Numerical Methods Bootcamp

Lecture 3
Solving nonlinear rational expectations models

Pontus Rendahl
rpk22@cam.ac.uk

2018

Introduction

- In yesterday's lecture I showed you how a more "advanced" method gave more accurate results using $N = 5$ grid points than the discretised method using $N = 1,000$ grid points.
- I also told you that you needed to know about functional approximations and nonlinear equations solvers in order to know how to do that yourself.
- So now that you know, how do you do it?

Introduction

$$V(k) = \max_{k'}\{u(f(k) + (1-\delta) - k') + \beta V(k')\}, \quad \text{for } k \in \mathcal{K}.$$

- Notice that k' is not restricted to be in $\mathcal{K}$.
- One way to solve:
  - Guess for $\hat{V}_0(k)$ for all $k \in \mathcal{K}$.
  - Use some approximation method to evaluate $V_0(k')$ for $k \notin \mathcal{K}$
  - Use some optimisation routine (e.g. `fminunc` in matlab) to solve the maximisation problem.
  - Rinse and repeat.

Introduction

$$V(k) = \max_{k'} \{u(f(k) + (1 - \delta) - k') + \beta V(k')\}, \quad \text{for } k \in \mathcal{K}.$$

- But for our problem, and for many other problems, we know that $V(k)$ ought to be
  1. Concave
  2. Differentiable
- So first order conditions are necessary and sufficient.
- Optimality satisfies

$$u'(f(k) + (1 - \delta)k - k') = \beta V'(k'), \quad \text{for } k \in \mathcal{K}.$$

Introduction

- In addition, the envelope theorem tells us that

$$V'(k) = (1 + f'(k) - \delta)u'(c)$$

- So what I did was starting with a guess $k' = g_0(k)$ for k $\in \mathcal{K}$.
- Found $V_0'(k)$ using the envelope theorem

$$V_0'(k) = (1 + f'(k) - \delta)u'(f(k) + (1 - \delta) - g_0(k)), \quad \text{for } k \in \mathcal{K}.$$

Introduction

- Used a linear approximation to create the function $\hat{V}_0'(k)$ $\forall k$
- Found k' as the solution to

$$u'(f(k)+(1-\delta)k-k')=\beta\hat{V}_0'(k'),\quad \text{for } k\in\mathcal{K}.$$

- And updated to $k'=g_1(k)$ for k $\in\mathcal{K}$.
- Then I repeated this until

$$||u'(f(k)+(1-\delta)k-g_n(k))-\beta\hat{V}_n'(g_n(k))|\ <\varepsilon,\quad \text{for } k\in\mathcal{K}.$$

Introduction

- Notice that I never updated or computed the value function itself
    - But only its slope
- And slope information is all that I need to solve for the solution to the nonlinear equation
- And the solution to the nonlinear equation is all I need to update the slope of the value function!

Introduction

- If I instead would have approximated the value function, I would have
  1. Approximated level information
  2. Use the derivative of this level information (which could be wildly inaccurate) to find the policy function
  3. Used the policy function to update level information.
- That's an unnecessary and inaccurate procedure (for differentiable problems)

Introduction

- Yet the contraction property still kicks in, and I will find $k' = g(k)$
- And if I would need the value function, I could simply iterate on

$$V_{n+1}(k) = u(f(k) + (1-\delta)k - g(k)) + \beta V_n(g(k)), \quad \text{for } k \in \mathcal{K}$$

  with $V_0(k) = 0$.
- Until

$$||V_{n+1} - V_n|| < \varepsilon$$

Introduction

Why does the contraction property kick in?

- We know that the sequence $\{V_n\}$ defined as

$$V_{n+1}(k) = \max_{k'}\{u(f(k) + (1-\delta) - k') + \beta V_n(k')\}$$

converges to V.

- We also know that each associated policy function $g_{n+1}(k)$ such that

$$V_{n+1}(k) = u(f(k) + (1-\delta) - g(k)) + \beta V_n(g(k)),$$

converges to g.

Introduction

Why does the contraction property kick in?

- We also know that

$$u'(f(k) + (1 - \delta) - k') - \beta V_n'(k') = 0,$$

  is necessary and sufficient for an optimum
- Turns out that the envelope theorem also says that

$$V_n'(k) = (1 + f'(k) - \delta)u'(f(k) + (1 - \delta)k - g_n(k))$$

Introduction

Why does the contraction property kick in?

- So the sequence $\{g_n\}$ defined as

$$u'(f(k) + (1 - \delta) - g_{n+1}(k)) - \beta(1 + f'(g_{n+1}(k)) - \delta)$$
$$\times u'(f(g_{n+1}(k)) + (1 - \delta)g_{n+1}(k) - g_n(g_{n+1}(k))) = 0,$$

  converges to $g(k)$.
- So we can also just iterate on the Euler equation

Introduction

Why does the contraction property kick in?

- That is, suppose you have $\hat{g}_n(k)$.
- For each $k \in \mathcal{K}$ find k' as

$$u'(f(k) + (1-\delta) - k') - \beta(1 + f'(k') - \delta) \\ \times u'(f(k') + (1-\delta)k' - \hat{g}_n(k')) = 0,$$

and use the k' values to construct $\hat{g}_{n+1}(k)$ using some functional approximation.

Introduction

Why does the contraction property kick in?

- This is basically what I did.
- But I started with a guess $\hat{V}'_n(k)$.
- For each $k \in \mathcal{K}$ I found k' as

$$u'(f(k) + (1-\delta) - k') - \beta \hat{V}'_n(k') = 0,$$

  and use the k' values to construct $\hat{V}'_{n+1}(k)$ using

$$V'_{n+1}(k) = (1 + f'(k') - \delta)u'(f(k') + (1-\delta)k' - \hat{g}_n(k')) = 0$$

- Same same but different.

Introduction

- One last thing
- How do I solve

$$u'(f(k) + (1-\delta)k - k') = \beta \hat{V}_0'(k'), \quad \text{for } k \in \mathcal{K}?$$

- Newton's method!
- Set the problem up as

$$-u'(f(k) + (1-\delta)k - k') + \beta \hat{V}_n'(k') = 0, \quad \text{for } k \in \mathcal{K}$$

- We can call this $F_n(k, k')$.

Introduction

- And notice that

$$\frac{\partial F(k,k')}{\partial k'} = u''(f(k) + (1-\delta)k - k') + \beta \frac{\partial \hat{V}_n'(k')}{\partial k'}$$

- We have an initial guess for $F(k,k')$, namely $k' = g_n^0(k)$.
- Using this initial guess $k' = g_n^0(k)$, $k \in \mathcal{K}$, the last derivative can be approximated as

$$\frac{\partial \hat{V}_n'(k')}{\partial k'} \approx \frac{\texttt{gradient}(\hat{V}_n'(g_n(k)))}{\texttt{gradient}(g_n(k))}$$

Introduction

- So define

$$dE(k) = u''(f(k) + (1-\delta)k - g_n^0(k)) + \beta \frac{\texttt{gradient}(\hat{V}_n'(g_n^0(k)))}{\texttt{gradient}(g_n^0(k))}, \quad \text{for } k \in \mathcal{K}$$

- And

$$E(k) = -u'(f(k) + (1-\delta)k - k') + \beta \hat{V}_n'(k') = 0, \quad \text{for } k \in \mathcal{K}$$

- And update

$$g_n^1(k) = g_n^0(k) - \frac{E(k)}{dE(k)}$$

- Until (weak) convergence.

Introduction

- That is until

$$\|g_n^i - g_n^{i-1}\| < 1e(-2)$$

  or so (or perhaps just one iteration!)
- Doing this step at a tighter tolerance level is often unnecessary and takes up (a lot) of computer time.
- Notice that the real constraint is rather on

$$\|g_n - g_{n-1}\| < \varepsilon$$

- Which should be tight, $\varepsilon = 1e(-6)$ is a good rule of thumb.

Solving functional equations

- In general, our functional equations boil down to something like

$$E_{z'}[f(x, x', x'', z, z') = 0, \quad \text{all } x \in \mathcal{X} \text{ and } z \in \mathcal{Z}$$

- And we would like to find a function $x' = g(x, z)$ such that

$$E_{z'}[f(x, g(x, z), g(g(x, z), z'), z, z') = 0$$

Solving functional equations

- There are several approaches to this
- Recall that what we will find is only an approximation $\hat{g}(\cdot) \approx g(\cdot)$.
- This approximation is, for a vector of $\mathcal{X}$ and $\mathcal{Z}$, given by a vector of x'.
- So in principle we can just use a nonlinear equation solver to find x' for all $x \in \mathcal{X}$ and all $z \in \mathcal{Z}$ such that

$$E_{z'}[f(x, x', \hat{g}(x', z'), z, z') = 0, \quad \text{all } x \in \mathcal{X} \text{ and } z \in \mathcal{Z}$$

- Where $\hat{g}(\cdot)$ is consistent with x'.

Solving functional equations

- This is a “direct approach”, and is normally very fast
- But it’s only fast when it works.
- And my experience is that it doesn’t work too often.
- Big systems of nonlinear equations are difficult to solve.
- Not very robust.
- Use only if speed is crucial.

Solving functional equations

Fixed point iteration

- Suppose we have a candidate policy function, $g_n(k, z)$
- Sometimes we can rewrite the problem

$$E_{z'}[f(x, x', x'', z, z')] = 0, \quad \text{all } x \in \mathcal{X} \text{ and } z \in \mathcal{Z}$$

as

$$E_{z'}[F(x, x', x'', z, z')] - x' = 0, \quad \text{all } x \in \mathcal{X} \text{ and } z \in \mathcal{Z}$$

- Insert $g_n(x, z)$ to get

$$E_{z'}[F(x, x', g_n(x, z), z, z')] - x' = 0, \quad \text{all } x \in \mathcal{X} \text{ and } z \in \mathcal{Z}$$

Solving functional equations

Fixed point iteration

- Problem again

$$E_{z'}[F(x, x', g_n(x', z'), z, z')] - x' = 0, \quad \text{all } x \in \mathcal{X} \text{ and } z \in \mathcal{Z}$$

- Fixed point iteration would then suggest that we find x' as

$$x' = E_{z'}[F(x, g_n(x, z), g_n(g_n(x, z), z'), z, z')],$$
$$\text{all } x \in \mathcal{X} \text{ and } z \in \mathcal{Z}$$

- Until

$$||g_n(x, z) - E_{z'}[F(x, g_n(x, z), g_n(g_n(x, z), z'), z, z')]|| < \varepsilon$$

Solving functional equations

Fixed point iteration

- Fixed point iteration is normally very fast
  - No derivatives to compute, just evaluate a function
- But it can have some serious convergence problems
  - A good initial guess is *very* helpful.
- It can be useful to first find

$$\hat{x}' = E_{z'}[F(x, g_n(x,z), g_n(g_n(x,z), z'), z, z')], \quad \text{all } x \in \mathcal{X} \text{ and } z \in \mathcal{Z}$$

- And find x' as

$$x' = \rho \hat{x}' + (1-\rho) g_n(x,z), \text{ all } x \in \mathcal{X} \text{ and } z \in \mathcal{Z}$$

for a low value of ρ.

Solving functional equations

Time iteration

- Suppose we have a candidate policy function, $g_n(k,z)$
- Our problem is

$$E_{z'}[f(x,x',x'',z,z') = 0, \quad \text{all } x \in \mathcal{X} \text{ and } z \in \mathcal{Z}$$

- Insert $g_n(k,z)$

$$E_{z'}[f(x,x',g_n(x',z'),z,z') = 0, \quad \text{all } x \in \mathcal{X} \text{ and } z \in \mathcal{Z}$$

- Use a nonlinear equation solver to find x', and update to $g_{n+1}(x,z)$.
- Iterate until

$$||E_{z'}[f(x,g_n(x,z),g_n(g_n(x,z),z'),z,z')|| < \varepsilon$$

Solving functional equations

Time iteration

- This is actually what I did in the beginning of this lecture
- It's my favourite method as it is a contraction mapping!
  - So convergence is at least theoretically guaranteed
- But it's slower than fixed point iteration.

Solving functional equations

Getting the best of both worlds

- Use time iteration until

$$||E_{z'}[f(x, g_n(x,z), g_n(g_n(x,z), z'), z, z')|| < \varepsilon$$

  for a large value of ε.
- Then switch to fixed point iteration.
- Possibly use the solution to a linearized version of your problem as an initial guess (if such a solution exist).

Solving functional equations

Improving on time iteration

- In the univariate case there exist an improvement technique for time iteration known as *the method of endogenous gridpoints*.
- It works like a charm
  - As fast as fixed point iteration
  - With sustained contraction property
- Only works for the univariate case, but still.

Solving functional equations

Method of Endogenous Gridpoints

- The idea is the following:
- It is impossible to get a closed form solution for x' from

$$E_{z'}[f(x, x', g_n(x', z'), z, z') = 0, \quad \text{all } x \in \mathcal{X} \text{ and } z \in \mathcal{Z}$$

- So for time iteration we used a nonlinear equation solver to find this.
- However, given a value of x', it is usually trivial to find a close for solution for x!

Solving functional equations

Method of Endogenous Gridpoints

- So create a grid for x' instead, $\mathcal{X}'$.
- And solve for x from

$$E_{z'}[f(x, x', g_n(x', z'), z, z') = 0, \quad \text{all } x' \in \mathcal{X}' \text{ and } z \in \mathcal{Z}$$

- Then use your vector of x' together with your solutions x to update $g_{n+1}(x, z)$!

Solving functional equations

Method of Endogenous Gridpoints

- For instance, the stochastic growth model is given by

$$
\begin{aligned}
u'(m-k') &= \beta E_{z'}[(1+z'f(k')-\delta)u'(m'-g_n(m',z'))] \\
\textbf{with} \quad m &= zf(k)+(1-\delta)k \\
m' &= z'f(k')+(1-\delta)k'
\end{aligned}
$$

- Notice that m is as a legitimate state variable as k.

Solving functional equations

Method of Endogenous Gridpoints

$$u'(m-k') = \beta E_{z'}[(1+z'f(k')-\delta)u'(m'-g_n(m',z'))]$$

- So given a grid for k' and z, calculate m', and find m as

  $$m = u'^{-1}\{\beta E_{z'}[(1+z'f(k')-\delta)u'(m'-g_n(m',z'))]\} + k'$$

- Then update $g_{n+1}(m,z)$ until convergence.

Occasionally binding constraints

- Many interesting economic problem involves various sorts of inequality constraints that "occasionally bind"
  - Borrowing constraints
  - Irreversibility constraints
  - Collateral constraints
  - Implementability constraints
- Linearisation techniques are (almost) useless to solve models with these types of constraints
- It is generally perceived as hard to solve such problems
- I disagree

Occasionally binding constraints

Let's take two examples

1. A borrowing constraint
2. Irreversible investment

A borrowing constraint

Consider the following optimisation problem

$$V(b_0, s_0) \max_{\{c_t(s^t), b_{t+1}(s^t)\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \sum_{s^t \in \mathcal{S}^{t+1}} \beta^t u(c_t(s^t)) P(s^t, s_0)$$

subject to $\quad c_t(s^t) + b_{t+1}(s^t) = s_t w + (1 - s_t)\mu w + (1 + r) b_t(s^t),$

$$b_{t+1}(s^t) \geq \underline{b}$$

$$\forall t, \forall s^t \in \mathcal{S}^{t+1} \quad b_0, s_0 \text{ are given}$$

Bellman equation

$$v(b, s) = \max_{b' \geq \underline{b}} \{u(b(1 + r) + w(s) - b') + \beta \sum_{s'=0}^{1} v(b', s') p(s', s)\}$$

A borrowing constraint

$$V(b,s) = \max_{b' \geq \underline{b}} \{u(b(1+r) + w(s) - b') + \beta \sum_{s'=0}^{1} V(b',s')p(s',s)\}$$

- First order condition

$$u'(b(1+r) + w(s) - b') - \mu(b,s) = \beta \sum_{s'=0}^{1} V_{b'}(b',s')p(s',s)$$

where $\mu(b,s)$ is the Lagrange multiplier on the borrowing constraint.

A borrowing constraint

- Using the envelope theorem

$$u'(b(1+r)+w(s)-b')-\mu(b,s) =$$

$$\beta(1+r)\sum_{s'=0}^{1} u'(b'(1+r)+w(s')-g(b',s'))p(s',s)$$

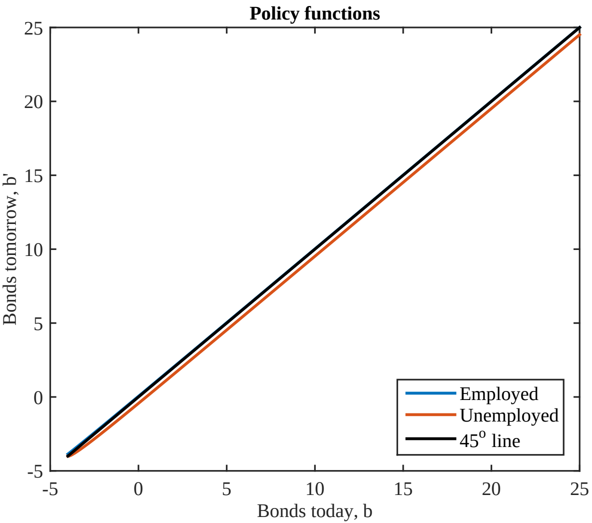
- Suppose we have a guess $g_n(b,s)$. Then find $\tilde{b}'$ as

$$u'(b(1+r)+w(s)-\tilde{b}') =$$

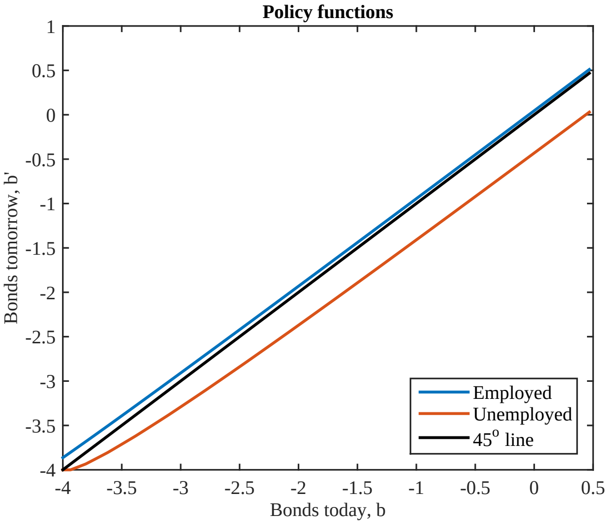
$$\beta(1+r)\sum_{s'=0}^{1} u'(\tilde{b}'(1+r)+w(s')-g_n(\tilde{b}',s'))p(s',s)$$

- Construct $\tilde{g}_{n+1}(b,s)$ using $\tilde{b}'$.
- Update $g_{n+1}(b,s) = \max\{\tilde{g}_{n+1}(b,s), \underline{b}\}$

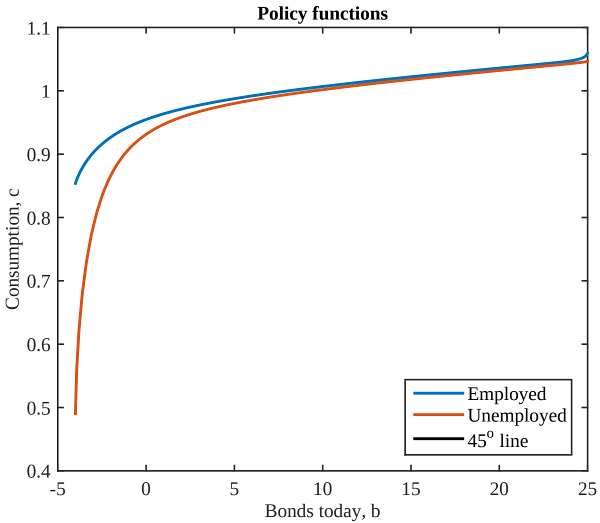
Income fluctuation problem



Income fluctuation problem



Income fluctuation problem



Long run distribution

- When we analysed the problem solved by discretised value function iteration we could represent our policy function as a transition matrix
- From this we could calculate the transition matrix for the entire economy
- Find the long run distribution of capital
- And therefore calculate things like the average capital stock.

Long run distribution

- In this problem, however, the policy function is of the type $g(b, s)$.
- This cannot be represented as a transition matrix.
- So how can we find the long run distribution of bonds (and employment status) in this case?

Long run distribution

- In general, suppose that $\psi_0(b, s)$ is a probability density function in period zero
- Then

$$\psi_1(b', s') = \sum_{s \in \mathcal{S}} \sum_{\{b : b' = g(b,s)\}} \psi_0(b, s) p(s', s)$$

- And in general

$$\psi_{t+1}(b', s') = \sum_{s \in \mathcal{S}} \sum_{\{b : b' = g(b,s)\}} \psi_t(b, s) p(s', s)$$

Complications

- A stationary cross-sectional distribution, ψ, is such that

$$\psi(b', s') = \sum_{s \in \mathcal{S}} \sum_{\{b : b' = g(b,s)\}} \psi(b, s) p(s', s)$$

- This is very tricky to compute. The best practice I am aware of is to convert $g(a, \theta)$ into a transition matrix!

Solutions

Consider the following policy function

$$\begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix} \rightarrow \begin{pmatrix} 1.8 \\ 2.4 \\ 3 \\ 3.6 \\ 4.2 \end{pmatrix}$$

Solutions

Nearest neighbor interpolation

$$\begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix} \to \begin{pmatrix} 2 \\ 2 \\ 3 \\ 4 \\ 4 \end{pmatrix}$$

Solutions

Can be written as transition matrix

$$\begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Solutions

But we can be a bit smarter. Policy function

$$\begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{pmatrix} \to \begin{pmatrix} 1.8 \\ 2.4 \\ 3 \\ 3.6 \\ 4.2 \end{pmatrix}$$

Solutions

Can be written as

$$\begin{pmatrix} 0.2 & 0.8 & 0 & 0 & 0 \\ 0 & 0.6 & 0.4 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0.4 & 0.6 & 0 \\ 0 & 0 & 0 & 0.8 & 0.2 \end{pmatrix}$$

Solutions

But in the income fluctuation problem (and many others) we normally have two (or many) policy functions

$$\begin{pmatrix}1\\2\\3\\4\\5\end{pmatrix} \underset{\text{if good state}}{\rightarrow} \begin{pmatrix}2.2\\2.8\\3.4\\4\\4.6\end{pmatrix}, \quad \text{and} \quad \begin{pmatrix}1\\2\\3\\4\\5\end{pmatrix} \underset{\text{if bad state}}{\rightarrow} \begin{pmatrix}1.4\\2\\2.6\\3.2\\3.8\end{pmatrix}$$

With some transition matrix for good and bad states

$$T = \begin{pmatrix}0.8 & 0.2\\0.3 & 0.7\end{pmatrix}$$

Solutions

Nearest neighbor interpolation

$$\begin{pmatrix}1\\2\\3\\4\\5\end{pmatrix} \underset{\text{if good state}}{\longrightarrow} \begin{pmatrix}2\\3\\3\\4\\5\end{pmatrix}, \quad \text{and} \quad \begin{pmatrix}1\\2\\3\\4\\5\end{pmatrix} \underset{\text{if bad state}}{\longrightarrow} \begin{pmatrix}1\\2\\3\\3\\4\end{pmatrix}$$

With some transition matrix for good and bad states

$$T = \begin{pmatrix}0.8 & 0.2\\0.3 & 0.7\end{pmatrix}$$

Solutions

Two transition matrices

$$M_g = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}, \quad \text{and} \quad M_b = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

With some transition matrix for good and bad states

$$T = \begin{pmatrix} 0.8 & 0.2 \\ 0.3 & 0.7 \end{pmatrix}$$

Solutions

Full transition matrix is given by

$$\begin{pmatrix} T(1,1)\cdot M_g & T(1,2)\cdot M_g \\ T(2,1)\cdot M_b & T(2,2)\cdot M_b \end{pmatrix}$$

Solutions

Full transition matrix is given by

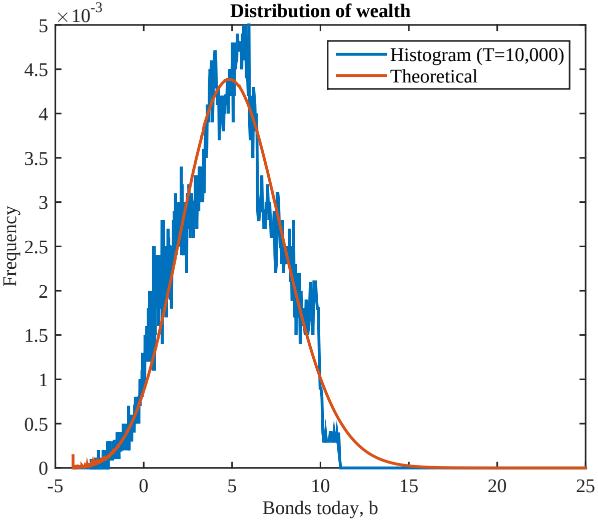
$$\begin{pmatrix} 0 & 0.8 & 0 & 0 & 0 & 0 & 0.2 & 0 & 0 & 0 \\ 0 & 0 & 0.8 & 0 & 0 & 0 & 0 & 0.2 & 0 & 0 \\ 0 & 0 & 0.8 & 0 & 0 & 0 & 0 & 0.2 & 0 & 0 \\ 0 & 0 & 0 & 0.8 & 0 & 0 & 0 & 0 & 0.2 & 0 \\ 0 & 0 & 0 & 0 & 0.8 & 0 & 0 & 0 & 0 & 0.2 \\ 0.3 & 0 & 0 & 0 & 0 & 0.7 & 0 & 0 & 0 & 0 \\ 0 & 0.3 & 0 & 0 & 0 & 0 & 0.7 & 0 & 0 & 0 \\ 0 & 0 & 0.3 & 0 & 0 & 0 & 0 & 0.7 & 0 & 0 \\ 0 & 0 & 0.3 & 0 & 0 & 0 & 0 & 0.7 & 0 & 0 \\ 0 & 0 & 0 & 0.3 & 0 & 0 & 0 & 0 & 0.7 & 0 \end{pmatrix}$$

Solutions

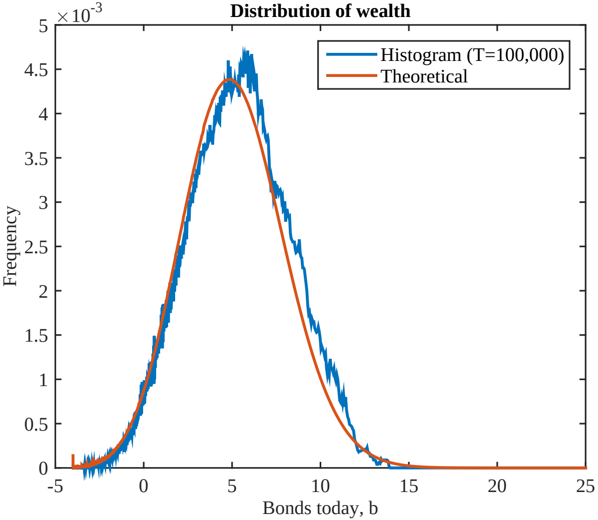
Doing the same thing with the smarter method gives

$$\begin{pmatrix}
0 & 0.64 & 0.16 & 0 & 0 & 0 & 0.16 & 0.04 & 0 & 0 \\
0 & 0.16 & 0.64 & 0 & 0 & 0 & 0.04 & 0.16 & 0 & 0 \\
0 & 0 & 0.48 & 0.32 & 0 & 0 & 0 & 0.12 & 0.08 & 0 \\
0 & 0 & 0 & 0.8 & 0 & 0 & 0 & 0 & 0.2 & 0 \\
0 & 0 & 0 & 0.32 & 0.48 & 0 & 0 & 0 & 0.08 & 0.12 \\
0.18 & 0.12 & 0 & 0 & 0 & 0.42 & 0.28 & 0 & 0 & 0 \\
0 & 0.3 & 0 & 0 & 0 & 0 & 0.7 & 0 & 0 & 0 \\
0 & 0.12 & 0.18 & 0 & 0 & 0 & 0.28 & 0.42 & 0 & 0 \\
0 & 0 & 0.24 & 0.06 & 0 & 0 & 0 & 0.56 & 0.14 & 0 \\
0 & 0 & 0.06 & 0.24 & 0 & 0 & 0 & 0.14 & 0.56 & 0
\end{pmatrix}$$

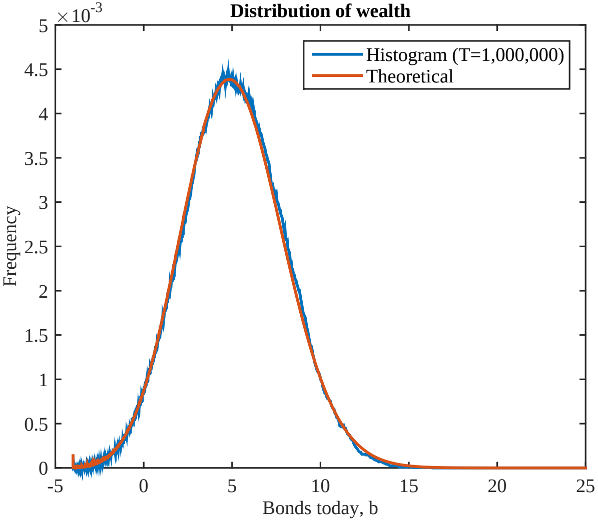
Income fluctuation problem



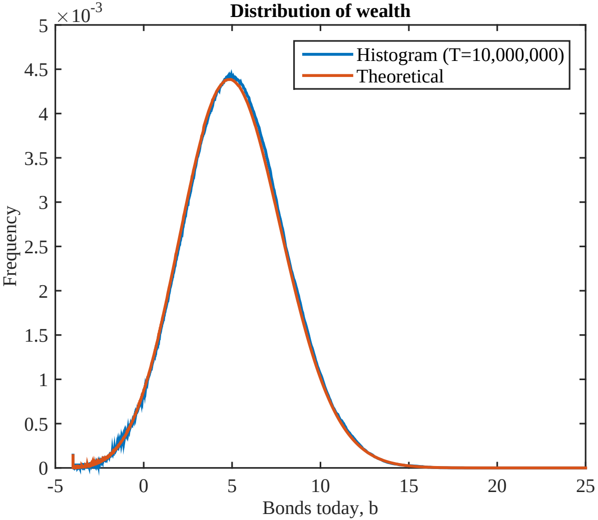
Income fluctuation problem



Income fluctuation problem



Income fluctuation problem



Irreversible investment

Consider the following optimisation problem

$$V(k_0, z_0) \max_{\{c_t(z^t), k_{t+1}(z^t)\}_{t=0}^{\infty}} \sum_{t=0}^{\infty} \sum_{z^t \in \mathcal{Z}^{t+1}} \beta^t u(c_t(z^t)) P(z^t, z_0)$$

$$\text{subject to} \quad c_t(z^t) + k_{t+1}(z^t) = z_t f(k_t(z_{t-1})) + (1-\delta) k_t(z^{t-1}),$$
$$k_{t+1}(z^t) \geq (1-\delta) k_t(z^{t-1})$$
$$\forall t, \forall z^t \in \mathcal{Z}^{t+1} \quad k_0, z_0 \text{ are given}$$

Bellman equation

$$v(k, z) = \max_{k' \geq (1-\delta)k} \{u(zf(k) + (1-\delta)k - k') + \beta \sum_{z' \in \mathcal{Z}} v(k', z') p(z', z)\}$$

Irreversible investment

$$v(k,z) = \max_{k' \geq (1-\delta)k} \{u(zf(k) + (1-\delta)k - k') + \beta \sum_{z' \in \mathcal{Z}} v(k',z')p(z',z)\}$$

- First order condition

$$u'(zf(k) + (1-\delta)k - k') - \mu(k,z) = \beta \sum_{z' \in \mathcal{Z}} V_{k'}(k',z')p(z',z)$$

  where $\mu(k,z)$ is the Lagrange multiplier on the irreversibility constraint.

Irreversible investment

- The envelope theorem is slightly different here

$$V_k(k,z) = (1 + zf'(k) - \delta)u'(zf(k) + (1-\delta)k - g(k,z)) \\ - \mu(k,z)(1-\delta)$$

- So our first order condition is

$$u'(zf(k) + (1-\delta)k - k') - \mu(k,z) \\ = \beta \sum_{z' \in \mathcal{Z}} [(1 + z'f'(k') - \delta)u'(z'f(k') + (1-\delta)k' - g(k',z')) \\ - \mu(k',z')(1-\delta)]p(z',z)$$

Irreversible investment

- Suppose we have guesses $g_n(k,z)$ and $\mu_n(k,z)$. Then find $\tilde{k}'$ as

$$
\begin{aligned}
& u'(zf(k)+(1-\delta)k-\tilde{k}') \\
&= \beta \sum_{z' \in \mathcal{Z}} [(1+z'f'(\tilde{k}')-\delta)u'(z'f(\tilde{k}')+(1-\delta)\tilde{k}'-g_n(\tilde{k}',z')) \\
&\qquad - \mu_n(\tilde{k}',z')(1-\delta)]p(z',z)
\end{aligned}
$$

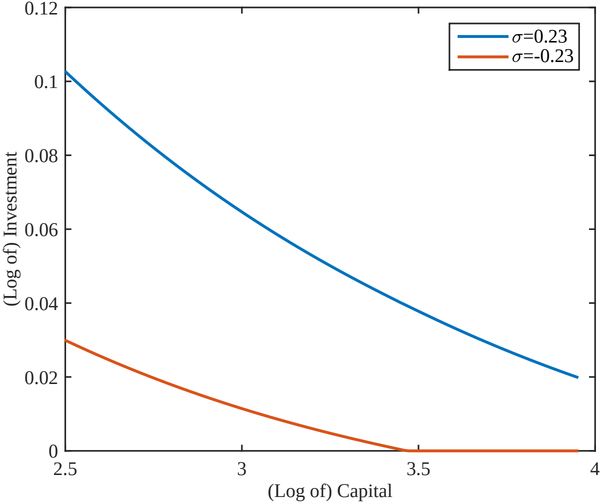
Irreversible investment

- Construct $\tilde{g}_{n+1}(k, z)$ using $\tilde{k}'$.
- Update $g_{n+1}(k, z) = \max\{\tilde{g}_{n+1}(k, z), (1 - \delta)k\}$
- Find $\tilde{\mu}$ as

$$\begin{aligned}\tilde{\mu} = {} & u'(zf(k) + (1-\delta)k - g_{n+1}(k, z)) \\ & - \beta \sum_{z' \in \mathcal{Z}} [(1 + z' f'(g_{n+1}(k, z)) - \delta) u'(z' f(g_{n+1}(k, z)) \\ & \qquad + (1-\delta) g_{n+1}(k, z) - g_n(g_{n+1}(k, z), z')) \\ & \qquad\qquad - \mu_n(g_{n+1}(k, z), z')(1-\delta)] p(z', z)\end{aligned}$$

- Construct $\tilde{\mu}_{n+1}(k, z)$ using $\tilde{\mu}$.
- Update $\mu_{n+1}(k, z) = \max\{\tilde{m}u_{n+1}(k, z), 0\}$

Irreversible investment



Irreversible investment