BASICS OF DYNAMIC PROGRAMMING (CONTINUED)

JANUARY 26, 2012

RECURSIVE REPRESENTATION

- **State variables**
  - **A sufficient summary, as of the start of period *t*, of the dynamic position of the environment in which the maximizing agent operates**
  - **"Environment" of the agent – what needs to be known in order to optimize in period *t*?**
    - **Individual-specific quantities**
    - **Market prices**
    - **Government policies**
    - **(Fixed structural parameters – will omit from state vector for parsimony)**

The usual suspects

Important: states can be endogenous or exogenous

- **Ljungqvist and Sargent (2004, p. 16)**

  "The art in applying recursive methods is to find a convenient definition of a state. It is often not obvious what the state is, or even whether a finite-dimensional state exists."

Recursive Representation

- **State variables**
  - **A sufficient summary, as of the start of period *t*, of the dynamic position of the environment in which the maximizing agent operates**
  - **"Environment" of the agent – what needs to be known in order to optimize in period *t*?**
    - **Individual-specific quantities**
    - **Market prices**
    - **Government policies**
    - **(Fixed structural parameters – will omit from state vector for parsimony)**

    The usual suspects

    Important: states can be endogenous or exogenous
  - **"Sufficient" – there are no other objects (quantities, prices, govt policies, etc.) that must be known in order to optimize in period *t***
  - **Concept well-defined for both finite-*T* and *T* → ∞ problems**
  - **KEY: Period-*t* decisions are function of the period-*t* state variables**
- **Ljungqvist and Sargent (2004, p. 16)**

  "The art in applying recursive methods is to find a convenient definition of a state. It is often not obvious what the state is, or even whether a finite-dimensional state exists."

Bellman Equation

- **Bellman Equation**

$$V^0(a_{-1},r_0;.) \equiv \max_{c_0,a_0} \left\{ u(c_0) + \lambda_0 \left(y_0 + (1+r_{-1})a_{-1} - c_0 - a_0 \right) + \beta \cdot V^1(a_0,r_1;.) \right\}$$

  - **Starting point for recursive analysis**
  - **Applicable to finite *T*-period or *T* → ∞ problems**
  - **Construction requires identifying state variables**
- ***T*-period problem**
  - **Solution involves sequence of functions $V^0(.)$, $V^1(.)$, … , $V^{T-1}(.)$, $V^T(.)$**
  - **$V^i(.)$ functions in general will differ – reflecting time until end of planning horizon**
  - **E.g., maximized value starting from age = 60 different from maximized value starting from age = 30 (intuitively)**
- **Infinite-horizon problem ("stationary" environment)**
  - **Deterministic case: $V(.) \equiv V^i(.) = V^j(.)\ \forall\, i,j$**
  - **Always an infinity of periods left to go**

Bellman Equation

- **Bellman Equation (for $T \to \infty$)**

$$V(a_{-1}, r_0;.) \equiv \max_{c_0, a_0} \left\{ u(c_0) + \lambda_0 \left(y_0 + (1+r_{-1})a_{-1} - c_0 - a_0 \right) + \beta \cdot V(a_0, r_1;.) \right\}$$

- **Use to characterize optimal decisions**

- **Period-0 FOCs, evaluated using time-invariant $c(a_{-1})$, $a(a_{-1})$**

$$\left.\begin{array}{ll} \mathbf{c_0}: & u'(c(a_{-1})) - \lambda_0 = 0 \\ \mathbf{a_0}: & -\lambda_0 + \beta V_1(a_0(a_{-1}), r_1;.) = 0 \\ \mathbf{Env}: & V_1(a(a_{-1}), r_1;.) = \lambda_1(1+r_0) \end{array}\right\} \; u'(c(a_{-1})) = \beta(1+r_0)u'(c(a_0))$$

- **Seems like usual Euler equation from sequential analysis (deterministic)…**

Bellman Equation

- **Bellman Equation (for $T \to \infty$)**

$$V(a_{-1}, r_0;.) \equiv u(c(a_{-1})) + \lambda_0 \left(y_0 + (1+r_{-1})a_{-1} - c(a_{-1}) - a(a_{-1}) \right) + \beta \cdot V(a(a_{-1}), r_1;.)$$

- **Seems like a two-period problem**
  - **In terms of (value) functions, not in terms of choice variables**
  - **Optimize in current period**
  - **Optimize next period (Bellman's Principle of Optimality)**

Bellman Equation

- **Bellman Equation (for $T \to \infty$)**

$$V(a_{-1}, r_0;.) \equiv u(c(a_{-1})) + \lambda_0 \left(y_0 + (1+r_{-1})a_{-1} - c(a_{-1}) - a(a_{-1})\right) + \beta \cdot V(a(a_{-1}), r_1;.)$$

- **Seems like a two-period problem**
  - **In terms of (value) functions, not in terms of choice variables**
  - **Optimize in current period**
  - **Optimize next period (Bellman's Principle of Optimality)**
- **Common notation**
  - **Use *x* for current-period variables**
  - **Use *x'* for next-period variables**
- **Bellman Equation**

$$V(a, r;.) \equiv u(\underbrace{c(a)}_{=c}) + \lambda\left(y + (1+r_{-1})a - \underbrace{c(a)}_{=c} - \underbrace{a(a)}_{=a'}\right) + \beta \cdot V(\underbrace{a(a)}_{=a'}, r';.)$$

- **Euler equation**

$$u'(\overbrace{c(a)}^{=c}) = \beta(1+r)u'(\overbrace{c(a')}^{=c'})$$

Recursive vs. Sequential Analysis

- **So why go recursive?**
  - **Allows application of series of theorems/results that guarantee a solution exists in the space of functions**
  - **Allows application of series of theorems/results that help find solution in the space of functions**

Underlying Theory:
Contraction Mapping Theorem, Blackwell's Sufficient Conditions for a Contraction, Theorem of the Maximum

- **Suppose *V*(.) exists**
- **Procedure for finding *V*(.) and associated decision rules: iterate on Bellman Equation starting from any arbitrary initial guess**

Recursive vs. Sequential Analysis

- So why go recursive?
  - Allows application of series of theorems/results that guarantee a solution exists in the space of functions
  - Allows application of series of theorems/results that help find solution in the space of functions

Underlying Theory:
Contraction Mapping Theorem, Blackwell's Sufficient Conditions for a Contraction, Theorem of the Maximum

- Suppose *V*(.) exists
- Procedure for finding *V*(.) and associated decision rules: iterate on Bellman Equation starting from any arbitrary initial guess – call it $V^1(.)$

initial guess (some parametric form)

$$V(a,r;.) \equiv \max_{c,a'} \left\{ u(c) + \lambda\left(y + (1+r)a - c - a'\right) + \beta \cdot V^1(a',r';.) \right\}$$

Recursive vs. Sequential Analysis

- So why go recursive?
  - Allows application of series of theorems/results that guarantee a solution exists in the space of functions
  - Allows application of series of theorems/results that help find solution in the space of functions

Underlying Theory:
Contraction Mapping Theorem, Blackwell's Sufficient Conditions for a Contraction, Theorem of the Maximum

- Suppose *V*(.) exists
- Procedure for finding *V*(.) and associated decision rules: iterate on Bellman Equation starting from any arbitrary initial guess – call it $V^1(.)$

initial guess (some parametric form)

$$V(a,r;.) \equiv \max_{c,a'} \left\{ u(c) + \lambda\left(y + (1+r)a - c - a'\right) + \beta \cdot V^1(a',r';.) \right\}$$

- Conduct maximization
  - Gives functions $c(a)$ and $a(a)$
  - These are candidate (optimal) decision rules
- Insert candidate $c(a)$ and $a(a)$ into RHS of Bellman Equation – generates $V^2(.)$

Recursive vs. Sequential Analysis

- So why go recursive?
  - Allows application of series of theorems/results that guarantee a solution exists in the space of functions
  - Allows application of series of theorems/results that help find solution in the space of functions

Underlying Theory:
Contraction Mapping Theorem, Blackwell's Sufficient Conditions for a Contraction, Theorem of the Maximum

- Suppose $V(.)$ exists
- Procedure for finding $V(.)$ and associated decision rules: iterate on Bellman Equation starting from any arbitrary initial guess – call it $V^1(.)$

initial guess (some parametric form)

$$V(a,r;.) \equiv \max_{c,a'} \left\{ u(c) + \lambda \left(y + (1+r)a - c - a' \right) + \beta \cdot V^1(a',r';.) \right\}$$

  - Conduct maximization
    - Gives functions $c(a)$ and $a(a)$
    - These are candidate (optimal) decision rules
  - Insert candidate $c(a)$ and $a(a)$ into RHS of Bellman Equation – generates $V^2(.)$

Does $V^2(.) = V^1(.)$?

Dynamic Programming

Recursive vs. Sequential Analysis

- So why go recursive?
  - Allows application of series of theorems/results that guarantee a solution exists in the space of functions
  - Allows application of series of theorems/results that help find solution in the space of functions

Underlying Theory:
Contraction Mapping Theorem, Blackwell's Sufficient Conditions for a Contraction, Theorem of the Maximum

- Suppose $V(.)$ exists
- Procedure for finding $V(.)$ and associated decision rules: iterate on Bellman Equation starting from any arbitrary initial guess – call it $V^1(.)$

initial guess (some parametric form)

$$V(a,r;.) \equiv \max_{c,a'} \left\{ u(c) + \lambda \left(y + (1+r)a - c - a' \right) + \beta \cdot V^1(a',r';.) \right\}$$

  - Conduct maximization
    - Gives functions $c(a)$ and $a(a)$
    - These are candidate (optimal) decision rules

If no, insert $V^2(.)$ on RHS and repeat

  - Insert candidate $c(a)$ and $a(a)$ into RHS of Bellman Equation – generates $V^2(.)$

Does $V^2(.) = V^1(.)$? If yes, stop. Have found $V(.)$ ($= V^2(.) = V^1(.)$)

RECURSIVE VS. SEQUENTIAL ANALYSIS

- So why go recursive?
  - Allows application of series of theorems/results that guarantee a solution exists in the space of functions
  - Allows application of series of theorems/results that help find solution in the space of functions

Underlying Theory:
Contraction Mapping Theorem, Blackwell's Sufficient Conditions for a Contraction, Theorem of the Maximum

e.g., value function iteration

  - Computational algorithms require it – computers can't handle infinite-dimensional objects!
    - Soon: simple computational algorithms
- Can't "choose" whether to analyze problem sequentially or recursively
  - All but the most limited of problems require computational solution
  - In which case model analysis is recursive

Dynamic Programming

RECURSIVE VS. SEQUENTIAL ANALYSIS

- So why go recursive?
  - Allows application of series of theorems/results that guarantee a solution exists in the space of functions
  - Allows application of series of theorems/results that help find solution in the space of functions

Underlying Theory:
Contraction Mapping Theorem, Blackwell's Sufficient Conditions for a Contraction, Theorem of the Maximum

e.g., value function iteration

  - Computational algorithms require it – computers can't handle infinite-dimensional objects!
    - Soon: simple computational algorithms
- Can't "choose" whether to analyze problem sequentially or recursively
  - All but the most limited of problems require computational solution
  - In which case model analysis is recursive
- "Solving model sequentially"
  - Doesn't seem recursive… $\longrightarrow$ $u(c_t) = \beta(1+r_t)u'(c_{t+1})$
  - …but computational implementation requires time-invariant decision rule

Imposing recursivity on solution

$$u(c(a_{t-1})) = \beta(1+r_t)u'(c(a_t))$$

RECURSIVE VS. SEQUENTIAL ANALYSIS

- So why go recursive?
  - Allows application of series of theorems/results that guarantee a solution exists in the space of functions
  - Allows application of series of theorems/results that help find solution in the space of functions

Underlying Theory:
Contraction Mapping Theorem, Blackwell's Sufficient Conditions for a Contraction, Theorem of the Maximum

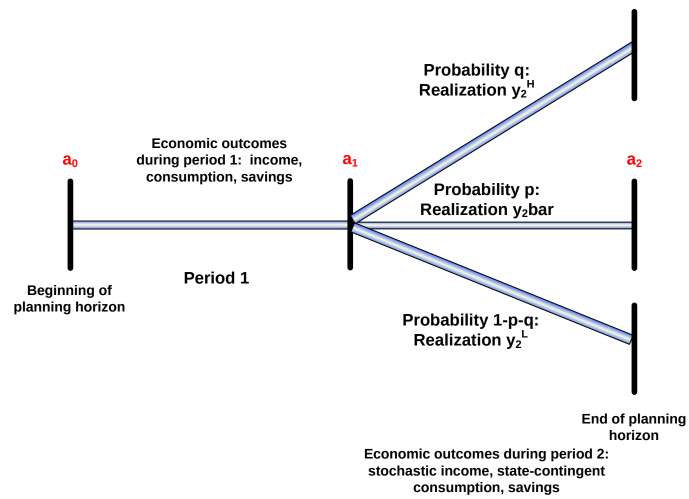
e.g., value function iteration

  - Computational algorithms require it – computers can't handle infinite-dimensional objects!
    - Soon: simple computational algorithms
- Can't "choose" whether to analyze problem sequentially or recursively
  - All but the most limited of problems require computational solution
  - In which case model analysis is recursive
- What about stochastic dynamic programming?
  - Even more structure required....
  - The key assumption is Markov risk

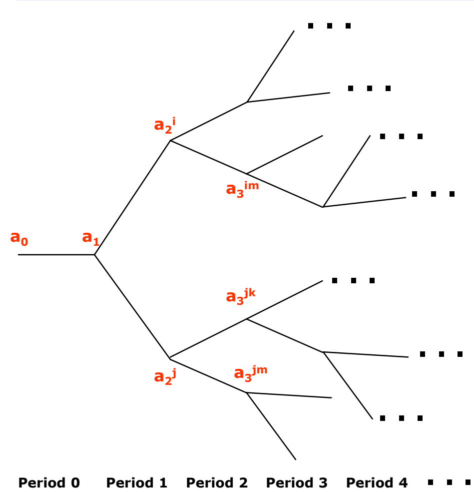
STOCHASTIC DYNAMIC PROGRAMMING

- Even more structure required on the problem to recursively solve dynamic stochastic optimization problems
- Main (new) technical problem
  - Branching of event tree at each of T periods (possibly $T \rightarrow \infty$)
- Main technical solution/assumption
  - Assume risk follows Markov process
  - Which enables series of theorems/results from deterministic dynamic programming to work in stochastic case…
  - …given further technical regularity assumptions

Event Tree



Event Tree



- **In general could be any arbitrary unfolding of exogenous risk**

  e.g., if state *i* realized in period 16, then 14 possible states in period 17; but if state *j* realized in period 16, then 8 possible states in period 17

  OR

  e.g., probability of state *i* in period *t* depends on event in period *t*-100000

- **Number of decision rules to solve explodes**
- **Intractable!!!**
- **"Curse of dimensionality"**
- **Requires a lot of structure on exogenous risk**

RISK STRUCTURE

Assumptions

- **Set of realizations of exogenous state variable is independent of date**

$$S2 = S3 = S4 = S5 = \dots = ST\text{-}1 = ST$$

RISK STRUCTURE

Assumptions

- **Set of realizations of exogenous state variable is independent of date**
- **Probability of realization of exogenous state variable in period *t* depends only on outcomes in period *t*-1**
  - **Suppose X_t is a stochastic process and x_t is a particular realization**
  - **X_t is a Markov process if**

$$\Pr\left(X_t = x_t \mid X_{t-1} = x_{t-1}, X_{t-2} = x_{t-2}, X_{t-3} = x_{t-3}, \dots, X_{t-10000} = x_{t-10000}, \dots\right)$$

$$= \Pr\left(X_t = x_t \mid X_{t-1} = x_{t-1}\right)$$ **CONDITIONAL probability depends on only *t*-1**

Risk Structure

Assumptions

- **Set of realizations of exogenous state variable is independent of date**
- **Probability of realization of exogenous state variable in period *t* depends only on outcomes in period *t*-1**
  - **Suppose X_t is a stochastic process and x_t is a particular realization**
  - **X_t is a Markov process if**

$$\Pr\left(X_t = x_t \mid X_{t-1} = x_{t-1}, X_{t-2} = x_{t-2}, X_{t-3} = x_{t-3}, \ldots, X_{t-10000} = x_{t-10000}, \ldots\right)$$

$$= \Pr\left(X_t = x_t \mid X_{t-1} = x_{t-1}\right)$$ **CONDITIONAL probability depends on only *t*-1**

  - **Not as restrictive as it may seem – could have finite lags in process**
  - **E.g.**
  - **Just can't have infinite lags (in principle) or "too many" (finite) lags (in computational practice)**

Risk Structure

Assumptions

- **Set of realizations of exogenous state variable is independent of date**
- **Probability of realization of exogenous state variable in period *t* depends only on outcomes in period *t*-1**
  - **Suppose X_t is a stochastic process and x_t is a particular realization**
  - **X_t is a Markov process if**

$$\Pr\left(X_t = x_t \mid X_{t-1} = x_{t-1}, X_{t-2} = x_{t-2}, X_{t-3} = x_{t-3}, \ldots, X_{t-10000} = x_{t-10000}, \ldots\right)$$

$$= \Pr\left(X_t = x_t \mid X_{t-1} = x_{t-1}\right)$$ **CONDITIONAL probability depends on only *t*-1**

  - **Not as restrictive as it may seem – could have finite lags in process**

- **Exogenous state variable is Markov process + assumption/result that decision rules are time-invariant (for $T \to \infty$) functions of state variables**

$\Rightarrow$ **Endogenous processes are Markov**
given several regularity assumptions

Underlying theory:
Stokey, Lucas, Prescott (1989, Chapters 8-12)

STOCHASTIC – SEQUENTIAL ANALYSIS

- **Planning horizon $T \to \infty$**
- **Exogenous state drawn from set S (could be continuous or discrete)**
- **Suppose single asset with state-contingent r (will illustrate main ideas)**

$$\max_{\{c_t, a_t\}_{t=0}^{T}} E_0 \sum_{t=0}^{T} \beta^t u(c_t) \quad \textbf{subject to} \quad \begin{cases} c_t + a_t = y_t + (1+r_t)a_{t-1}, \quad t = 0,1,2,...T \\ \textbf{state-contingent budget constraints in } t > 0 \end{cases}$$

- **FOCs**

c_0: $u'(c_0) - \lambda_0 = 0$

a_0:

c_1: $\beta E_0 u'(c_1) - \beta E_0 \lambda_1 = 0$ → **Holds for each state** → $E_0 u'(c_1^j) = E_0 \lambda_1^j, \; \forall j \in S$

Model Analysis

STOCHASTIC – SEQUENTIAL ANALYSIS

- **Planning horizon $T \to \infty$**
- **Exogenous state drawn from set S (could be continuous or discrete)**
- **Suppose single asset with state-contingent r (will illustrate main ideas)**

$$\max_{\{c_t, a_t\}_{t=0}^{T}} E_0 \sum_{t=0}^{T} \beta^t u(c_t) \quad \textbf{subject to} \quad \begin{cases} c_t + a_t = y_t + (1+r_t)a_{t-1}, \quad t = 0,1,2,...T \\ \textbf{state-contingent budget constraints in } t > 0 \end{cases}$$

- **FOCs**

c_0: $u'(c_0) - \lambda_0 = 0$

a_0: $-\lambda_0 + \beta E_0\left[\lambda_1(1+r_1)\right] = 0$ → **Just as in stochastic two-period model** → $1 = E_0\left[\frac{\beta\lambda_1}{\lambda_0}(1+r_1)\right]$

c_1: $\beta E_0 u'(c_1) - \beta E_0 \lambda_1 = 0$ → **Holds for each state** → $E_0 u'(c_1^j) = E_0 \lambda_1^j, \; \forall j \in S$

Model Analysis

STOCHASTIC – SEQUENTIAL ANALYSIS

- **Planning horizon $T \rightarrow \infty$**
- **Exogenous state drawn from set S (could be continuous or discrete)**
- **Suppose single asset with state-contingent r (will illustrate main ideas)**

$$\max_{\{c_t,a_t\}_{t=0}^{T}} E_0 \sum_{t=0}^{T} \beta^t u(c_t) \quad \textbf{subject to} \quad \begin{cases} c_t + a_t = y_t + (1+r_t)a_{t-1}, \quad t = 0,1,2,\ldots T \\ \text{state-contingent budget constraints in } t > 0 \end{cases}$$

- **FOCs**

c_1: $\beta E_0 u'(c_1) - \beta E_0 \lambda_1 = 0 \longrightarrow$ Holds for each state $\quad E_0 u'(c_1^j) = E_0 \lambda_1^j, \ \forall j \in S$

a_1: $-\beta E_0 \lambda_1 + \beta^2 E_0 \left[\lambda_2 (1+r_2) \right] = 0$

c_2: $\beta^2 E_0 u'(c_2) - \beta^2 E_0 \lambda_2 = 0 \longrightarrow$ Holds for each state $\quad E_0 u'(c_2^j) = E_0 \lambda_2^j, \ \forall j \in S$

Model Analysis

STOCHASTIC – MARKOV SOLUTION

- **$\{X_t\}_{t=0,1,2,\ldots}$ is Markov process (exogenous and endogenous states)**
  - **Nothing about the probability distribution of X_{t+2} is known in period t that is not known in period $t+1$**
    - **Information set of period t+1 is superset of information set of period t**
- **Allows applying a law of iterated expectations**
  - **$E_t X_{t+2} = E_t [E_{t+1} X_{t+2}]$**

$$E_0 \lambda_1 = \beta E_0 \left[\lambda_2 (1+r_2) \right] \longrightarrow E_0 \lambda_1 = \beta E_0 \left[E_1 \left(\lambda_2 (1+r_2) \right) \right]$$

Stochastic – Markov Solution

- $\{X_t\}_{t=0,1,2,...}$ is Markov process (exogenous and endogenous states)
  - Nothing about the probability distribution of X_{t+2} is known in period t that is not known in period $t+1$
    - Information set of period $t+1$ is superset of information set of period t
- Allows applying a law of iterated expectations
  - $E_t X_{t+2} = E_t [E_{t+1} X_{t+2}]$

$$E_0\lambda_1 = \beta E_0\left[\lambda_2(1+r_2)\right] \longrightarrow E_0\lambda_1 = \beta E_0\left[E_1\left(\lambda_2(1+r_2)\right)\right]$$

- Date- and state-contingent decisions: decisions governed by this Euler condition are conditional on information set of period 1 (i.e., recursivity)

$$\longrightarrow E_1\lambda_1 = \beta E_1\left[\lambda_2(1+r_2)\right] \longrightarrow \lambda_1 = \beta E_1\left[\lambda_2(1+r_2)\right]$$

Stochastic – Sequential Analysis

- Planning horizon $T \to \infty$
- Exogenous state drawn from set S (could be continuous or discrete)
- Suppose single asset with state-contingent r (will illustrate main ideas)

$$\max_{\{c_t,a_t\}_{t=0}^{T}} E_0\sum_{t=0}^{T}\beta^t u(c_t) \text{ subject to } \begin{cases} c_t + a_t = y_t + (1+r_t)a_{t-1}, \quad t = 0,1,2,...T \\ \text{state-contingent budget constraints in } t > 0 \end{cases}$$

- FOCs

c_1: $\beta E_0 u'(c_1) - \beta E_0\lambda_1 = 0 \longrightarrow$ Holds for each state $\longrightarrow E_1 u'(c_1^j) = E_1\lambda_1^j, \ \forall j \in S$

a_1: $-\beta E_0\lambda_1 + \beta^2 E_0\left[\lambda_2(1+r_2)\right] = 0 \longrightarrow$ Because Markov and state- and date-contingent decisions $\longrightarrow \lambda_1 = \beta E_1\left[\lambda_2(1+r_2)\right]$

c_2: $\beta^2 E_0 u'(c_2) - \beta^2 E_0\lambda_2 = 0 \longrightarrow$ Holds for each state $\longrightarrow E_2 u'(c_2^j) = E_2\lambda_2^j, \ \forall j \in S$

STOCHASTIC – MARKOV SOLUTION

- Planning horizon $T \to \infty$
- Exogenous state drawn from set S (could be continuous or discrete)
- Suppose single asset with state-contingent r (will illustrate main ideas)

$$\max_{\{c_t, a_t\}_{t=0}^{T}} E_0 \sum_{t=0}^{T} \beta^t u(c_t) \quad \text{subject to} \begin{cases} c_t + a_t = y_t + (1+r_t)a_{t-1}, \quad t = 0,1,2,...T \\ \text{with uncertain realizations in } t > 0 \end{cases}$$

- FOCs

c_t: $\beta^t E_0 u'(c_t) - \beta^t E_0 \lambda_t = 0 \xrightarrow{\text{Holds for each date and state}} u'(c_1^j) = \lambda_1^j, \ \forall j \in S$

a_t: $-\beta^t E_0 \lambda_t + \beta^{t+1} E_0 \left[\lambda_{t+1}(1+r_{t+1})\right] = 0 \xrightarrow{\text{Because Markov and state- and date-contingent decisions}} \lambda_t = \beta E_t \left[\lambda_{t+1}(1+r_{t+1})\right]$

c_{t+1}: $\beta^{t+1} E_0 u'(c_{t+1}) - \beta^{t+1} E_0 \lambda_{t+1} = 0 \xrightarrow{\text{Holds for each date and state}} u'(c_2^j) = \lambda_2^j, \ \forall j \in S$

- One-period-ahead conditional expectation governs stochastic Euler condition

STOCHASTIC – MARKOV SOLUTION

- Denote exogenous state variables as z (e.g., $z_t = [y_t, r_t]$)
- Solution of infinite-horizon consumer problem is a consumption decision rule $c(a, z;.)$, asset decision rule $a(a, z;.)$, and value function $V(a, z;.)$ that satisfies

Stochastic – Markov Solution

- **Denote exogenous state variables as z (e.g., $z_t = [y_t, r_t]$)**
- **Solution of infinite-horizon consumer problem is a consumption decision rule $c(a, z;.)$, asset decision rule $a(a, z;.)$, and value function $V(a, z;.)$ that satisfies**
  - **(Stochastic) Euler equation**

    $$u'(c(a,z)) = \beta E\left[u'(c(a',z'))(1+r')\right]$$

    - **which is the (expectational) TVC in the limit $t \rightarrow \infty$:**

      $$\lim_{t\to\infty} E_0 \beta^t u'(c(a,z)) \cdot a(a,z) = 0$$

  - **Budget constraint**

    $$y + (1+r)a - c(a,z) - a(a,z) = 0$$

Model Solution

Stochastic – Markov Solution

- **Denote exogenous state variables as z (e.g., $z_t = [y_t, r_t]$)**
- **Solution of infinite-horizon consumer problem is a consumption decision rule $c(a, z;.)$, asset decision rule $a(a, z;.)$, and value function $V(a, z;.)$ that satisfies**
  - **(Stochastic) Euler equation**

    $$u'(c(a,z)) = \beta E\left[u'(c(a',z'))(1+r')\right]$$

    - **which is the (expectational) TVC in the limit $t \rightarrow \infty$:**

      $$\lim_{t\to\infty} E_0 \beta^t u'(c(a,z)) \cdot a(a,z) = 0$$

  - **Budget constraint**

    $$y + (1+r)a - c(a,z) - a(a,z) = 0$$

Expectation in Bellman Equation **Transition from $z \rightarrow z'$**

  - **Bellman Equation**

$$V(a,z;.) \equiv u(c(a,z)) + \lambda\big(y + (1+r)a - c(a,z) - a(a,z)\big) + \beta \cdot EV(a(a,z), z(a,z);.)$$

taking as given (y, a, r) and (Markov) transition function for $z \rightarrow z'$

BELLMAN EQUATION

- **Bellman Equation (for $T \to \infty$)**

$$V(a,z;.) \equiv \max_{c,a'} \left\{ u(c) + \lambda\left(y + (1+r)a - c - a'\right) + \beta \cdot EV(a',z';.) \right\}$$

Expectation in Bellman Equation

Transition from $z \to z'$

- **Use to characterize optimal decisions**
- **Current-period FOCs, evaluated using $c(a,z;.)$, $a(a,z;.)$**

c:

a':

Env:

Stochastic Dynamic Programming

BELLMAN EQUATION

- **Bellman Equation (for $T \to \infty$)**

$$V(a,z;.) \equiv \max_{c,a'} \left\{ u(c) + \lambda\left(y + (1+r)a - c - a'\right) + \beta \cdot EV(a',z';.) \right\}$$

Expectation in Bellman Equation

Transition from $z \to z'$

- **Use to characterize optimal decisions**
- **Current-period FOCs, evaluated using $c(a,z;.)$, $a(a,z;.)$**

c: $u'(c(a,z)) - \lambda = 0$

a': $-\lambda + \beta EV_1(a(a,z), z(a,z);.) = 0$

Env: $EV_1(a,z;.) = \lambda(1+r)$

$$u'(c(a,z)) = \beta E\left[u'(c(a,z))(1+r)\right]$$

- **Bellman analysis goes through as in deterministic case**
  - **(Given further technical conditions we won't study – see SLP)**

Markov Risk

- Why does Markov assumption make everything work?
- Main issue in moving from deterministic dynamic programming to stochastic dynamic programming: preserving recursivity
  - So exogenous states must also have recursive structure
- Shocks that have this recursive structure are Markov processes
- Markov has property that given the current realization, future realizations are independent of the past
  - "Limited history dependence"
  - "Finite memory"
- In environments in which the "regularity conditions" that ensure standard Bellman analysis applies to stochastic problems are not satisfied…
- …often simply need to ASSUME decision rules are Markov to make progress