Numerical Methods for Time Inconsistency, Private Information and Limited Commitment

Antonio Mele

University of Surrey

July 2017

WHY NUMERICAL METHODS?

WHY NUMERICAL METHODS?

- "Can't I just work with my beloved pencil?"

WHY NUMERICAL METHODS?

- “Can’t I just work with my beloved pencil?”
- Models became more and more complicated

WHY NUMERICAL METHODS?

- "Can't I just work with my beloved pencil?"
- Models became more and more complicated
- In macro, use of numerical solutions is the default option now

WHY NUMERICAL METHODS?

- "Can't I just work with my beloved pencil?"
- Models became more and more complicated
- In macro, use of numerical solutions is the default option now
- Micro is getting there: dynamic games, IO applications, auctions,...

WHY NUMERICAL METHODS?

- "Can't I just work with my beloved pencil?"
- Models became more and more complicated
- In macro, use of numerical solutions is the default option now
- Micro is getting there: dynamic games, IO applications, auctions,...
- In summary: every economist nowadays need to know some basic numerical techniques

ROAD MAP

- Dynamic programming in theory

ROAD MAP

- Dynamic programming in theory
- Dynamic programming in practice

ROAD MAP

- Dynamic programming in theory
- Dynamic programming in practice
    - Value function iteration

ROAD MAP

- Dynamic programming in theory
- Dynamic programming in practice
    - Value function iteration
    - Policy function iteration

ROAD MAP

- Dynamic programming in theory
- Dynamic programming in practice
  - Value function iteration
  - Policy function iteration
- Basic numerical techniques

ROAD MAP

- Dynamic programming in theory
- Dynamic programming in practice
    - Value function iteration
    - Policy function iteration
- Basic numerical techniques
    - Integrals, nonlinear equations, optimization

ROAD MAP

- Dynamic programming in theory
- Dynamic programming in practice
    - Value function iteration
    - Policy function iteration
- Basic numerical techniques
    - Integrals, nonlinear equations, optimization
- Projection methods

ROAD MAP

- Dynamic programming in theory
- Dynamic programming in practice
    - Value function iteration
    - Policy function iteration
- Basic numerical techniques
    - Integrals, nonlinear equations, optimization
- Projection methods
- Models with time inconsistency

ROAD MAP

- Dynamic programming in theory
- Dynamic programming in practice
    - Value function iteration
    - Policy function iteration
- Basic numerical techniques
    - Integrals, nonlinear equations, optimization
- Projection methods
- Models with time inconsistency
    - Promised utilities approach

ROAD MAP

- Dynamic programming in theory
- Dynamic programming in practice
    - Value function iteration
    - Policy function iteration
- Basic numerical techniques
    - Integrals, nonlinear equations, optimization
- Projection methods
- Models with time inconsistency
    - Promised utilities approach
    - Lagrangean approach

MATERIAL

- Lecture notes

MATERIAL

- Lecture notes
- Slides

MATERIAL

- Lecture notes
- Slides
- Codes (available at **this online repository**)

ADDITIONAL MATERIAL

- Dynamic programming (and much more!):
  - [LS] Ljungqvist, L., and T. J. Sargent (2004), *Recursive Macroeconomic Theory*, Second Edition, MIT Press
  - [AC] Adda J., and Cooper (2003) *Dynamic Economics: Quantitative Methods and Applications*, MIT Press
  - [SLP] Stokey, N., R. Lucas and E. Prescott (1989), *Recursive Methods for Economic Dynamics*, Harvard University Press

Additional material

- Dynamic programming (and much more!):
  - [LS] Ljungqvist, L., and T. J. Sargent (2004), *Recursive Macroeconomic Theory*, Second Edition, MIT Press
  - [AC] Adda J., and Cooper (2003) *Dynamic Economics: Quantitative Methods and Applications*, MIT Press
  - [SLP] Stokey, N., R. Lucas and E. Prescott (1989), *Recursive Methods for Economic Dynamics*, Harvard University Press
- Books on numerical methods:
  - [Judd] Judd K. (1998), *Numerical Methods in Economics*, MIT Press
  - [MF] Miranda, M. J. and P. L. Fackler (2002), *Applied Computational Economics and Finance*, MIT Press

NEOCLASSICAL GROWTH MODEL

We can present main ideas of DP by using the neoclassical growth model

Production function is

$$y_t = F\left(k_t, n_t\right)$$

where $F : R_+^2 \to R_+$ continuously differentiable, strictly increasing, homogeneous of degree 1 and strictly quasi-concave, with

$$F_1 > 0; F_{11} < 0; F_2 > 0; F\left(0, n\right) = 0 \qquad \forall k, n > 0$$

$$\lim_{k \to 0} F_k\left(k, 1\right) = \infty, \lim_{k \to \infty} F_k\left(k, 1\right) = 0$$

AN EXAMPLE: NEOCLASSICAL GROWTH MODEL (CONT.)

Labor force equal 1:

$$0 \leq n_t \leq 1 \quad \forall t \tag{1}$$

Resource constraint:

$$c_t + k_{t+1} - (1-\delta) k_t \leq F(k_t, n_t) \tag{2}$$

HOUSEHOLDS

- Identical households, additively separable preferences

$$U(c_0, c_1, ...) = \sum_{t=0}^{\infty} \beta^t u(c_t) \tag{3}$$

with $\beta \in (0,1)$, $u : R_+ \to R$ is bounded, continuously differentiable, strictly increasing, strictly concave and $\lim_{c \to 0} u'(c) = \infty$.

HOUSEHOLDS

- Identical households, additively separable preferences

$$U(c_0, c_1, \ldots) = \sum_{t=0}^{\infty} \beta^t u(c_t) \tag{3}$$

with $\beta \in (0,1)$, $u : R_+ \to R$ is bounded, continuously differentiable, strictly increasing, strictly concave and $\lim_{c \to 0} u'(c) = \infty$.

- $\Rightarrow$ Representative household

HOUSEHOLDS

- Identical households, additively separable preferences

$$U(c_0, c_1, \ldots) = \sum_{t=0}^{\infty} \beta^t u(c_t) \tag{3}$$

with $\beta \in (0,1)$, $u : R_+ \to R$ is bounded, continuously differentiable, strictly increasing, strictly concave and $\lim_{c \to 0} u'(c) = \infty$.

- $\Rightarrow$ Representative household
- Leisure has no value.
- No uncertainty

First welfare theorem

- No market failures

FIRST WELFARE THEOREM

- No market failures
- First welfare theorem applies
- Competitive equilibrium is Pareto efficient

FIRST WELFARE THEOREM

- No market failures
- First welfare theorem applies
- Competitive equilibrium is Pareto efficient
- Benevolent social planner problem: maximize HH discounted utility subject to resource constraint and labor force constraint, k_0 given
- No waste of output and no value of leisure $\Rightarrow$

$$c_t + k_{t+1} - (1-\delta)k_t = F(k_t, n_t)$$

$$n_t = 1$$

Another way of writing it...

Define:

$$f(k) \equiv F(k,1) + (1-\delta)k$$

Another way of writing it...

Define:

$$f(k) \equiv F(k,1) + (1-\delta)k$$

$\Rightarrow$ Social planner problem is

$$\max_{\{k_{t+1}\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t u\left(f(k_t) - k_{t+1}\right) \tag{4}$$

$$s.t. \quad 0 \leq k_{t+1} \leq f(k_t), \quad t = 0, 1, \ldots \tag{5}$$

k_0 given

FINITE HORIZON

FINITE HORIZON

- Imagine we have finite horizon T
- Standard concave program in $\{k_{t+1}\}_{t=0}^{T}$
  - Objective fct in (4) is strictly concave and continuous
  - Constraint (5) defines a closed, bounded and convex set

FINITE HORIZON

- Imagine we have finite horizon T
- Standard concave program in $\{k_{t+1}\}_{t=0}^{T}$
  - Objective fct in (4) is strictly concave and continuous
  - Constraint (5) defines a closed, bounded and convex set
- Unique solution
- Kuhn-Tucker conditions:

$$\beta f'(k_t)\, u'(f(k_t) - k_{t+1}) = u'(f(k_{t-1}) - k_t) \tag{6}$$

and $k_{T+1} = 0, \quad k_0 > 0$ given.

Infinite horizon

How to solve it?

Infinite horizon

How to solve it?

- First idea: Solve finite horizon, take the limit ($T \to \infty$)

Infinite horizon

How to solve it?

- First idea: Solve finite horizon, take the limit ($T \to \infty$)
    - Complicated to prove it

Infinite horizon

How to solve it?

- First idea: Solve finite horizon, take the limit ($T \to \infty$)
  - Complicated to prove it
- Guess: $k_{t+1} = g(k_t)$ with $g : R_+ \to R_+$.

Infinite horizon

How to solve it?

- First idea: Solve finite horizon, take the limit ($T \to \infty$)
  - Complicated to prove it
- Guess: $k_{t+1} = g(k_t)$ with $g : R_+ \to R_+$.
  - Intuition: In each period, the planning problem is always the same, and only the capital that we have at the beginning of each period changes, so the choice of future capital stock and consumption must be a function of current capital stock.

Value function

Imagine we already solved the problem for any k_0. Define

$$V(k_0) \equiv \max_{\{k_{t+1}\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t u(f(k_t) - k_{t+1})$$

$$s.t. \quad 0 \leq k_{t+1} \leq f(k_t), \quad t = 0, 1, \ldots$$

The function V is called value function.

Principle of optimality

An optimal policy has the property that whatever the initial state and initial decision are, the remaining decisions must constitute an optimal policy with regard to the state resulting from the first decision.

Bellman, 1957, Chap. III.3.

Recursive formulation

The planner problem can be rewritten as:

$$\max_{c_0,k_1}\left[u(c_0)+\beta \max_{\{0\leq k_{t+1}\leq f(k_t)\}_{t=1}^{\infty}}\sum_{j=0}^{\infty}\beta^j u\left(f\left(k_{t+j}\right)-k_{t+j+1}\right)\right]$$

$$s.t.\quad c_0+k_1\leq f(k_0)$$

$$c_0,k_1\geq 0,\quad k_0>0\quad \text{given}$$

RECURSIVE FORMULATION

The planner problem can be rewritten as:

$$\max_{c_0, k_1} \left[u(c_0) + \beta \underbrace{\max_{\{0 \le k_{t+1} \le f(k_t)\}_{t=1}^{\infty}} \sum_{j=0}^{\infty} \beta^j u\left(f\left(k_{t+j}\right) - k_{t+j+1}\right)}_{V(k_1)} \right]$$

$$s.t. \quad c_0 + k_1 \le f(k_0)$$

$$c_0, k_1 \ge 0, \quad k_0 > 0 \quad \text{given}$$

$V(k_1)$ is the value of the utility from period 1 on, obtained with an initial capital k_1.

Recursive formulation

The planner problem can be rewritten as:

$$\max_{c_0, k_1} \left[u(c_0) + \beta V(k_1) \right] \tag{7}$$

$$s.t. \quad c_0 + k_1 \leq f(k_0)$$

$$c_0, k_1 \geq 0, \quad k_0 > 0 \quad \text{given}$$

$V(k_1)$ is the value of the utility from period 1 on, obtained with an initial capital k_1.

RECURSIVE FORMULATION (ALMOST THERE...)

- If we knew V:

Recursive formulation (almost there...)

- If we knew V:
  - Get g from (7), by letting $k_1 = g(k_0)$ and $c_0 = f(k_0) - g(k_0)$ be the maximizers in (7).
  - Therefore, $k_{t+1} = g(k_t)$ completely describes the dynamics.

RECURSIVE FORMULATION (ALMOST THERE...)

- If we knew V:
  - Get g from (7), by letting $k_1 = g(k_0)$ and $c_0 = f(k_0) - g(k_0)$ be the maximizers in (7).
  - Therefore, $k_{t+1} = g(k_t)$ completely describes the dynamics.
- However,we don't know V! What can we do?

RECURSIVE FORMULATION (HERE WE GO!)

- It must be that

$$V(k_0) = \max_{0 \leq k_1 \leq f(k_0)} \{u(f(k_0 - k_1)) + \beta V(k_1)\}$$

Recursive formulation (here we go!)

- It must be that

$$V(k_0) = \max_{0 \le k_1 \le f(k_0)} \{u(f(k_0 - k_1)) + \beta V(k_1)\}$$

- We don't need time subscripts, this is a static problem!

$$V(k) = \max_{0 \le y \le f(k)} \{u(f(k - y)) + \beta V(y)\} \tag{8}$$

- The unknown is a function (the value function V), it is a *functional equation* or *Bellman equation*

THE BELLMAN EQUATION

The Bellman equation

- In each period, all the information about the past is contained in the current level of capital

THE BELLMAN EQUATION

- In each period, all the information about the past is contained in the current level of capital
- We call this property **stationarity**.
- We call **state variable(s)** the variable(s) that summarize information about the past.

THE BELLMAN EQUATION

- In each period, all the information about the past is contained in the current level of capital
- We call this property **stationarity**.
- We call **state variable(s)** the variable(s) that summarize information about the past.
- The solution is a value function V to which corresponds a maximizer function $y = g(k)$ that we call *policy function*.
- We can use g to calculate the optimal sequence of capital starting from k_0.

A MORE GENERAL PROBLEM

$$\max_{\{u_t\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t r(x_t, u_t) \tag{9}$$

$$s.t. \quad x_{t+1} = h(x_t, u_t), \quad t = 0, 1, \ldots \tag{10}$$

$$x_0 \in X \text{ given}$$

- $x_t \in X$: vector of **state variables**
- $u_t \in U$: vector of **control variables**
- $r(x_t, u_t)$: **return function**

A more general problem (cont.)

We can always rewrite the problem:

$$\max_{\{x_{t+1}\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t f\left(x_t, x_{t+1}\right) \tag{11}$$

$$s.t. \quad x_{t+1} \in \Gamma\left(x_t\right), \quad t = 0, 1, \ldots$$

$$x_0 \in X \text{ given}$$

A MORE GENERAL PROBLEM (CONT.)

We can always rewrite the problem:

$$\max_{\{x_{t+1}\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t f(x_t, x_{t+1}) \tag{11}$$

$$s.t. \quad x_{t+1} \in \Gamma(x_t), \quad t = 0, 1, \dots$$

$$x_0 \in X \text{ given}$$

$\Gamma : X \to X$ is a correspondence (i.e., a mapping from each point of set X to subsets of X) that describes feasibility constraints and the law of motion for state variables.

The general Bellman equation

To this sequential problem we can associate a **Bellman equation**:

$$V(x) = \max_{y \in \Gamma(x)} f(x,y) + \beta V(y) \qquad \forall x \in X \tag{12}$$

We need few assumptions to be sure that the solution of sequential and functional problem are the same.

Assumptions

Assumption 1

$\Gamma(x)$ is nonempty $\forall x \in X$

Assumption 2

For any feasible plan $\{x_t\}_{t=0}^{\infty}$, $\lim_{n\to\infty}\sum_{t=0}^{n}\beta^t f(x_t, x_{t+1})$ exists

Value function

Define:

$$V^*(x_0) \equiv \max_{\{x_{t+1}\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t f(x_t, x_{t+1})$$

$$s.t. \quad x_{t+1} \in \Gamma(x_t), \quad t = 0, 1, \ldots$$

i.e., the **value of the sequential problem**

Same solutions I

Theorem 1

*Let Assumptions 1-2 be satisfied. Then V^* satisfies the **Bellman equation** (12).*

Same solutions I

Theorem 1

Let Assumptions 1-2 be satisfied. Then V^* satisfies the ***Bellman equation*** *(12).*

Theorem 2

Let Assumptions 1-2 be satisfied. If V is a solution to the ***Bellman equation*** *(12) and satisfies*

$$\lim_{n\to\infty} \beta^n V(x_n) = 0$$

for any feasible plan $\{x_t\}_{t=0}^{\infty}$, then $V = V^*$.

SAME SOLUTION II

Other results:

- Under Assumptions 1-2, a feasible plan $\{x_t^*\}_{t=0}^{\infty}$ that attains the maximum in the sequential program (11) is such that

$$V^*\left(x_t^*\right) = f\left(x_t^*, x_{t+1}^*\right) + \beta V^*\left(x_{t+1}^*\right), \quad t = 0, 1, \ldots \tag{13}$$

SAME SOLUTION II

Other results:

- Under Assumptions 1-2, a feasible plan $\{x_t^*\}_{t=0}^{\infty}$ that attains the maximum in the sequential program (11) is such that

$$V^*\left(x_t^*\right) = f\left(x_t^*, x_{t+1}^*\right) + \beta V^*\left(x_{t+1}^*\right), \quad t = 0, 1, \ldots \tag{13}$$

- A partial converse is true: if a feasible plan $\{x_t^*\}_{t=0}^{\infty}$ satisfies equation (13) and $\limsup_{t\to\infty} \beta^t V^*\left(x_t^*\right) \leq 0$, then this plan attains the maximum of the sequential problem.

BOUNDED RETURNS

ASSUMPTION 3

X is a convex subset of $\mathbb{R}^l$, and the correspondence $\Gamma : X \to X$ is non-empty, compact-valued and continuous

ASSUMPTION 4

The function $f(\cdot,\cdot)$ is bounded and continuous, and $0 < \beta < 1$

THE BELLMAN OPERATOR

Define an **operator**, i.e. a mapping from a certain space of functions to another space of functions.

THE BELLMAN OPERATOR

Define an **operator**, i.e. a mapping from a certain space of functions to another space of functions.

- $C(X)$: space of continuous bounded real-valued functions

The Bellman operator

Define an **operator**, i.e. a mapping from a certain space of functions to another space of functions.

- $C(X)$: space of continuous bounded real-valued functions
- $T : C(X) \rightarrow C(X)$ is defined as

$$(TW)(x) \equiv \max_{y \in \Gamma(x)} f(x,y) + \beta W(y) \tag{14}$$

THE BELLMAN OPERATOR

Define an **operator**, i.e. a mapping from a certain space of functions to another space of functions.

- $C(X)$: space of continuous bounded real-valued functions
- $T : C(X) \to C(X)$ is defined as

$$(TW)(x) \equiv \max_{y\in\Gamma(x)} f(x,y) + \beta W(y) \tag{14}$$

- Bellman equation is $V = TV$.

The policy correspondence

We can also define the policy correspondence as

$$G(x) \equiv \{y \in \Gamma(x) : V(x) = f(x,y) + \beta V(y)\}$$

The policy correspondence

We can also define the policy correspondence as

$$G(x) \equiv \{y \in \Gamma(x) : V(x) = f(x,y) + \beta V(y)\}$$

If it is single-valued: policy function (and we call it g)

UNIQUENESS OF V

THEOREM 3

Let Assumptions 1-4 be satisfied. Therefore, there exists a unique function V that solves the Bellman equation (12).

SKETCH OF THE PROOF

- We can write our Bellman equation as:

$$V = TV$$

Sketch of the proof

- We can write our Bellman equation as:

$$V = TV$$

- $\Rightarrow$ Find the fixed point of the operator T

SKETCH OF THE PROOF

- We can write our Bellman equation as:

$$V = TV$$

- $\Rightarrow$ Find the fixed point of the operator T
- Use Contraction Mapping Theorem
  - Show that Blackwell's conditions apply
  - Hence there is a unique value function that solves the Bellman equation

Few more assumptions

Assumption 5

For each y, $f(\cdot, y)$ is strictly increasing in its first arguments

Assumption 6

Γ *is monotone:* $x \leq x' \Rightarrow \Gamma(x) \subseteq \Gamma(x')$

Value function is strictly increasing

Theorem 4

Let Assumptions 3-6 be satisfied, and V be the unique solution of the Bellman equation (12). Then V is strictly increasing.

CRUCIAL ASSUMPTIONS

ASSUMPTION 7

$f(\cdot,\cdot)$ is strictly concave, i.e. $\forall (x,y), (x',y') \in A \quad \forall \alpha \in (0,1)$

$$f\left[\alpha f(x,y) + (1-\alpha)\left(x',y'\right)\right] \geq \alpha f(x,y) + (1-\alpha) f\left(x',y'\right)$$

and with strict inequality if $x \neq x'$.

ASSUMPTION 8

Γ *is convex, i.e.* $\forall \alpha \in [0,1]$ *and* $\forall x, x' \in X$

$$y \in \Gamma(x) \text{ and } y' \in \Gamma\left(x'\right) \Rightarrow$$

$$\alpha y + (1-\alpha) y' \in \Gamma\left[\alpha x + (1-\alpha) x'\right]$$

V is strictly concave and G is a function

Theorem 5

Let Assumptions 3-4 and 7-8 be satisfied. Then V is strictly concave and the policy correspondence G is a continuous, single-valued function.

LIFE IS STOCHASTIC

LIFE IS STOCHASTIC

- Until now: deterministic economy

Life is stochastic

- Until now: deterministic economy
- However, life is stochastic, i.e. there is uncertainty about the possible events in the future
- Examples: consumption-saving decisions, investment plans, financial markets, etc.

LIFE IS STOCHASTIC

- Until now: deterministic economy
- However, life is stochastic, i.e. there is uncertainty about the possible events in the future
- Examples: consumption-saving decisions, investment plans, financial markets, etc.
- Dynamic programming can be easily extended to the stochastic case
- We need assumptions about the probability distribution.

SKIPPING THE DETAILS...

Skipping the details...

- For our purposes, we will just state the extended problem and claim that we can use functional equations to solve it.
- The reader interested in the details can refer to SLP.

UNCERTAINTY

UNCERTAINTY

- Uncertainty: shocks $\{z_t\}$
- For simplicity, possible realizations of $\{z_t\}$ are a finite number

UNCERTAINTY

- Uncertainty: shocks $\{z_t\}$
- For simplicity, possible realizations of $\{z_t\}$ are a finite number

Crucial assumption about the prob. distribution of z:

DEFINITION 6

A stochastic process $\{z_t\}$ is said to have the **Markov property** if for all $k \geq 1$ and all t ,

$$Prob(z_{t+1}|z_t, z_{t-1}, ..., z_{t-k)} = Prob(z_{t+1}|z_t)$$

MARKOV SHOCKS

Markov shocks

- We assume $\{z_t\}$ satisfies the Markov property
- Conditional probabilities of state z' tomorrow, given state z today is $\pi(z'|z)$.

Stochastic Bellman equation

STOCHASTIC BELLMAN EQUATION

- We can solve the sequential problem by analysing a slightly different Bellman equation:

$$V(x,z) = \max_{y \in \Gamma(x,z)} f(x,y,z) + \beta \sum_{z'} \pi\left(z'|z\right) V\left(y,z'\right) \qquad \forall (x,z) \tag{15}$$

STOCHASTIC BELLMAN EQUATION

- We can solve the sequential problem by analysing a slightly different Bellman equation:

$$V(x,z) = \max_{y \in \Gamma(x,z)} f(x,y,z) + \beta \sum_{z'} \pi\left(z'|z\right) V\left(y,z'\right) \qquad \forall (x,z) \tag{15}$$

- Results similar to the deterministic case can be proved

THREE TECHNIQUES

There are essentially three ways to solve a dynamic programming problem:

1. Guess and verify (undetermined coefficients) either the policy or the value function
2. Value function iteration
3. Policy function iteration, a.k.a. Howard's improvement algorithm

1. Guess and verify

a. Guess the value function

1. Guess a value function
2. Substitute the guess into the Bellman equation
3. Find the optimal policy by maximizing the RHS of Bellman equation
4. Put the optimal policy found above into the Bellman equation and solve for the undetermined coefficients

1. Guess and verify

b. Guess the policy function

1. Guess a policy function
2. Find the optimal policy by maximizing the RHS of Bellman equation
3. Use the guess in the optimal policy found above and solve for the undetermined coefficients

2. Value function iteration

1. Start from an arbitrary value function $V^{(0)}$
2. Get $V^{(1)} = TV^{(0)}$
3. Keep iterating on step 2 until you reach the fixed point V

3. Policy function iteration

1. Start with a guess for the policy function
2. Calculate the value associated with this policy function
3. Get a new policy function by solving the RHS of the Bellman equation
4. Calculate the value associated with this policy function
5. Iterate over 3-4 until you reach the fixed point

Value function iteration

VALUE FUNCTION ITERATION

- Very general, can *potentially* solve any model

VALUE FUNCTION ITERATION

- Very general, can *potentially* solve any model
- Pretty *slow*, becomes *unmanageable* as the state space increases

VALUE FUNCTION ITERATION

- Very general, can *potentially* solve any model
- Pretty *slow*, becomes *unmanageable* as the state space increases
- Can be *adapted to models with time inconsistency, optimal policy problems and New Keynesian models*

How to find the solution

Algorithm

1. Start from an arbitrary value function $V^{(0)}$
2. Get $V^{(1)} = TV^{(0)}$,
3. Keep iterating on step 2 until you reach the fixed point V

Contraction mapping theorem: convergence is pointwise

DISCRETE GRID METHODS

- Computers and continuous variables do not love each other

DISCRETE GRID METHODS

- Computers and continuous variables do not love each other
- We need to **discretize** the number of possible choices we have (**grid**)
- This is a first source of approximation (depends on number of points)

DISCRETE GRID METHODS

- Computers and continuous variables do not love each other
- We need to **discretize** the number of possible choices we have (**grid**)
- This is a first source of approximation (depends on number of points)

Define a grid $G \equiv \{x_1, ..., x_m\}$

VALUE FUNCTION ITERATION ALGORITHM

VALUE FUNCTION ITERATION ALGORITHM

1. We form an initial guess for the value function for each point on the grid

VALUE FUNCTION ITERATION ALGORITHM

1. We form an initial guess for the value function for each point on the grid
2. For any $n \geq 0$, compute $V^{(n+1)}(\cdot) = TV^{(n)}(\cdot)$ for any point in the grid

VALUE FUNCTION ITERATION ALGORITHM

1. We form an initial guess for the value function for each point on the grid
2. For any $n \geq 0$, compute $V^{(n+1)}(\cdot) = TV^{(n)}(\cdot)$ for any point in the grid
3. Stop if $\left\| V^{(n+1)} - V^{(n)} \right\| < \varepsilon$

VALUE FUNCTION ITERATION ALGORITHM

1. We form an initial guess for the value function for each point on the grid
2. For any $n \geq 0$, compute $V^{(n+1)}(\cdot) = TV^{(n)}(\cdot)$ for any point in the grid
3. Stop if $\left\| V^{(n+1)} - V^{(n)} \right\| < \varepsilon$

SOLVING STOCHASTIC GROWTH MODEL

Solving stochastic growth model

Same as deterministic, except for:

$$y_t = A_t F(k_t, n_t)$$

where A_t is a Markov process with transition probability matrix $\mathscr{P}$

BRINGING THE ALGORITHM TO MATLAB

Translating an algorithm in a Matlab code can be troubling

BRINGING THE ALGORITHM TO MATLAB

Translating an algorithm in a Matlab code can be troubling

- Assumptions on functional forms, parameter values, convergence criterion, etc.

Bringing the algorithm to Matlab

Translating an algorithm in a Matlab code can be troubling

- Assumptions on functional forms, parameter values, convergence criterion, etc.
- Matlab is not very good in loops!

BRINGING THE ALGORITHM TO MATLAB

Translating an algorithm in a Matlab code can be troubling

- Assumptions on functional forms, parameter values, convergence criterion, etc.
- Matlab is not very good in loops!
    - (Loop: a piece of code that performs a series of instructions repeatedly until some specified condition is satisfied)

BRINGING THE ALGORITHM TO MATLAB

Translating an algorithm in a Matlab code can be troubling

- Assumptions on functional forms, parameter values, convergence criterion, etc.
- Matlab is not very good in loops!
    - (Loop: a piece of code that performs a series of instructions repeatedly until some specified condition is satisfied)
    - Slow in Matlab, work with matrix algebra to avoid loops

MATLAB CLUB

MATLAB CLUB

- The First Rule of Matlab Club is, you do not use loops.

MATLAB CLUB

- The First Rule of Matlab Club is, you do not use loops.
- The Second Rule of Matlab Club is,
  YOU DO NOT USE LOOPS.

VECTORIZING THE PROBLEM

VECTORIZING THE PROBLEM

- Productivity shock: $[A_1, A_2]$
- n grid points for capital $[k_1, k_2, ..., k_n]$

VECTORIZING THE PROBLEM

- Productivity shock: $[A_1, A_2]$
- n grid points for capital $[k_1, k_2, ..., k_n]$
- Define two matrices U_j such that:

$$U_j(i,h) = u\left(A_j f(k_i) + (1-\delta)k_i - k_h\right), \quad i = 1, ..., n, \quad h = 1, ..., n$$

VECTORIZING THE PROBLEM (CONT.)

VECTORIZING THE PROBLEM (CONT.)

- Define two $n \times 1$ vectors $V_j, j = 1, 2$ such that

  $V_j\left(i\right) = V_j\left(k_i, A_j\right), i = 1, ..., n.$

VECTORIZING THE PROBLEM (CONT.)

- Define two $n \times 1$ vectors $V_j, j = 1, 2$ such that $V_j(i) = V_j(k_i, A_j), i = 1, \ldots, n.$
- Define an operator $T([V_1, V_2])$ that maps couples of vectors $[V_1, V_2]$ into a couple of vectors$[TV_1, TV_2]$:

$$TV_1 = \max\{U_1 + \beta \mathscr{P}_{11} \mathbf{1} V_1' + \beta \mathscr{P}_{12} \mathbf{1} V_2'\}$$

$$TV_2 = \max\{U_2 + \beta \mathscr{P}_{21} \mathbf{1} V_1' + \beta \mathscr{P}_{22} \mathbf{1} V_2'\}$$

COMPACT FORM

COMPACT FORM

We can write these equations in compact form:

$$\begin{bmatrix} TV_1 \\ TV_2 \end{bmatrix} = \max \left\{ \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} + \beta \left(\mathscr{P} \otimes \mathbf{1} \right) \begin{bmatrix} V_1' \\ V_2' \end{bmatrix} \right\} \tag{16}$$

We can solve by iterating on the operator T until convergence.

THE CODE

(BTW: now it's a good time to open Matlab...)

THE CODE

Three files:

THE CODE

Three files:

- `parameters.m`: contains parameters' values and the grid

THE CODE

Three files:

- `parameters.m`: contains parameters' values and the grid
- `vfi_AM.m`: main routine that implements the value function iteration

THE CODE

Three files:

- `parameters.m`: contains parameters' values and the grid
- `vfi_AM.m`: main routine that implements the value function iteration
- `figures.m`: draws graphs

THE CODE

Three files:

- `parameters.m`: contains parameters' values and the grid
- `vfi_AM.m`: main routine that implements the value function iteration
- `figures.m`: draws graphs

A do-file makes easy for you to run the code: `do_vfi_AM.m`

parameters.m

```
%%  set parameter values
alpha  = 0.40;                 % production parameter
beta   = 0.95;                 % subjective discount factor
prob   = [ .5 .5; .5 .5]; % prob(A(t+1)=Aj | A(t) = Ai)
delta  = .90;                  % 1 - depreciation rate
A_high = 1.5;                  % high value for technology
A_low  = 0.5;                  % low value for technology
convcrit = 1e-7;        % convergence criterion (epsilon)

%%   generate capital grid
mink =   0.01;     % minimum value of the capital grid
maxk =  25.01;    % maximum value of the capital grid
nk   = 1000;              % number of grid points
kgrid = linspace(mink,maxk,nk)';  % the grid (linearly spaced
    )
ink = kgrid(2) - kgrid(1);                % increments
```

vfi_AM.m (I)

```
%  create the utility function matrices such that, for
%  zero or negative  consumption, utility remains a
%  large negative number so that such values will never
%  be chosen as utility maximizing

cons1 = bsxfun(@minus, A_high*kgrid'.^alpha  + delta*kgrid' ,
    kgrid);
cons2 = bsxfun(@minus, A_low*kgrid'.^alpha  + delta*kgrid' ,
    kgrid);

cons1(cons1<=0) = NaN;
cons2(cons2<=0) = NaN;

util1 =  log(cons1);
util2 =  log(cons2);

util1(isnan(util1)) = -inf;
util2(isnan(util2)) = -inf;
```

vfi_AM.m (II)

```
%%  initialize some variables

v = zeros(nk,2); % initial guess for value function:
                 % set to zero for simplicity
decis = zeros(nk,2);% initial value for policy function
metric = 10; % initial value for the convergence metric
iter = 0;
tme = cputime;
[rs,cs] = size(util1);
```

vfi_AM.m (III): THE MAIN LOOP

```
while metric > convcrit;
    contv= beta*v*prob'; % continuation value

    [tv1,tdecis1]=max(bsxfun(@plus,util1,contv(:,1)) );
    [tv2,tdecis2]=max(bsxfun(@plus,util2,contv(:,2)) );

    tdecis=[tdecis1' tdecis2'];
    tv=[tv1' tv2'];

    metric=max(max(abs((tv-v)./tv)));
    v=   tv; % .15*tv+.85*v; %
    decis= tdecis;%
    iter = iter+1;
    metric_vector(iter) = metric;
    disp(sprintf('iter = %g ; metric = %e', iter,metric));
end;
disp('  ');
disp(sprintf('computation time = %f', cputime-tme));

% transform the decision index in capital choice
decis=(decis-1)*ink + mink;
```

do_vfi_AM.m

```
%% do file for vfi_AM.m
clear all

% load parameters and grid
parameters;

% run the code
vfi_AM;

% generate figures
figures;
```

PLAYING WITH THE CODE

1. Run the code `do_vfi_AM.m`. Familiarize with the output and the graphs. Now, after the line which loads parameter values, change the value for the discount factor to 0.995, by adding the following line:

```
beta = 0.995
```

This line changes the value set in the file `parameters.m` to a new value (this is a general way to do comparative statics by using a baseline set of parameters.) Save this file as `do_vfi_AM_beta.m` and run it. What do you notice in the convergence process? What happens to computational time? (Why?)

PLAYING WITH THE CODE (II)

2. We now want to see what happens if we change the number of gridpoints. Change the code so that the grid has 100 gridpoints. (**Hint**: notice that after you set `nk=100`, then you also have to modify the variables `kgrid` and `ink`, therefore you need to add those lines too!!!). Save the file as `do_vfi_AM_smallgrid.m` and run it. What can you notice? Now try with 2000 gridpoints, save the file as `do_vfi_AM_largegrid.m`, and run it. Do you see any change?

PLAYING WITH THE CODE (III)

3. We want to do a series of comparative statics exercises. In order to do that, we can modify the file `do_compstat`, which is a basic structure for this task. Open the file `do_compstat.m`. You should see the following lines (I have excluded the lines that plot the simulated results for brevity):

PLAYING WITH THE CODE (III)

```
%% Generate solution and simulation for case A
% load parameters and grid
parameters;
% solve the model via VFI
vfi_AM;
% store results in few new variables
v_A = v;
decis_A = decis;
controls_A = controls;
%% Generate solution and simulation for case B
% load parameters and grid
parameters;
% modify parameters here

% solve the model via VFI
vfi_AM;
% store results in few new variables
v_B = v;
decis_B = decis;
controls_B = controls;
```

Playing with the code (III)

This file compares a case A and a case B, where the case A is the benchmark (i.e. the solution with default parameters), and case B is the one with the updated parameter value. Notice the line that says:

```
% modify parameters here
```

We can just put the new value we want to consider there. Then by running the file we should get the graphs of case A and B and compare them. As a first try, set `delta = 0.8` (notice that this implies a larger depreciation rate for capital, given the definition of δ in the code). Save the file as `do_compstat_delta.m`. Run the do file, what changes with respect to

PLAYING WITH THE CODE (IV)

4. Now let's change the values for the shock realizations `A_high` and `A_low`. In particular, let's have a mean-preserving-spread transformation such that the the mean is the same given the i.i.d hypothesis for the transition matrix. Set `A_high=1.25` and `A_low=0.75`, save the file as `do_compstat_shock.m` and run the code. What can you notice?

PLAYING WITH THE CODE (V)

5. Let's now relax the assumption of i.i.d shocks. In order to do that, we need to change the transition matrix `prob` by inducing some persistence in the stochastic process. Modify the matrix in such a way that each realizations has a probability of repeating itself in the following period of 95%, i.e. the probability of a high (low) realization tomorrow given that the realization today is high (low) is 0.95. Save the file as `do_compstat_persistent.m` and run the code. What conclusions can you draw from the graphs?

Comparative static for irreversible investment model

Repeat the previous exercise for the model with irreversible investment. The relevant codes are `do_vfi_AM_irrinv.m` and `do_compstat_irrinv.m`.

Reversible vs. irreversible investment

Now let's compare the reversible investment model with the one with irreversible investment. In order to do that, we use the file `do_compare.m`. (Notice that in this case we will have to change parameters in two points of the code!)

1. Run the code as it is, and look at the graphs. What are the main differences between the reversible and irreversible investment models? Check in particular the simulated series of consumption and investment.

Exercise: Reversible vs irreversible investment (II)

2. Now let's how the depreciation rate is crucial. First change the depreciation rate of capital to `delta=0.99`. You can do this by inserting `delta=0.99` in the appropriate spaces (remember: you have two do it twice for this code!!!). Save the new file as `do_compare_delta.m` and run it. What do you observe? Can you explain it intuitively? Now set `delta=0.5` (in both the appropriate spaces!!!) and run the code again. What now?

EXERCISE: REVERSIBLE VS IRREVERSIBLE INVESTMENT (III)

3. Let's see how the model with irreversible investment reacts when the variance of the shocks is reduced. Set `A_high=1.25` and `A_low=0.75`, save the file as `do_compare_shock.m` and run the code. This must be surprising for you! (Is it?)

EXERCISE: REVERSIBLE VS IRREVERSIBLE INVESTMENT (IV)

4. Finally, what about persistence? Set the transition matrix `prob` such that each realization has a probability of repeating itself in the following period of 95%, i.e. the probability of a high (low) realization tomorrow given that the realization today is high (low) is 0.95. Save the file as `do_compare_persistent.m` and run the code.

Many realizations of A_t (difficult)

More than two realizations of the shocks: we need to modify the code in several points

1. we need a new variable that stores the number of realizations of the shocks (call it `num_realiz`) in the parameters' file
2. we need to provide a transition matrix `prob` which adapts to `num_realiz` (for the moment we stick to the i.i.d case for simplicity) in the parameters' file
3. we need a vector where we store the actual values for technology shock realizations in the parameters' file
4. we need to adapt the way in which we create the matrices U_j (hint: use the Kronecker product, use Matlab help for the `kron` command)
5. we need to adjust the way in which we compute the Bellman operator, in particular for the expectations' part
6. finally, we must adapt the simulations to the generic case with many possible realizations of the Markov chain

We can do the same for the model with irreversible investment. The two solution codes are `vfi_AM_general.m` and `vfi_AM_irrinv_general.m`.

POLICY FUNCTION ITERATION

Policy function iteration

- Value function iteration is slow, and for many economic problems computational speed is important

Policy function iteration

- Value function iteration is slow, and for many economic problems computational speed is important
- PFI is similar approach, but faster
- Also known as Howard's improvement algorithm

Policy function iteration

- Value function iteration is slow, and for many economic problems computational speed is important
- PFI is similar approach, but faster
- Also known as Howard's improvement algorithm
- Remember our problem:

$$V(K_{-1}) \equiv \max_{\{K_t\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t U(f(K_{t-1}) - K_t)$$

$$s.t. \quad 0 \leq K_t \leq f(K_{t-1}), \quad t = 0, 1, \ldots$$

THE ALGORITHM

THE ALGORITHM

There are 5 steps:

1. Start with a guess for the policy function $K' = g^{(0)}(K)$

THE ALGORITHM

There are 5 steps:

1. Start with a guess for the policy function $K' = g^{(0)}(K)$
2. Calculate the value associated with this policy function:

$$V^{(0)}(K) = \sum_{t=0}^{\infty} \beta^t U\left(f(K_{t-1}) - g^{(0)}(K_{t-1})\right)$$

The algorithm (cont.)

3 For any $n \geq 0$, get a new policy function $K' = g^{(n+1)}(K)$ by solving

$$\max_{K'} \left\{ U\left(f(K) - K'\right) + \beta V^{(n)}\left(K'\right) \right\}$$

THE ALGORITHM (CONT.)

3 For any $n \geq 0$, get a new policy function $K' = g^{(n+1)}(K)$ by solving

$$\max_{K'} \left\{ U\left(f(K) - K'\right) + \beta V^{(n)}\left(K'\right) \right\}$$

4 Calculate the value associated with this policy function:

$$V^{(n+1)}(x) = \sum_{t=0}^{\infty} \beta^t U\left(f\left(K_{t-1}\right) - g^{(n+1)}\left(K_{t-1}\right)\right)$$

The algorithm (cont.)

3 For any $n \geq 0$, get a new policy function $K' = g^{(n+1)}(K)$ by solving

$$\max_{K'} \left\{ U\left(f(K) - K'\right) + \beta V^{(n)}\left(K'\right) \right\}$$

4 Calculate the value associated with this policy function:

$$V^{(n+1)}(x) = \sum_{t=0}^{\infty} \beta^t U\left(f\left(K_{t-1}\right) - g^{(n+1)}\left(K_{t-1}\right)\right)$$

5 Iterate over 3-4. Stop if $\left\| V^{(n+1)} - V^{(n)} \right\| < \varepsilon$

THE STOCHASTIC GROWTH MODEL

- The system is in one of N predetermined positions x_i, $i = 1, 2, ..., N$.
- Matrices P where $P_{ij} = Prob\{x_{t+1} = x_j \mid x_t = x_i\}$ are our choice
- We can write the Bellman equation as:

$$v(x_i) = \max_{P \in \mathscr{M}} \left\{ u(x_i) + \beta \sum_{j=1}^{N} P_{ij} v(x_j) \right\}$$

- In a more compact form:

$$v = \max_{P \in \mathscr{M}} \{u + \beta P v\} \tag{17}$$

THE STOCHASTIC GROWTH MODEL (CONT.)

We can rewrite it as as

$$v = Tv$$

where T is the operator corresponding to the RHS of (17). Define another operator $B \equiv T - I$ such that

$$Bv = \max_{P \in \mathscr{M}} \{u + \beta P v\} - v$$

and therefore we can see the policy function iteration as solving $Bv = 0$.

The algorithm reformulated

THE ALGORITHM REFORMULATED

The algorithm becomes:

1. Given $P^{(n)}$, get $v^{(n)}$ from

$$\left(I-\beta P^{(n)}\right)v^{(n)}=u^{(n)} \tag{18}$$

THE ALGORITHM REFORMULATED

The algorithm becomes:

1. Given $P^{(n)}$, get $v^{(n)}$ from

$$\left(I - \beta P^{(n)}\right) v^{(n)} = u^{(n)} \tag{18}$$

2. Find $P^{(n+1)}$ such that

$$u^{(n+1)} + \left(\beta P^{(n+1)} - I\right) v^{(n)} = Bv^{(n)} \tag{19}$$

The algorithm reformulated

The algorithm becomes:

1. Given $P^{(n)}$, get $v^{(n)}$ from

$$\left(I-\beta P^{(n)}\right)v^{(n)}=u^{(n)} \tag{18}$$

2. Find $P^{(n+1)}$ such that

$$u^{(n+1)}+\left(\beta P^{(n+1)}-I\right)v^{(n)}=Bv^{(n)} \tag{19}$$

THE ALGORITHM REFORMULATED (CONT.)

First step is a linear algebra problem:

$$v^{(n)} = \left(I - \beta P^{(n)}\right)^{-1} u^{(n)}$$

PFI AS NEWTON'S METHOD

- You can interpret the policy iteration algorithm as a form of the Newton's method to find the zeroes of $Bv = 0$.

PFI AS NEWTON'S METHOD

- You can interpret the policy iteration algorithm as a form of the Newton's method to find the zeroes of $Bv = 0$.
- What is Newton's method?

PFI AS NEWTON'S METHOD

- You can interpret the policy iteration algorithm as a form of the Newton's method to find the zeroes of $Bv = 0$.
- What is Newton's method?
- If we want to solve:

$$G(z) = 0$$

we can do it by iterating on:

$$z_{n+1} = z_n - G'(z_n)^{-1} G(z_n)$$

PFI AS NEWTON'S METHOD (CONT.)

- By using (18) rewrite equation (19) as:

$$\left(I-\beta P^{(n+1)}\right)v^{(n+1)}+\left(\beta P^{(n+1)}-I\right)v^{(n)}=Bv^{(n)}$$

PFI as Newton's method (cont.)

- By using (18) rewrite equation (19) as:

$$\left(I-\beta P^{(n+1)}\right)v^{(n+1)}+\left(\beta P^{(n+1)}-I\right)v^{(n)}=Bv^{(n)}$$

- or in other terms

$$v^{(n+1)}=v^{(n)}+\left(I-\beta P^{(n+1)}\right)^{-1}Bv^{(n)} \tag{20}$$

PFI as Newton's method (cont.)

- By using (18) rewrite equation (19) as:

$$\left(I-\beta P^{(n+1)}\right)v^{(n+1)}+\left(\beta P^{(n+1)}-I\right)v^{(n)}=Bv^{(n)}$$

- or in other terms

$$v^{(n+1)}=v^{(n)}+\left(I-\beta P^{(n+1)}\right)^{-1}Bv^{(n)} \tag{20}$$

- $\left(I-\beta P^{(n+1)}\right)$ is the gradient of $Bv^{(n)}$ (see equation (18))

PFI AS NEWTON'S METHOD (CONT.)

- By using (18) rewrite equation (19) as:

$$\left(I-\beta P^{(n+1)}\right)v^{(n+1)}+\left(\beta P^{(n+1)}-I\right)v^{(n)}=Bv^{(n)}$$

- or in other terms

$$v^{(n+1)}=v^{(n)}+\left(I-\beta P^{(n+1)}\right)^{-1}Bv^{(n)} \tag{20}$$

- $\left(I-\beta P^{(n+1)}\right)$ is the gradient of $Bv^{(n)}$ (see equation (18))
- $\Rightarrow$ PFI = Newton's method to solve $Bv=0$.

NUMERICAL IMPLEMENTATION

NUMERICAL IMPLEMENTATION

- Define the two $n \times n$ matrices:

$$J_h(k_i, k_j) = \left\{ \begin{array}{lll} 1 & \textit{if} & g(k_i, A_h) = k_j \\ 0 & o/w & \end{array} \right\}$$

NUMERICAL IMPLEMENTATION

- Define the two $n \times n$ matrices:

$$J_h(k_i, k_j) = \left\{ \begin{array}{lll} 1 & \mathit{if} & g(k_i, A_h) = k_j \\ 0 & o/w & \end{array} \right\}$$

- Given $k' = g(k, A)$, define two vectors U_h such that:

$$U_h(k_i) = u(A_h f(k_i) + (1-\delta) k_i - g(k_i, A_h))$$

Numerical implementation (cont.)

Numerical implementation (cont.)

- Assume the policy function is used forever
- We can associate the two vectors $V_h(k_i)$ as the values associated with starting from state (k_i, A_h):

$$\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} + \beta \begin{bmatrix} \mathscr{P}_{11} J_1 & \mathscr{P}_{12} J_1 \\ \mathscr{P}_{21} J_2 & \mathscr{P}_{22} J_2 \end{bmatrix} \begin{bmatrix} V_1 \\ V_2 \end{bmatrix}$$

SOLVING BY LINEAR ALGEBRA

We can therefore solve for V_h by means of elementary linear algebra and have:

$$\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \left[I - \beta \begin{pmatrix} \mathscr{P}_{11}J_1 & \mathscr{P}_{12}J_1 \\ \mathscr{P}_{21}J_2 & \mathscr{P}_{22}J_2 \end{pmatrix} \right]^{-1} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix} \tag{21}$$

Algorithm reformulated in linear algebra

Algorithm reformulated in linear algebra

1. Given an initial feasible policy function, calculate U_h and find V_h with

$$\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \left[I - \beta \begin{pmatrix} \mathscr{P}_{11} J_1 & \mathscr{P}_{12} J_1 \\ \mathscr{P}_{21} J_2 & \mathscr{P}_{22} J_2 \end{pmatrix} \right]^{-1} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix}$$

2. Do one iteration on the Bellman equation, by using value functions found in step 1, and find a new policy function

ALGORITHM REFORMULATED IN LINEAR ALGEBRA

1. Given an initial feasible policy function, calculate U_h and find V_h with

$$\begin{bmatrix} V_1 \\ V_2 \end{bmatrix} = \left[I - \beta \begin{pmatrix} \mathscr{P}_{11} J_1 & \mathscr{P}_{12} J_1 \\ \mathscr{P}_{21} J_2 & \mathscr{P}_{22} J_2 \end{pmatrix} \right]^{-1} \begin{bmatrix} U_1 \\ U_2 \end{bmatrix}$$

2. Do one iteration on the Bellman equation, by using value functions found in step 1, and find a new policy function

3. Iterate until convergence

pfi_AM.m, part 1

```
while metric > convcrit;
    contv= beta*v*prob'; % continuation value
    [tv1,tdecis1]=max(bsxfun(@plus,util1,contv(:,1)) );
    [tv2,tdecis2]=max(bsxfun(@plus,util2,contv(:,2)) );

    tdecis=[tdecis1' tdecis2'];
```

pfi_AM.m, PART 2

```
% Build return vectors
r1 = zeros(cs,1);
r2 = zeros(cs,1);
for i=1:cs
    r1(i) = util1(tdecis1(i),i);
    r2(i) = util2(tdecis2(i),i);
end
% create matrices Js (see lecture notes)
g2=sparse(cs,cs);
g1=sparse(cs,cs);
for i=1:cs
    g1(i,tdecis1(i))=1;
    g2(i,tdecis2(i))=1;
end
% This is the marix P (see lecture notes)
trans=[ prob(1,1)*g1 prob(1,2)*g1; prob(2,1)*g2 prob(2,2)
   *g2];
% Linear algebra step to get the value function
   associated with P
tv(:) = ((speye(2*cs) - beta.*trans))\[ r1; r2 ];
```

Exercise: Irreversible investment vs. adjustment costs II (the revenge)

Modify the PFI code to analyse the model with irreversible investment and the model with investment adjustment costs seen in the section about value function iteration. (Hint: are the modifications done for the VFI code enough?)


KEEP
CALM
AND
CODE
ON