) rather than loops.
  - Using loops in other languages (C++, Fortran) rather than heavy linear algebra.
  - Parallel computing to distribute the individual optimization problems within VFI steps to different processors, perhaps in a university cluster computing environment.

By Far the Most Important Step
Demonstrate grit by abandoning MATLAB as soon as possible in order to actually finish your dissertation in finite time.

Roadmap

Tweaking VFI: More Speed, Please

Avoiding Monte Carlo: Nonstochastic Simulation

Equilibrium: Inner and Outer Loops

The Curse of Dimensionality

Fun with MATLAB