# Introduction to Dynamic Programming: Basic theory and numerical tools

QuantEcon Honours workshop

Fedor Iskhakov, Australian National University

QuantEcon-RSE
March 10, 2018

# Value of an infinite stream of payments

$$\begin{array}{ccccccc} & \nearrow & & \searrow & \searrow & \searrow & \\ V & & & c & c & c & \dots \end{array}$$

- Interest rate $r > 0$
- What is the value of the annuity $V$?

# Value of an infinite stream of payments

$$
\begin{array}{ccccc}
\nearrow & \searrow & \searrow & \searrow & \\
V & c & c & c & \dots
\end{array}
$$

- Interest rate $r > 0$
- What is the value of the annuity $V$?
- Discounted present values

$$\frac{c}{(1+r)^0} \qquad \frac{c}{(1+r)^1} \qquad \frac{c}{(1+r)^2} \qquad \cdots$$

- Let $\beta = \frac{1}{1+r}$

$$V = c + \beta c + \beta^2 c + \cdots = \sum_{t=0}^{\infty} \beta^t c$$

# Value of an infinite stream of payments

- Note that $\beta = \frac{1}{1+r} < 1$ because $r > 0 \Rightarrow$ converging geometric series

$$V = c + \beta c + \beta^2 c + \dots = \sum_{t=0}^{\infty} \beta^t c = \frac{c}{1-\beta}$$

- Another approach: reformulate this as a recursive equation

$$\begin{aligned} V &= c + \beta(c + \beta^2 c + \dots) \\ &= c + \beta V \end{aligned}$$

- Extremely simple case here, leads to the same answer
- In general, we need a range of computational tools which we can first try on this simple problem

# Backward induction

1. Start with a guess $V_0$
2. Insert into the recursive equation $V_1 = c + \beta V_0$
3. Insert new value $V_1$ into the recursive equation again $V_2 = c + \beta V_1$
4. Repeat until convergence ($i$ denotes iteration number)

$$||V_i - V_{i-1}|| \leq \varepsilon \text{ (small number)}$$

- Will this converge?

$$||V_i - V_{i-1}|| = ||(c + \beta V_{i-1}) - (c + \beta V_{i-2})|| = \beta||V_{i-1} - V_{i-2}||$$

- $\beta < 1 \Rightarrow$ with every iteration $i$ the difference $||V_i - V_{i-1}||$ becomes smaller (recursive formula is a contraction mapping)
- Banach fixed point theorem guarantees unique solution!

# Backward induction

1. Start with a guess $V_0$
2. Insert into the recursive equation $V_1 = c + \beta V_0$
3. Insert new value $V_1$ into the recursive equation again $V_2 = c + \beta V_1$
4. Repeat until convergence ($i$ denotes iteration number)

$$||V_i - V_{i-1}|| \leq \varepsilon \text{ (small number)}$$

- Will this converge?

$$||V_i - V_{i-1}|| = ||(c + \beta V_{i-1}) - (c + \beta V_{i-2})|| = \beta ||V_{i-1} - V_{i-2}||$$

- $\beta < 1 \Rightarrow$ with every iteration $i$ the difference $||V_i - V_{i-1}||$ becomes smaller (recursive formula is a contraction mapping)
- Banach fixed point theorem guarantees unique solution!

# Backward induction

1. Start with a guess $V_0$
2. Insert into the recursive equation $V_1 = c + \beta V_0$
3. Insert new value $V_1$ into the recursive equation again $V_2 = c + \beta V_1$
4. Repeat until convergence ($i$ denotes iteration number)

$$||V_i - V_{i-1}|| \leq \varepsilon \text{ (small number)}$$

- Will this converge?

$$||V_i - V_{i-1}|| = ||(c + \beta V_{i-1}) - (c + \beta V_{i-2})|| = \beta ||V_{i-1} - V_{i-2}||$$

- $\beta < 1 \Rightarrow$ with every iteration $i$ the difference $||V_i - V_{i-1}||$ becomes smaller (recursive formula is a contraction mapping)
- Banach fixed point theorem guarantees unique solution!

# Backward induction

1. Start with a guess $V_0$
2. Insert into the recursive equation $V_1 = c + \beta V_0$
3. Insert new value $V_1$ into the recursive equation again $V_2 = c + \beta V_1$
4. Repeat until convergence ($i$ denotes iteration number)

$$||V_i - V_{i-1}|| \leq \varepsilon \text{ (small number)}$$

- Will this converge?

$$||V_i - V_{i-1}|| = ||(c + \beta V_{i-1}) - (c + \beta V_{i-2})|| = \beta ||V_{i-1} - V_{i-2}||$$

- $\beta < 1 \Rightarrow$ with every iteration $i$ the difference $||V_i - V_{i-1}||$ becomes smaller (recursive formula is a contraction mapping)
- Banach fixed point theorem guarantees unique solution!

# Backward induction

1. Start with a guess $V_0$
2. Insert into the recursive equation $V_1 = c + \beta V_0$
3. Insert new value $V_1$ into the recursive equation again $V_2 = c + \beta V_1$
4. Repeat until convergence ($i$ denotes iteration number)

$$||V_i - V_{i-1}|| \leq \varepsilon \text{ (small number)}$$

- Will this converge?

$$||V_i - V_{i-1}|| = ||(c + \beta V_{i-1}) - (c + \beta V_{i-2})|| = \beta ||V_{i-1} - V_{i-2}||$$

- $\beta < 1 \Rightarrow$ with every iteration $i$ the difference $||V_i - V_{i-1}||$ becomes smaller (recursive formula is a contraction mapping)
- Banach fixed point theorem guarantees unique solution!

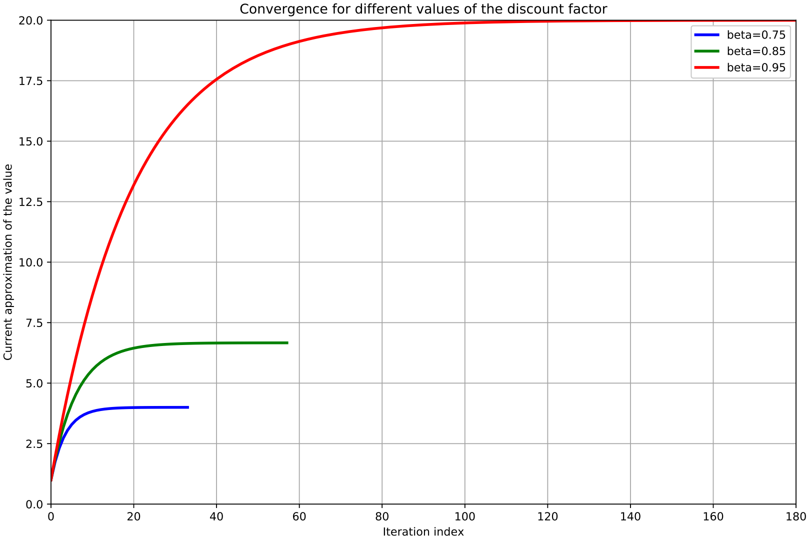
# Numerical illustration

The code:
**annuity.ipynb**

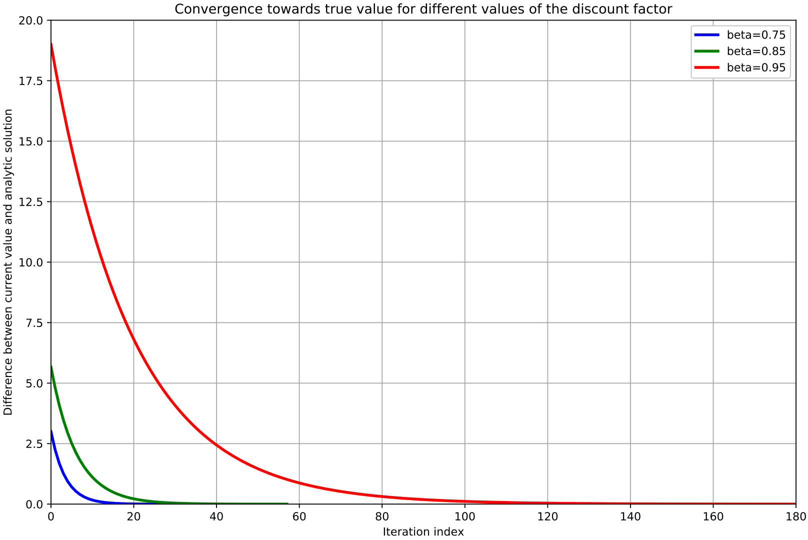
## Questions and discussion

1. Convergence to the analytical solution?
2. What determines the speed of convergence?
3. What happens when $\beta = 1$?

# Convergence speed and the role of $\beta$



# Convergence speed and the role of $\beta$



# Cake eating problem

- Cake of initial size $W_0$
- How much of the cake to eat each period $t$?
- What is not eaten in period $t$ is left for the future

$$W_{t+1} = W_t - c_t$$

- Utility flow from cake consumption

$$u(c_t) = \log(c_t)$$

- Future is discounted with discount factor $\beta$

- Optimization problem

$$\max_{\{c_t\}_0^\infty} \sum_{t=0}^{\infty} \beta^t u(c_t) \longrightarrow \max$$

# Recursive formulation = Bellman equation

- Value function $V(W_t)$ = the maximum attainable value given the size of cake $W_t$ (in period $t$)

$$
\begin{aligned}
V(W_0) &= \max_{\{c_t\}_0^\infty} \sum_{t=0}^{\infty} \beta^t u(c_t) \\
&= \max_{c_0} \{u(c_0) + \beta \max_{\{c_t\}_1^\infty} \sum_{t=1}^{\infty} \beta^{t-1} u(c_t)\} \\
&= \max_{c_0} \{u(c_0) + \beta V(W_1)\}
\end{aligned}
$$

- The Bellman equation is

$$
V(W_t) = \max_{0 \le c_t \le W_t} \left\{u(c_t) + \beta V(\underbrace{W_{t+1}}_{=W_t - c_t})\right\}
$$

# Dynamic programming

"DP is a recursive method for solving sequential decision problems"

John Rust 2006 *New Palgrave Dictionary of Economics*

**State variables** vector of variables that describe all relevant information about the modeled decision process, $W_t$

**Decision variables** vector of variables describing the choices, $c_t$

**Instantaneous payoff** utility function, $u(c_t)$, with time separable discounted utility

**Timing scale** discrete, finite/infinite horizon, discount factor $\beta$

**Motion rules** agent's beliefs of how state variable evolve through time, conditional on choices, $W_{t+1} = W_t - c_t$

**Value function** maximum attainable utility, function of the state variables, $V(W_t)$

**Policy function** mapping from state space to action space that returns the optimal choice, $c^\star(W_t)$

# Cake Eating: Analytical Solution

## Guess and verify

- Start with a (good) guess of $V(W) = A + B \log W$

$$\begin{aligned} V(W) &= \max_c \left\{ u(c) + \beta V(W-c) \right\} \\ A + B \log W &= \max_c \left\{ \log c + \beta (A + B \log(W-c)) \right\} \end{aligned}$$

Exercise: Determine $A$ and $B$ and find the optimal rule for cake consumption.

- This is only possible in *few* models!

- For cake eating problem

$$c^\star(W) = \arg\max_c \left\{ \log(c) + \beta V(W-c) \right\} = (1-\beta)W$$

# Cake Eating: Numerical Solution

Will **backward induction** work as before?

- Value of annuity:
$$V = c + \beta V$$
- Cake eating:
$$V(W) = \max_{0 \le c \le W} \left\{ u(c) + \beta V(W - c) \right\}$$
- Have to solve the functional equation for $V(W)$
- The Bellman operator in functional space
$$T(V)(W) \equiv \max_{0 \le c \le W} \{ u(c) + \beta V(W - c) \}$$
- The Bellman equations is then $V(W) = T(V)(W)$, with the solution given by the fixed point

# Can we find the fixed point by iterations?

- Need contraction property for $T(V)(W)$
- Blackwell sufficient conditions for contraction
  1. Monotonicity: satisfied due to maximization in $T(V)(W)$
  2. Discounting: satisfied by elementary argument when $\beta < 1$
- The Bellman operator is a contraction mapping!

**Contraction Mapping Theorem (Banach Fixed Point Theorem)**

Let $(S, \rho)$ be a complete metric space with a contraction mapping $T : S \to S$. Then

1. $T$ admits a unique fixed-point $V^\star \in S$, i.e. $T(V^\star) = V^\star$.
2. $V^\star$ can be found by repeated application of the operator $T$, i.e. $T^n(V) \to V^\star$ as $n \to \infty$.

# Value function iterations (VFI)

1. Start with an arbitrary guess $V_0(W)$
2. At each iteration $i$ compute

$$\begin{aligned} V_i(W) = T(V_{i-1})(W) &= \max_{0 \le c \le W} \{u(c) + \beta V_{i-1}(W-c)\} \\ c_{i-1}(W) &= \arg\max_{0 \le c \le W} \{u(c) + \beta V_{i-1}(W-c)\} \end{aligned}$$

3. Repeat until convergence

$$||V_i(W) - V_{i-1}(W)|| \le \varepsilon \text{ (small number,} || \cdot || \text{ sup norm)}$$

The contraction mapping theorem implies:

- Unique fixed point $\Leftrightarrow$ unique solution to the Bellman equation
- The fixed point can be reached by an iterative process using an arbitrary initial guess!
- Therefore VFI algorithm converges globally

# Cake Eating: numerical implementation

How to numerical implement the Bellman operator?

- Cake is continuous and thus value function is a function of continuous variable
- Solution: discretize $W$
  Construct a *grid* (vector) of cake-sizes $\vec{W} \in \{0, \dots \overline{W}\}$

$$V_i(\vec{W}) = \max_{0 \leq c \leq \vec{W}} \{u(c) + \beta V_{i-1}(\vec{W} - c)\}$$

- Compute value and policy function sequentially point-by-point
- May need to compute the value function between grid points
  $\Rightarrow$ Interpolation and function approximation

# Cake Eating: decision-state grid

Can interpolation be avoided?

- Note that conditional on $W_t$, the choice of $c$ defines $W_{t+1}$
- Can replace $c$ with $W_{t+1}$ in Bellman equation so that next period cake size is the decision variable
- "Dual" formulation of the same problem

$$V_i(\vec{W}) = \max_{0 \leq \vec{W}' \leq \vec{W}} \{u(\vec{W} - \vec{W}') + \beta V_{i-1}(\vec{W}')\}$$

- Compute value and policy function sequentially point-by-point
- Note that grid $\vec{W} \in \{0, \ldots \overline{W}\}$ is used twice: for state space and for decision space
- Only precise when number of grid points is large

# Cake eating: Numerical implementation I

The code:
**cake1.ipynb**

## Questions and discussion

1. Speed of convergence?
2. Magnitude of numerical errors?
3. The role of grid density?

# How to measure numerical errors?

- In our case there is an analytic solution

$$c^{\star}(W) = (1-\beta)W$$

- Typically very dense (slow) grid is used in place of true solution

- Can control for max or mean error at the grid points of value and policy functions

# Cake Eating: another numerical implementation

Control for grid over state space separately from the discretization of the choice variables to increase accuracy

- As before solve cake eating Bellman equation by VFI

$$V(W) = \max_{0 \le c \le W} \{u(c) + \beta V(W - c)\}$$

- Discretize state space with $\vec{W} \in \{0, \dots \overline{W}\}$
- Discretize decision space with $\vec{D} \in \{0, \dots \overline{D}\}$, usually $\overline{D} = \overline{W}$

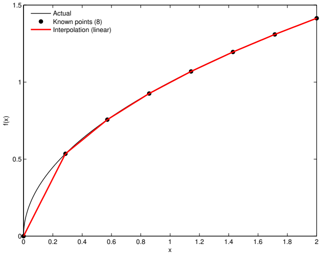
$$V_i(\vec{W}) = \max_{0 \le \vec{D} \le \vec{W}} \{u(c) + \beta V_{i-1}(\vec{W} - c)\}$$

- Compute value/policy function point-by-point on grid $\vec{W}$
- Find the maximum over the points of grid $\vec{D}$ that satisfy the choice set condition $0 \le \vec{D} \le W$

- Now have to compute the value function between grid points

# Function interpolation

- Cake consumption can take on infinitely many cake sizes $\rightarrow$ we need to be able to approximate the value function for all resulting levels of cake left to subsequent periods, $W' = W - c$.
- We have a grid $\vec{W}$, a set of points for which we have explicitly found the value function, $V(\vec{W})$




- For now, assume interpolation function is $\check{V}(W)$

# Cake eating: Numerical implementation II

The code:
**cake2.ipynb**

## Questions and discussion

1. Did convergence speed change?
2. Did magnitude of numerical errors change?

# Cake Eating: continuous choice implementation

Is it possible not to discretize the choice variable? Yes!

- Discretize state space with $\vec{W} \in \{0, \dots \overline{W}\}$

$$V_i(\vec{W}) = \max_{0 \leq c \leq \vec{W}} \{u(c) + \beta V_{i-1}(\vec{W} - c)\}$$

- Compute value/policy function point-by-point on grid $\vec{W}$
- Apply continuous solver to find the solution of the maximization problem inside Bellman equation for each point of the grid $\vec{W}$
  OR
- Solve the first order conditions given by the Eurler equation

- Contraction mapping property does not rely in the nature of the control variable $\Rightarrow$ VFI still works!

# Euler Equation

1. First order condition (FOC) of the Bellman equation w.r.t. $c$

$$0 = u'(c_t) + \beta \frac{\partial V(W_{t+1})}{\partial c_t} = u'(c_t) + \beta V'(W_{t+1}) \underbrace{\frac{\partial W_{t+1}}{\partial c_t}}_{-1},$$

$$\Rightarrow u'(c_t^\star) = \beta V'(W_{t+1})$$

2. From the Envelope theorem we have

$$V'(W_t) = \frac{\partial}{\partial W_t} \big[u(c_t) + \beta V(W_{t+1})\big]\Big|_{c_t^\star} = \beta V'(W_{t+1}) \underbrace{\frac{\partial W_{t+1}}{\partial W_t}}_{1}$$

3. Combine and plug back into FOC

$$u'(c_t^\star) = V'(W_t) \Rightarrow u'(c_{t+1}^\star) = V'(W_{t+1}) \Rightarrow$$

$$u'(c_t^\star) = \beta u'(c_{t+1}^\star)$$

# VFI with non-linear optimizer or solver

1. Discretize state space with $\vec{W} \in \{0, \dots \overline{W}\}$
2. Start with an arbitrary guess $V_0(\vec{W})$ or $c_0(\vec{W})$
3. At each iteration $i$ and for each point $W_j$ on grid $\vec{W}$ compute

$$V_i(W_j) = \max_{0 \le c \le W_j} \{u(c) + \beta V_{i-1}(W_j - c)\}$$

or solve

$$u'\big(c_i(W_j)\big) = \beta u'\big(c_{i-1}(W_j - c)\big)$$

4. Repeat until convergence

$$\max_j \big(V_i(W_j) - V_{i-1}(W_j)\big) \le \varepsilon \text{ (small number)}$$

$$\max_j \big(c_i(W) - c_{i-1}(W)\big) \le \varepsilon$$

# Cake eating: Numerical implementation III

The code:
**cake3.ipynb**

## Questions and discussion

1. Did convergence speed change?
2. Did magnitude of numerical errors change?
3. Any ideas about how to speed up?

# Only a primer in dynamic programming

**Many topics and extensions left out:**

1. Finite horizon formulation
    - Time scripts essential $V_t(W_t)$ and $c_t(W_t)$
    - Final period $T$ with $c_T(W_T) = W_T$ and $V_T(W_T) = u(W_T)$
    - Convergence replaced with backward calculation until $t = 0$
2. Stochastic models
    - Stochastic evolution of state space, controlled by decisions: random returns, controlled and exogenous Markov processes
    - Idiosyncratic random components in preferences (taste shocks)
    - Need for numerical integration in Bellman equation

# Only a primer in dynamic programming

**Many topics and extensions left out:**

3. Higher dimension
    - Multiple state variables
    - Multi-dimensional decisions
    - Need for high power computing methods

4. Alternative (faster) solution approaches
    - Policy function iterations, faster convergence
    - Polyalgorithm using VFI and Newton-Raphson iteration for robust speed up
    - Endogenous gridpoint methods to avoid root-finding operations in models of applicable class

Thank you!

Questions?