Lecture 1: Function approximation

Administrative details

- Macro PhD sequence
  - Quantitative Macro Theory (this part)
  - Consumption
- One common exam for the two parts
- Exam format:
  - One computer project
  - Text of the project distributed in February.
  - Solution discussed in an individual oral presentation with Roman Sustek and myself on Monday 10th March.

1 Introduction to the course

Roadmap for this part

- Aim of the course: learning to solve dynamic programming problems.
- Necessary tools from numerical analysis
  - Function approximation
  - Numerical integration
  - Numerical optimization (just a hint).
- Putting it all together: solving the Bellman equation

  Two methods:
  - Discretized value function iteration.
  - The method of endogenous grid points (time permitting).

Readings for this lecture

1. Section 3.2 in LS (just skim it, to frame the problem)
2. The notes on "Function Approximation" by Wouter den Haan at http://tinyurl.com/5tv93vr
3. Chapter 6 (selectively) in Judd (1998)

A dynamic optimization problem

- Consider the stochastic control (sequence) problem of choosing $\{u_t, x_{t+1}\}$

$$\begin{aligned} &\mathbb{E}_0 \sum_{t=0}^{\infty} \beta^t r(x_t, u_t),\ 0 < \beta < 1 \\ &\text{s.t. } x_{t+1} = g(x_t, u_t, \epsilon_t) \\ &x_0, \epsilon_0 \text{ given.} \end{aligned}$$

  with ϵ_t i.i.d. with density function $G(\epsilon)$.

- The solution is an (infinite) sequence $\{u_t, x_{t+1}\}_{t=0}^{\infty}$ for *each* possible history of the shock $\{\epsilon_t\}_{t=0}^{\infty}$.

The (equivalent) dynamic programming problem

- The dynamic programming (recursive) counterpart of the above problem is

$$V(x) = \max_u r(x, u) + \beta \mathbb{E} V(g(x, u, \epsilon))$$

- The solution is a pair of *functions* $\{u(x), V(x)\}$.

Numerical versus analytical solutions

- Computers can only deal with finite-dimensional objects. Namely

  1. Finitely large or small (rational) numbers.
  2. Finite series.

- Numerical solution are always approximate. Two sources of error.

  1. Roundoff error. Real numbers are approximated by the nearest rational number.
  2. Truncation error. Functions are approximated by finite series or other discrete representations.

     E.g. The order of approximation in a Taylor series expansion is bounded above.

Numerical solution to dynamic programming problems

In solving a functional equation like

$$V(x) = \max_u r(x, u) + \beta \mathbb{E} V(g(x, u, \epsilon))$$

we have to tackle the following general problems.

1. How to approximate the unknown functions u and V

2. How to approximate the integral in the expectation
3. How to solve the maximization step

These correspond to the following areas of numerical analysis.

1. Function approximation.
2. Numerical integration.
3. Numerical optimization.

2 Function approximation

Function approximation

To approximate a function

$$f(x)$$

when $f(x)$ is

1. known but too complex to evaluate; or
2. unknown but we have some information about it; namely
   - we know its value (and/or that of its derivatives) at some points
   - we know the system of (functional) equation it satisfies.

Information available

- Finite set of derivatives
  - Usually at one point → **local approximation methods**
  - Function is differentiable
  - e.g. Taylor method
  - usually inaccurate away from chosen point
- Set of function values → **projection methods**
  - $f_0, \cdots, f_m$ at m nodes $x_0, \cdots, x_m$
  - m is usually finite

3 Function approximation: projection methods

3.1 General specification of projection method

Projection methods

- We want to approximate a (known or unknown) function $f(x) : \mathbb{R} \to \mathbb{R}$ that solves a functional equation of the form

$$\mathcal{H}(f(x)) = 0 \text{ for } x \in X \subseteq \mathbb{R}$$

- (Linear) projection method solve the problem by specifying an approximating function

$$f^n(x;\theta) = \sum_{i=0}^{n} \theta_i \Psi_i(x)$$

- Namely, choose a basis $\{\Psi_i(x)\}_{i=0}^n$ and project $f(\cdot)$ onto the basis to find the vector $\theta = \{\theta_i\}_{i=0}^n$.

Remarks

- Very general framework/representation. It captures most problems of interest.
- In general same number of parameters as basis functions.
- Linear projection; i.e. linear combination of basis functions. Very similar to OLS.

  Theory of non-linear approximations (e.g. neural networks) is not as well developed and possibly overkill for most economics.
- We first discuss the case in which $f : \mathbb{R} \to \mathbb{R}$. Easily generalized (with some complications) later.

Algorithm

1. Choose n known, linearly-independent basis functions $\Psi_i(x) : \mathbb{R} \to \mathbb{R}$.
2. Define the linear projection

$$f^n(x;\theta) = \sum_{i=0}^{n} \theta_i \Psi_i(x)$$

3. Plug $f^n(x;\theta)$ into $\mathcal{H}(\cdot)$ to obtain the *residual function*

$$R(x;\theta) = \mathcal{H}(f^n(x;\theta))$$

4. Find θ such that weighted averages with weight functions $\phi_i(x)$, of the residual function are zero

$$W_i(\theta) = \int_{x \in X} \phi_i(x) R(x;\theta)) dx = 0, \quad i = 0, \cdots, m$$

Interpolation

- m weighted residual equations $W_i(\theta) = 0$. We need $m \geq n$ for θ to be determined.
- Finite $m = n \rightarrow$ interpolation
  - θ is such that the residual equation is zero at the (interpolation) nodes x_i

$$R(x_j; \theta) = 0, \quad j = 0, \cdots, m$$

  - Obtains if the weight functions $\phi_i(x) = 1$ at the interpolation nodes x_i and are zero otherwise.

3.2 Approximating a known function

Approximating a function by interpolation

- Function is known but is too costly to compute. We want to approximate it.
- Subcase of the general one.
- Functional equation used is

$$\mathcal{H}(f(x_j)) = f_j - f(x_j) = 0, \quad j = 0, \cdots, m.$$

- The residual function is

$$R(x_j; \theta) = f_j - \sum_{i=0}^{n} \theta_i \Psi_i(x_j), \quad j = 0, \cdots, m$$

- Interpolation solves

$$f_j = \sum_{i=0}^{n} \theta_i \Psi_i(x_j).$$

Compare to OLS

Let

$$Y = \begin{bmatrix} f_0 \\ \vdots \\ f_n \end{bmatrix}, \; X = \begin{bmatrix} \Psi_0(x_0) & \cdots & \Psi_n(x_0) \\ \vdots & \ddots & \vdots \\ \Psi_0(x_n) & \cdots & \Psi_n(x_n) \end{bmatrix}.$$

Then

$$Y = X\theta.$$

- Same number of points as parameters to determine. Is the problem as bad as it would be in empirical work?
- What happens if we decide to increase n?
- Important when choosing basis.

Choice of basis

1. **Spectral methods:** each element of the basis is non-zero for almost all $x \in X$ (global basis).
   - E.g. monomial basis $\{1, x, x^2, ..., x^n\}$.
2. **Finite elements methods:** divide X into non-intersecting subdomains (elements). Set the weighted residual functions to zero on each of the elements.
   - Splines: e.g. piecewise linear interpolation.

3.3 Spectral bases

Spectral bases: polynomials

Theorem 1 (Weierstrass) *A function $f : [a, b] \to \mathbb{R}$ is approximated "arbitrarily well" by the polynomial*

$$\sum_{i=0}^{n} \theta_i x^i$$

for n large enough.

- f does not need to be continuous
- but n may have to be large to get a good approximation if f is discontinous.

Spectral bases: monomials

$$\Psi_i(x) = x^i \quad i = 0, \cdots, n$$

- Simple and intuitive
- Problems:
  - Near multicollinearity
  - Vary considering in size $\to$ scaling problems and accumulation of numerical error
- We want an orthogonal basis.

Spectral bases: orthogonal polynomials

- Choose orthogonal basis functions; i.e.

$$\int_a^b \Psi_i(x)\Psi_j(x)w(x)dx = 0, \quad \forall i, j \text{ with } \; i \neq j$$

- Different families associated with different weighting function $w(x)$ and ranges $[a, b]$.

Spectral bases: Chebyshev orthogonal polynomials

- $[a, b] = [-1, 1]$ and $w(x) = \frac{1}{(1-x^2)^{1/2}}$
- The basis functions may more easily be recovered from the recursive formula

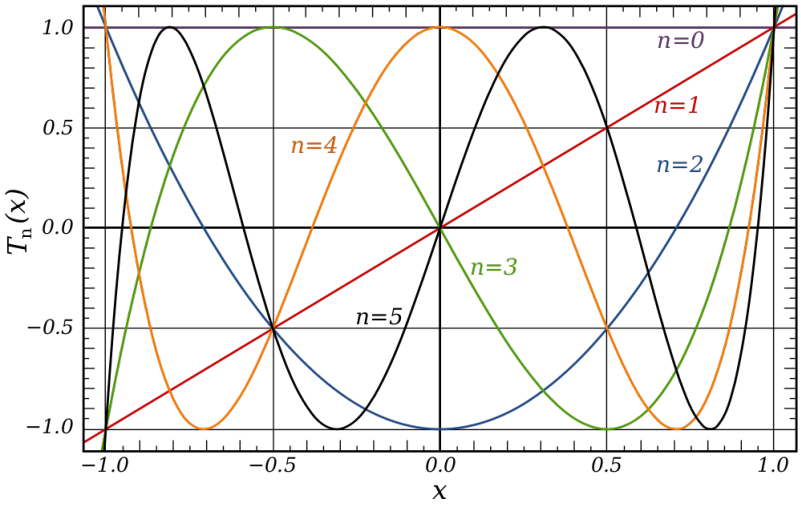
$$\Psi_0^c(x) = 1$$
$$\Psi_1^c(x) = x$$
$$\Psi_i^c(x) = 2x\Psi_{i-1}^c(x) - \Psi_{i-2}^c(x)$$

Chebyshev nodes

- The n^{th}-order Chebyshev basis function has n zeros
- These are the n Chebyshev nodes for an n^{th}-order approximation
- They satisfy the formula

$$z_{j-1} = -cos\left(\frac{2j-1}{2n}\pi\right) \quad j = 1, \cdots, n$$

Some Chebyshev polynomials



Chebyshev interpolation over a generic interval

- Suppose $f(x), f : [a, b] \to \mathbb{R}$
- Chebyshev polynomials are defined over $[-1, 1]$
- Find the points in $[a, b]$ corresponding to the Chebyshev nodes z_j

$$x_j = a + \frac{z_j + 1}{2}(b - a)$$

- Calculate the functions values $f_j = f(x_j)$ at the nodes x_j
- θ solves the projection system

$$f_j = \sum_{i=0}^{n} \theta_i \Psi_i^c(z_j) \quad j = 0, \cdots, n$$

Orthogonality at the nodes

- The Chebyshev polynomials evaluated at the nodes satisfy the orthogonality property

$$\sum_{j=0}^{n} \Psi_i^c(z_j)\Psi_k^c(z_j) = 0 \quad \text{for } i \neq k$$

- It follows that if

$$X = \begin{bmatrix} \Psi_0^c(z_0) & \cdots & \Psi_n^c(z_0) \\ \vdots & \ddots & \vdots \\ \Psi_0^c(z_n) & \cdots & \Psi_n^c(z_n) \end{bmatrix}.$$

  then $X'X$ is a diagonal matrix
- Each θ_i is just a function of $\Psi_i^c(z_j)$ and $f(x_j)$
- Of course... omitting a variable orthogonal to the included regressors has no effect on the regression coefficients

Uniform convergence

- Weierstrass theorem implies there is always a polynomial that gives a good enough approximation
- It does not imply that the quality of *any* approximation improves monotonically as the order increases.
- Instead, the polynomial approximation converges uniformly to the function to be approximated if the polynomials are fitted on the Chebyshev nodes.

Chebyshev regression

- Like standard regression
- n nodes but polynomial of degree $m < n$
- Trade-off between the various points (no longer exact approximation at the nodes)

3.4 Finite elements

Finite elements: splines

- Spectral (polynomial) methods use the same polynomial over the whole domain of x
- Finite element methods split the domain of x into non-intersecting subdomains (elements) and fit a different polynomial for each element
  - Advantageous if the function can only be approximated well by a high order polynomial over the entire domain but by low-order polynomials over each subdomain
  - Elements do not need to have equal size: can be smaller in regions were the function is more "difficult"
- $n+1$ nodes $x_0, \cdots, x_n$ and corresponding function values $f_0, \cdots, f_n$
  - Still interpolation

Finite elements as projection

- Two equivalent ways to think about finite elements/splines as a projection method.
  1. They fit to each subinterval basis functions which are non-zero over most of the subinterval
     - Polynomial bases in the case of splines
  2. They fit the same set of basis functions to all the domain of x but the functions are zero over most of the interval. The basis functions apply to all the domain but they are zero
- Example: step function
  1. The basis functions $\Psi_0 = 1$ in all subintervals
  2. The basis functions are $\Psi_i = \mathbb{I}_{x_i \leq x < x_{i+1}}$ where $\mathbb{I}$ is the indicator function.

Finite elements: piece-wise linear splines

- For $x \in [x_i, x_{i+1}]$

$$f(x) \approx f^1(x) = \left(1 - \frac{x - x_i}{x_{i+1} - x_i}\right) f_i + \left(\frac{x - x_i}{x_{i+1} - x_i}\right) f_{i+1}$$

- One can think of the two terms in parentheses as the two basis in each interval and the function values as the associated coefficients

Finite elements: piece-wise linear splines

- Piece-wise linear splines preserve shape, namely monotonicity and (weak) concavity.
- Yet they are non-differentiable at the nodes.
- Easily solved by fitting higher order polynomial in each subdomain.

Finite elements: higher order splines

- Still $n+1$ nodes and associated function values.
- Now second order polynomial in each interval

$$f(x) \approx f^2(x) = a_i + b_i x + c_i x^2 \quad \text{for } x \in [x_i, x_{i+1}]$$

- Now we have $3n$ parameters to determine

Quadratic splines: levels

- $2 + 2(n-1)$ value matching conditions as in the linear case.
  - For the intermediate nodes the quadratic approximations on both sides have to coincide; e.g.

$$f_1 = a_1 + b_1 x_1 + c_1 x_1^2$$
$$f_1 = a_2 + b_2 x_1 + c_2 x_1^2$$

- Only one quadratic has to satisfy value matching at the two endpoints x_0 and x_n.

Quadratic splines: slopes

- Differentiability at the intermediate nodes requires smooth pasting (same derivatives on both sides); e.g.

$$b_1 + 2c_1 x_1 = b_2 + 2c_2 x_1$$

- $n-1$ more conditions
- We need one more: arbitrary.
  - e.g. set slope at one of the two terminal nodes equal to some value.

Shape-preserving splines

- Higher order splines do not preserve monotonicity and concavity in general.
- Schumacher splines do

4 Extensions

Functions of more than one variable

- Extending polynomial approximation to variables of more than one variable is relative straightforward (but curse of dimensionality)
- n^{th}-order approximation to the function f(x,y)
  - Complete polynomial

$$\sum_{i+j \leq n} \Psi_i(x)\Psi_j(y)$$

  - Tensor product polynomial

$$\sum_{i,j \leq n} \Psi_i(x)\Psi_j(y)$$

Lecture 2: Numerical integration and contraction mappings

Roadmap for this part

- Numerical integration
  - Quadrature techniques
    - Newton Cotes
    - Gaussian quadrature
  - Monte Carlo
- Bellman equation and the contraction mapping theorem.

Readings for this lecture

1. The notes on "Numerical Integration" by Wouter den Haan at http://tinyurl.com/6cbrqqr
2. Chapters 7.2, 8.1 and 8.2 in Judd (1998)
3. Chapter 3 and Theorem 4.6 in (SL) and appendix A in (LS).

1 Numerical integration

1.1 Quadrature techniques

Quadrature techniques

Approximating an integral by a finite sum

$$\int_a^b f(x)dx \approx \sum_{i=1}^{n} \omega_i f(x_i)$$

- Newton Cotes
  - Arbitrary (usually equidistant) nodes x_i and efficient weights ω_i
  - Will not consider further
- Gaussian quadrature
  - Both nodes and weights chosen efficiently

1.1.1 Gaussian quadrature

Gaussian quadrature

- Exact integration of
$$\int_a^b f(x)w(x)dx$$
if $f(x)$ is a polynomial of order $2n-1$
  - 7 nodes give exact integration for polynomials up to order 13!
- Different families for different $[a,b]$ and different *weighting functions* $w(x)$
- Good approximation if $f(x)$ is well approximated by polynomial of order up to $2n-1$

Gaussian-Legendre quadrature

- Defined over $[-1,1]$, $w(x)=1$
- Exact integration of
$$\int_a^b f(x)dx$$
if $f(x)$ is a polynomial of order $2n-1$
- For generic $[a,b]$ rescale Gauss-Legendre nodes x_i^{GL} and weights ω_i^{GL} using
$$x_i = a + \frac{x_i^{GL}+1}{2}(b-a)$$
$$\omega_i = \frac{b-a}{2}\omega_i^{GL}$$

Practical implementation

- Generate n Gauss-Legendre nodes x_i^{GL} and weights ω_i^{GL} with appropriate computer subroutine
- Rescale nodes
$$x_i = a + \frac{x_i^{GL}+1}{2}(b-a)$$
- Solution equals
$$\int_a^b f(x)dx \approx \sum_{i=1}^{n}\omega_i f(x_i) = \frac{b-a}{2}\sum_{i=1}^{n}\omega_i^{GL} f(x_i)$$

- REMARK: x_i^{GL} and ω_i^{GL} depend just on n NOT on $f(x)$
  - What determines nodes and weights?

Gauss-Legendre nodes and weights

- $2n$ unknows: n nodes x_i^{GL} + n weights ω_i^{GL}
- Chosen to ensure exact integration for polynomial of order $2n-1$ over $[-1,1]$ interval
  - Monomial $f(x) = 1$[1]

$$\int_{-1}^{1} 1dx = \sum_{i=1}^{n} \omega_i^{GL} 1$$

  - Monomials $f(x) = x^j$

$$\int_{-1}^{1} x^j dx = \sum_{i=1}^{n} \omega_i^{GL} (x_i^{GL})^j \quad j = 1, \cdots, 2n-1$$

  - $2n$ equations in $2n$ unknowns

What about general polynomial functions

- A generic polynomial is a linear combination of monomials
- Exact integration for *any* polynomial of order $2n-1$

1.1.2 Gaussian-Hermite quadrature

Gaussian-Hermite quadrature

- Defined over $[-\infty, +\infty]$, $w(x) = e^{-x^2}$
- Used for expectations of functions of normally distributed random variables
- We want to find nodes x_i and weights ω_i such that

$$\int_{-\infty}^{+\infty} f(x) e^{-x^2} dx \approx \sum_{i=1}^{n} \omega_i f(x_i)$$

- Cfr. Gaussian-Legendre
  - weighting function e^{-x^2} instead of 1
  - even if $f(x)$ is well approx. by a polynomial, $f(x)e^{-x^2}$ is not

[1] The rescaling $\omega_i = (b-a)\omega_i^{GL}/2$ comes from the fact that $\int_{-1}^{1} 1dx = 2 = \sum_{i=1}^{n} \omega_i^{GL} 1$ and $\int_a^b 1dx = b - a = \sum_{i=1}^{n} \omega_i 1$.

Practical implementation

- Generate n Gauss-Hermite x_i^{GH} and weights ω_i^{GH} with appropriate computer subroutine
- Solution equals
$$\int_{-\infty}^{+\infty} f(x)e^{-x^2}dx \approx \sum_{i=1}^{n} \omega_i^{GH} f(x_i^{GH})$$
- REMARK: weighting function e^{-x^2} is captured by the weights.

Expectations of functions of normally distributed r.v.

- Suppose $x \sim N(\mu, \sigma)$
- Expectation of $f(x)$ is
$$\mathbb{E}[f(x)] = \int_{-\infty}^{+\infty} \frac{1}{\sigma\sqrt{2\pi}} f(x)e^{-\frac{(x-\mu)^2}{2\sigma^2}}dx$$
- Not quite Gauss-Hermite weighting function, we need a change of variable

Change of variable

- Define the auxiliary variable $y = \frac{x-\mu}{\sqrt{2}\sigma}$ which implies
$$x = \mu + \sqrt{2}\sigma y, \quad dx = \sqrt{2}\sigma dy$$
- Replacing on RHS of $\mathbb{E}[f(x)] = \int_{-\infty}^{+\infty} \frac{1}{\sigma\sqrt{2\pi}} f(x)e^{-\frac{(x-\mu)^2}{2\sigma^2}}dx$
$$\mathbb{E}[f(x)] = \int_{-\infty}^{+\infty} \frac{1}{\sqrt{\pi}} f(\mu + \sqrt{2}\sigma y)e^{-y^2}dy$$
- Defining $x_i = \mu + \sqrt{2}\sigma x_i^{GH}$ yields
$$\mathbb{E}[f(x)] \approx \sum_{i=1}^{n} \omega_i f(x_i) = \sum_{i=1}^{n} \frac{1}{\sqrt{\pi}} \omega_i^{GH} f(x_i)$$

Practical implementation

- Generate n Gauss-Hermite nodes x_i^{GH} and weights ω_i^{GH} with appropriate computer subroutine
- Rescale nodes
$$x_i = \mu + \sqrt{2}\sigma x_i^{GH}$$
- Solution equals
$$\mathbb{E}[f(x)] \approx \sum_{i=1}^{n} \frac{1}{\sqrt{\pi}} \omega_i^{GH} f(x_i)$$
- REMARK: μ and σ just affect x_i.

Persistent processes

- If x follows an AR(1) process its conditional mean changes over time.
- Set of Gauss-Hermite nodes expands with time.
- Solutions (reference: lecture notes by Karen Kopecky at http://tinyurl.com/2cxw4rs)
  - Tauchen (86) method
  - Tauchen and Hussey (91) method
  - Rouwenhorst (95) method

1.2 Monte Carlo integration

Monte Carlo Integration

- Read the relevant section in Judd.

2 The Contraction Mapping theorem

Theorem of the maximum

Let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$. Let $\Gamma : X \to Y$ and $f : X \times Y \to \mathbb{R}$. Consider the problem of choosing $y \in \Gamma(x)$ to maximize the function $f(x, y)$. Let

$$h(x) = \max_{y \in \Gamma(x)} f(x, y)$$
$$\hat{y}(x) = \arg\max_{y \in \Gamma(x)} f(x, y)$$

Theorem 2 (Theorem of the Maximum) *Let $X \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}^m$. Let $\Gamma : X \to Y$ be a compact-valued and continuous correspondence and $f : X \times Y \to \mathbb{R}$ be a continuous function. Then*

1. *$\hat{y}(x)$ is a non-empty and compact-valued correspondence;*
2. *h(x) is a continuous function.*

Important property of the Bellman equation

$$v_t(x_t) = \max_{x_{t+1} \in \Gamma(x_t)} F(x_t, x_{t+1}) + \beta v_{t+1}(x_{t+1}) \tag{BE}$$

Assumption 1 (Assumption 1) *$x_t \in X \subseteq \mathbb{R}^n$, $\Gamma : X \to X$ is a continuous and compact-valued correspondence and $F : X \times X \to \mathbb{R}$ is a continuous function.*

Corollary 1 *If Assumption 1 is satisfied and $v_{t+1} : \mathbb{R}^n \to R$ is a continuous function, then v_t is a continuous function.*

- Bellman equation (BE) maps the space $C^0(X)$ onto itself
- Proof: straightforwardly from Theorem of the Maximum

Finite horizon dynamic programming

$$v_t(x_t) = \max_{x_{t+1} \in \Gamma(x_t)} F(x_t, x_{t+1}) + \beta v_{t+1}(x_{t+1})$$

with $t \leq T < \infty$ and $v_{T+1} = 0$.

- If Assumption 1 is satisfied then $\hat{x}_{t+1}(x_t)$ is a non-empty, compact-valued correspondence and $v_t(x_{t)}$ is a continuous function.
  - At $t = T$ it is $v_T(x_T) = \max_{x_{T+1} \in \Gamma(x_T)} F(x_T, x_{T+1})$
  - At any $t < T$ it follows from the previous corollary
- Solution $\{\hat{x}_{t+1}(x_t), v_t(x_{t)}\}_{t=0}^{T}$ also found by backward induction

Infinite horizon dynamic programming

$$v_t(x_t) = \max_{x_{t+1} \in \Gamma(x_t)} F(x_t, x_{t+1}) + \beta v_{t+1}(x_{t+1})$$

- Indeed the problem is stationary. The solution is a pair $\{x'(x), v(x)\}$ solving

$$v(x) = \max_{x' \in \Gamma(x)} F(x, x') + \beta v(x')$$

- Problem: no terminal date from which to start backward induction
  - Does a solution exist?
  - If a solution exists, how can we find it?

Main result (to prove)

- (Existence) If Assumption 1 is satisfied the infinite horizon dynamic programming problem has a unique solution
- (Solution) The solution can be found by iterating on the Bellman equation

$$v^{n+1}(x) = \max_{x' \in \Gamma(x)} F(x, x') + \beta v^n(x')$$

  starting from any continuous function v^n.
  - Same as finite horizon but starting from *any* arbitrary "guess" function.
  - ... but a better guess implies faster convergence!

Existence (to prove)

Let

$$T(v) = \max_{x' \in \Gamma(x)} F(x, x') + \beta v(x')$$

- Mapping $T(v)$ is a functional (maps fns into fns)
- The Bellman equation has a solution if $T(v)$ has a fixed point

$$v^* = T(v^*)$$

- We need a fixed-point theorem for functionals

Mathematical preliminaries

Definition 1 *A **metric space** is a set S together with a distance function $\rho : S \times S \to \mathbb{R}$, such that for all $x,\ y, z \in S$:*

1. *$\rho(x, y) \geq 0$ with equality iff $x = y$;*
2. *$\rho(x, y) = \rho(y, x)$;*
3. *$\rho(x, z) \leq \rho(x, y) + \rho(y, z)$.*

Mathematical preliminaries II

Definition 2 *A sequence $\{x_n\}_{n=0}^{\infty}$ in a metric space (S, ρ) **converges** to $x \in S$ if for each real scalar $\epsilon > 0$ there exists N_ϵ such that*

$$\rho(x_n, x) < \epsilon, \text{ for all } n > N_\epsilon.$$

Definition 3 *A sequence $\{x_n\}_{n=0}^{\infty}$ in a metric space (S, ρ) is **Cauchy** if for each real scalar $\epsilon > 0$ there exists N_ϵ such that*

$$\rho(x_n, x_m) < \epsilon, \text{ for all } n, m > N_\epsilon.$$

Cauchy vs convergent sequences

- The first definition requires knowledge of the limit point x to be operational.
- The second definition does not, but
  - a convergent sequence in a metric space (S, ρ) is Cauchy
  - a Cauchy sequence in a metric space (S, ρ) may be convergent only in a metric space other than (S, ρ)

    E.g. Let S be the set of rational number and $\rho(x, y) = |x - y|$. The sequence

$$x_n = \left(1 + \frac{1}{n}\right)^n$$

    is Cauchy in (S, ρ) but its limit is the irrational number e.

Complete metric spaces

Definition 4 *A metric space (S, ρ) is **complete** if every Cauchy sequence in (S, ρ) converges to a limit in S.*

- For complete metric spaces the convergence of a sequence can be verified by means of the Cauchy criterion.
- Working with complete metric spaces is easier.

Normed vector spaces

Definition 5 *A **real vector (or linear) space** is a set $X \subseteq R^n$ together with two operations - addition and multiplication by a real scalar - such that it has a zero element and is closed under the two operations.*

Definition 6 *A **normed vector space** is a vector space S, together with a norm $||\cdot|| : S \to \mathbb{R}$ such that for all $x,\ y \in S$ and $\alpha \in \mathbb{R}$*

1. $||x|| \geq 0$;
2. $||\alpha x|| = |a| \cdot ||x||$;
3. $||x + y|| \leq ||x|| + ||y||$.

Metric vs normed vector spaces

Remark 1 *For a normed vector space $(S, ||\cdot||)$ a metric can be defined by means of the norm $||\cdot||$; namely for all $x,\ y \in S$*

$$\rho(x, y) = ||x - y||.$$

- Every normed vector space is a metric vector space under the above metric.

Banach spaces

Definition 7 *A **Banach space** is a complete normed (metric) vector space.*

Theorem 3 *Let $X \subseteq \mathbb{R}^n$. The space $C^0(X)$ of bounded continuous functions $f : X \to \mathbb{R}$ together with the sup norm $||f||_\infty = \sup_{x \in X} |f(x)|$ is a Banach space.*

Contraction mapping

Definition 8 *Let (S, ρ) be a metric space and $T : S \to S$ a function. T is **contraction mapping** (with modulus β) if for some $\beta \in (0, 1)$ it is $\rho(T(x), T(y)) \leq \beta\rho(x, y)$ for all $(x,\ y) \in S$.*

- A contraction mapping on S shrinks the distance between any two points in S.
- The application of a contraction mapping on (S, ρ) generates a Cauchy sequence.

Contraction mapping theorem

Theorem 4 (Contraction mapping theorem) *If (S, ρ) is a complete metric space and $T : S \to S$ is a contraction mapping with modulus β, then*

1. *T has a unique fixed point in S;*
2. *for any $v^0 \in S$, $\rho(T^n(v^0), v) \leq \beta^n \rho(v_0, v)$.*

Blackwell sufficient conditions

Theorem 5 (Blackwell sufficient conditions) *Let $X \in R^n$ and $B(x)$ a space of bounded functions $f : X \to X$ equipped with the sup norm. An operator $T : B(X) \to B(X)$ is a contraction mapping if it satisfies*

1. *(monotonicity) given $f, g \in B(x)$ and $f(x) \leq g(x)$ for all $x \in X$ it is $T(f) \leq T(g)$ for all $x \in X$;*
2. *(discounting) there exists $\beta \in (0, 1)$ such that*

$$T(f(x) + a) \leq T(f(x)) + \beta a, \text{ for all } f \in B(X),\ x \in X,\ a \geq 0.$$

At last...

Assume that:

- Assumption 1 holds;
- either $F(x, x')$ $x,\ x' \in X$ is bounded or X is compact.

Then

- the Bellman operator $T(v) = \max_{x' \in \Gamma(x)} F(x, x') + \beta v(x')$ maps $C^0(x)$ onto itself
- The space $(C^0(X), ||\cdot||_\infty)$ is a Banach space
- If $T(v)$ is a contraction mapping it has a unique fixed point in $C^0(x)$
- $T(v)$ satisfied Blackwell sufficient conditions
- Discounted dynamic programming problems with bounded returns have a unique solution.

Approximation Error bound

- The Contraction Mapping Theorem bounds the distance between the $n-$th iteration and the true (limit) value by

$$\rho(T^n v^0, v) \leq \beta^n \rho(v^0, v)$$

- Interesting but not operational as v is in general unknown.
- Operational bound

$$\rho(T^n v^0, v) < \frac{1}{1-\beta} \rho(v^n, v^{n+1})$$

- Can be used to establish convergence up to desired tolerance!

Lecture 3: Two solution methods for DP problems

Roadmap for this part

- A refresher of optimization
- Two solution methods for DP problems.
  - Discretized value function iteration
  - The method of endogenous grid points

Readings for this lecture

1. p. 99-100 in Judd (1998)
2. Chapters 4.1-4.5 in LS
3. Carroll (2006) and Barillas Villaverde (2007).

1 A refresher of optimization

Locating maxima

- Theorem of the maximum gives sufficient conditions for existence of a maximum.
  - If objective is not continuous we are on shaky ground
  - We assume continuity in what follows
- How to locate a maximum
  - Non-differentiable vs differentiable problems
  - Concave vs non-concave problems

Locating maxima (non-differentiable problems)

- We cannot use first-order conditions
- We need to use global comparison methods
  - Optimization on a discrete domain (effectively grid search)
    - Always a good starting point
    - If maximand is continuous it finds an approximate global max
    - The finer the grid the better the approximation
  - Polytope methods
- Concave problem
  - Unique maximized value for objective
  - Strictly concave: unique maximum

Locating maxima (differentiable problems)

- First order condition (FOC) is necessary for a maximum
  - Reduces maximization problem to root-finding problem
- Concave problems
  - FOC is also sufficient
  - Strictly concave: FOC is necessary and sufficient for a unique maximum

2 Two methods for solving DP problems

Value function iteration

- We have a number of theoretical results
  - Unique solution under quite general assumptions; hence...
  - It will always work (though possibly slow)
- We have tight convergence properties and error bounds
- It can be easily parallelized

A workhorse example: the stochastic growth model

The stochastic growth model

$$\begin{aligned}\max_{\{c_t,k_{t+1}\}_{t=0}^{T}} \mathbb{E}_0 \sum_{t=0}^{T} \beta^t u(c_t) \\ \text{s.t. } k_{t+1} = e^{z_t} k_t^{\alpha} - c_t,\ k_{t+1} \geq \underline{k},\ c_t \geq 0 \\ z_t = \rho z_{t-1} + \epsilon_t,\ k_0, z_0 \text{ given},\ \epsilon_t \sim N(0,\sigma)\end{aligned}$$

can be written as

$$\begin{aligned}\max_{\{k_{t+1}\}_{t=0}^{T}} \mathbb{E}_0 \sum_{t=0}^{T} \beta^t u(e^{z_t} k_t^{\alpha} - k_{t+1}) \\ \text{s.t. } k_{t+1} = \Gamma(k_t, z_t) = [\underline{k}, e^{z_t} k_t^{\alpha}] \\ z_t = \rho z_{t-1} + \epsilon_t\end{aligned}$$

The stochastic growth model: Bellman equation

$$V(k,z) = \max_{k' \in \Gamma(k,z)} u(e^z k^\alpha - k') + \beta \mathbb{E}[V'(k',z')|z]$$

- $V(k,z)$ stands for $V_t(k_t, z_t)$ and $V'(k',z')$ stands for $V_{t+1}(k_{t+1}, z_{t+1})$.
- We can write the Bellman equation as

$$V = T(V')$$

  where $T(\cdot)$ is the right hand side of the previous equation.
- Given an initial/terminal value for V' repeated application of the operator yields an approximation arbitrary closed to the true value function.

Normalization

- Before starting the algorithm it is a good idea to normalize the problem by replacing $u(\cdot)$ by its linear transformation $(1-\beta)u(\cdot)$

$$V(k,z) = \max_{k' \in \Gamma(k,z)} (1-\beta) u(e^z k^\alpha - k') + \beta \mathbb{E}[V'(k',z')|z]$$

- Remember: expected utility is defined up to an affine transformation
- Advantages:
  - Stability: weighted average
  - Convergence bounds are easier to interpret
- We will not do this in what follows to simplify notation

Discretization

- We can evaluate the Bellman equations only at a finite number of points.
- If the state space is continuous we need to discretize it
  - Exogenous stochastic state variable
    - Grid $\mathcal{Z} = [z_1, z_2, \ldots, z_m]$
  - Endogenous state variables
    - Grid $\mathcal{K} = [k_1, k_2 \ldots, k_n]$
- Tradeoff
  - Accuracy vs curse of dimensionality

Choice of grid for endogenous state variables

- Ideally $\mathcal{K}$ has to contain $\Gamma(k, z)$ for all z and "relevant" k
  - $k_1 = \underline{k}$ but k_n is unknown if k is unbounded above
  - Choose large enough k_n and verify that it is never binding for the optimal choice.
- How to fill the interval $[k_1, k_n]$
  - Chebyshev nodes if polynomial approximation; otherwise...
  - Use economic theory and error analysis to assess where to cluster points
  - We usually want more grid points where value function has more curvature
  - Problem: just a heuristic argument, may be self-confirming

Choice of grid for exogenous stochastic variables

- Unless the stochastic process for z is already discrete (Markov chain) it has to be discretized.
- Nodes z_i and weights π_{ij} with $i, j = 1, \ldots, m$ with $\pi_{ij} = \Pr(z' = z_j | z = z_i)$.
  - Use appropriate quadrature nodes and weights if feasible.
  - Use approximate quadrature nodes and weights otherwise (i.e. trade-offs with persistent processes).

Implementation

1. Start with an initial guess
2. Apply the Bellman operator
   (a) Compute the expectation (integration)
   $$\tilde{V}(k', z) = \mathbb{E}V'[(k', z')|z]$$
   (b) Compute the optimal policy (maximization)
   $$\hat{k}' = \arg \max_{k' \in \Gamma(k,z)} u(e^z k^\alpha - \hat{k}') + \beta \tilde{V}(k', z)$$
   (c) Replace for the optimal policy to obtain
   $$V = T(V') = u(e^z k^\alpha - \hat{k}') + \beta \tilde{V}(\hat{k}', z)$$
3. Iterate on 2. until convergence.

Choice of initial guess (functional form)

- Finite horizon
  - $V'(k', z') = V_T(k_T, Z_T) = u(e^{z_T} k_T^{\alpha})$
- Infinite horizon
  - $V'(k', z') = V^0(k', z')$
  - The better the initial guess the faster convergence
  - Good guesses
    - Have same property as the solution (e.g. monotonicity, concavity)
    - Value fn in deterministic steady state $\rightarrow V^0(k, z) = u(e^z k^{\alpha})/(1 - \beta)$

Computing the expectation function (integration)

- Expected continuation value

$$\tilde{V}(k', z_i) = \sum_{j=1}^{n} \pi_{ij} V'(k', z_j), \quad i = 1, \ldots, m$$

- In matrix notations

$$\tilde{V}(k', z_i) = \Pi_i \cdot V'(k')$$

with $\Pi_i = [\pi_{i1}, \pi_{i2}, \ldots, \pi_{im}]$ and $V'(k') = \begin{bmatrix} V'(k', z_1) \\ V'(k', z_2) \\ \vdots \\ V'(k', z_m) \end{bmatrix}$

Computing the optimal policy (maximization)

- Most costly computational step
- Various methods
  - Discretized VFI
    - forces both k and k' to lie on the discrete grid $\mathcal{K}$
  - Endogenous grid method
    - force k' to lie on the discrete grid $\mathcal{K}$ and solves for k that satisfies the FOC
  - Other methods
    - Use numerical optimization algorithms

3 Discretized Value Function Iteration

Discretized VFI (maximization step)

- Finds optimum for the discretized problem by grid search

$$\max_{k' \in \mathcal{K}} u(e^{z_i} k_l^{\alpha} - k') + \beta \tilde{V}(k', z_i) \; i = 1, \ldots, m, \; = 1, \ldots, n$$

- Search over $\mathcal{K}$ rather than $\Gamma(k_l, z_i)$
- Easily implemented on a computer.
  - Just write in vector form and find largest component
  - Global comparison method (nearly always works)
- True problem is not discrete though
  - Good approximation requires lots of points.
  - Curse of dimensionality
  - Tradeoff: speed vs accuracy

Implementing discretized VFI (summary)

1. Choose grids $\mathcal{K} = \{k_l\}_{l=1}^n$ and $\mathcal{Z} = \{z_i\}_{i=1}^m$
2. Guess a value function $V^0(k, z)$
3. For $l = 1, \ldots, n$ and $i = 1, \ldots, n$ compute

$$V^{k+1}(k_l, z_i) = \max_{k' \in \mathcal{K}} u(e^{z_i} k_l^{\alpha} - k') + \beta \sum_j \pi_{ij} V_k(k', z_j)$$

4. If $||V^{k+1} - V^k||_{\infty} < \epsilon$ go to step 5; else got to step 3.
5. Stop (for a possible refinement see Step 3 on p. 413 in Judd).

Speeding up

- Monotonicity of policy function
- Concavity
- (Modified) Policy function iteration (aka Howard improvement)
  - Iterate on the Bellman equation for $n - 1$ times keeping policy function fixed
  - Solve maximization step every n iterations.

4 The endogenous grid method

The endogenous grid method

- Recently proposed by Carroll (2006) and Barillas and Fernández-Villaverde (2007)
- Differentiable and strictly concave problems
  - uses FOC
- Can be extended to certain non-differentiable and non-concave problems

Background: maximization using FOC

- Euler equation
$$u'(e^z k^\alpha - k') \geq \beta \tilde{V}_k(k', z),$$
with equality if $k' > \underline{k}$.
- Envelope condition
$$V_k(k, z) = u'(e^z k^\alpha - k'(k, z))\alpha e^z k^{\alpha-1}$$
and
$$\tilde{V}_k(k', z) = \mathbb{E}[V_k(k', z')|z]$$
- The three equations define an operator $T(\tilde{V}_k)$ mapping a function $\tilde{V}_k$ into a new function V_k.
  - We are looking for a fixed point $\tilde{V}_k = T(\tilde{V}_k)$

Maximization using FOC (implementation)

1. Start with an initial guess $V_k^0(k, z)$
2. Apply the Euler operator
   (a) Compute the expectation (integration)
   $$\tilde{V}_k^n(k', z) = \mathbb{E}[V_k^n(k', z')|z]$$
   (b) Compute the optimal policy (maximization)
   $$u'(e^z k^\alpha - \hat{k}') = \beta \tilde{V}_k^n(\hat{k}', z) \text{ or } \hat{k}' = \underline{k}$$
   (c) Replace for the optimal policy to obtain
   $$V_k^{n+1}(k, z) = u(e^z k^\alpha - \hat{k}'(k, z))\alpha e^z k^\alpha$$
3. Iterate on 2. until convergence of $\tilde{V}_k^n$

Remarks

- The algorithm effectively iterates on the policy function $k'(k, z)$ rather than the value function. It belongs to a class of algorithms known as "Time iteration".
  - Uniqueness of the solution follows from uniqueness of policy and value function.
  - We have no theoretical bounds for the error in the partial derivative of the value function $\tilde{V}_k$.
- Step 2.1 above is the usual one. Just apply quadrature.
- The main difference lies in the maximization step.
- Initially, we assume solution is interior in what follows.

Standard implementation of maximization step

- It is useful to define the intermediate state variable *total resources* $Y = e^z k^\alpha$
- For $z = z_i \in \mathcal{Z}$ and $k = k_l \in \mathcal{K}$ we have a grid point $Y_{il} = e^{z_i} k_l^\alpha$ for Y
- For all i, l, standard methods compute $k'(Y_{il}, z_i)$ solving

$$-(1-\beta)u'(Y_{il} - k') + \beta \tilde{V}_k(k', z_i) = 0$$

  By construction $k'(Y_{il}, z_i) = k'(k_l, z_i)$
- The Euler equation is non-linear in k', hence costly to solve

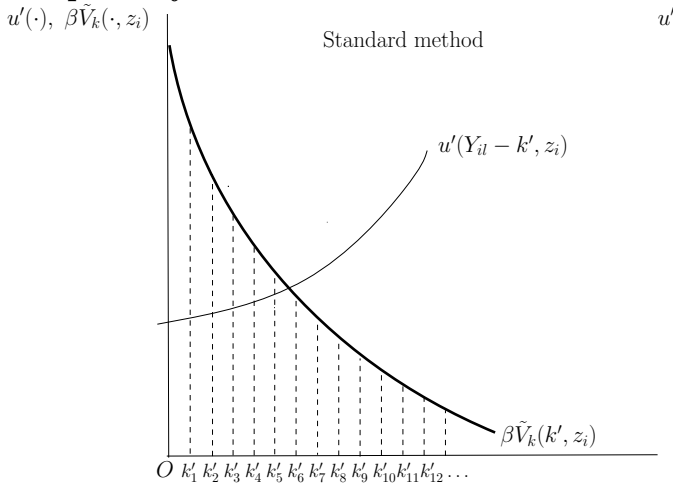
Maximization step in the endogenous grid method

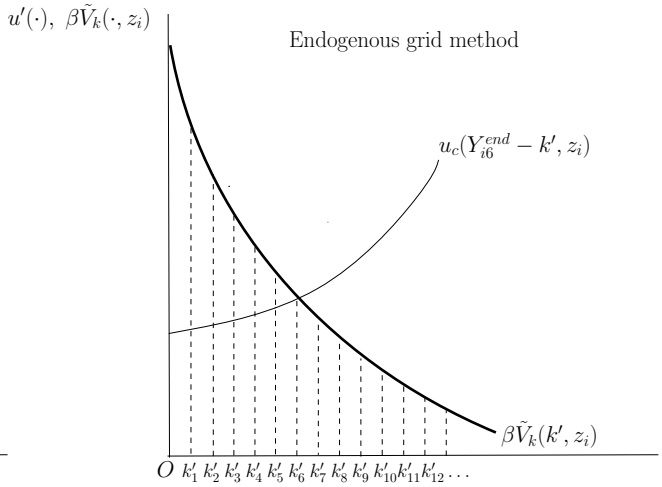
- Instead of making *current* k lie on a grid we solve for Y such that the optimal choice of *future* k' lies on a grid $\mathcal{K}$ with $k_1 = \underline{k}$
- For each $z = z_i \in \mathcal{Z}$ and $k' = k_m \in \mathcal{K}$ compute Y_{im}^{end}

$$Y_{im}^{end} - k_m = u'^{-1}\left(\beta \tilde{V}_k(k_m, z_i)\right)$$

  - Equivalent to standard methods as long as k' is invertible; but
  - The Euler eq. is linear in Y_{im}^{end}; no root finding!
- The set of pairs (k_m, Y_{im}^{end}) is the policy function $k'(Y_{im}^{end}, z_i)$ on the endogenous grid points Y_{im}^{end} for total resources.

Graphically





Recovering the policy fn on the exogenous grid

- The policy function $k'(Y^{end}_{im}, z_i)$ on the endogenous grid points for total resources Y^{end}_{im} implies a policy function $k'(k^{end}_{im}, z_i)$ where $Y^{end}_{im} = e^{z_i}(k^{end}_{im})^{\alpha}$.
- In general $k^{end}_{im} \notin \mathcal{K}$.
- To recover the policy function on $\mathcal{K} \times Z$ do the following
  - Construct the grid $Y_{il} = e^{z_i} k_l^a$ for all $z_i \in \mathcal{Z}, k_l \in \mathcal{K}$
  - For each (Y_{il}, z_i) "interpolate;" i.e.
    - If $Y_{il} > Y^{end}_{i1}$ obtain $k'(k_l, z_i)$ by linear interpolation of $k'(Y^{end}_{im}, z_i)$ on the two most adjacent nodes $Y^{end}_{ip}, Y^{end}_{i(p+1)}$ containing Y_{il}.
    - If $Y_{il} \leq Y^{end}_{i1}$, $k'(k_l, z_i) = \underline{k}$
      Total resources Y_{il} are below the minimum level Y^{end}_{i1} for which the Euler equation holds as an equality at $k' = \underline{k}$

Implementing EGM (summary) I

1. Define grids $\mathcal{K}$ and $\mathcal{Z}$. For each $z_i \in \mathcal{Z}$ construct a grid for total resources $Y_{il} = e^{z_i} k_l^{\alpha}$
2. Start with an initial guess $V^0_k(k, z)$
3. For each $z_i \in \mathcal{Z}$ and $k_m \in \mathcal{K}$
   - Compute the expectation
   $$\tilde{V}^n_k(k_m, z_i) = \sum\nolimits_j \pi_{ij} V^n_k(k_m, z_j) | z_i]$$
   - Compute Y^{end}_{im}
   $$Y^{end}_{im} - k_m = u'^{-1}\left(\beta\tilde{V}^n_k(k_m, z_i)\right)$$

Implementing EGM (summary) II

4. Recover $k'(k_l, z_i)$ by "interpolating" (k_m, Y_{im}^{end}) at the nodes Y_{il}
5. Replace for the optimal policy to obtain

$$V_k^{n+1}(k,z) = u(e^z k^\alpha - \hat{k}'(k,z))\alpha e^z k^\alpha$$

6. If $||V_k^{n+1}(k,z) - V_k^n(k,z)||\infty < \epsilon$ stop; else go to 3.