Methods for solving non-linear, rational expectations business cycle models

A 1-day worskhop for the Bath-Bristol-Exeter Doctoral Training Centre, by Tony Yates

University of Bristol +Centre for Macroeconomics

21 May 21-14

1. Introduction: milestones in history, purpose, curriculum for self-study, plan for today

Purpose

- Work of Samuelson, Bewley, Brock, Lucas, Kydland, Prescott and others
- Emphasis and later dominance of ‘microfounded’, often rational expectations models in macro
- Applies to models of growth as much as business cycles.
- Recent colonisation of finance, development, political economy.
- Also rich micro literature too [eg Pakes...]

Purpose (2)

- Dominant paradigm has generated new industries in applied econometrics
  - Methods for estimating models
  - Search for facts to confront with models
- Purpose is to begin a process of giving you access to these vast literatures

Some of the great debates employing computational methods for macro

- Causes of business cycles. Were they real or not?
- The great inflation, great moderation, great contractions.
- The costs of business cycles.
- Optimal monetary and fiscal policy, inc institutional design.
- Cause of and optimal response to changes in consumption and wealth distribution
- Finance as a source and propagator of business cycles, and of misallocation
- Labour market institutions (eg benefits), welfare and the business cycle
- R&D and growth; finance and growth

Applications of recursive, numerical methods

- IO: optimal entry-exit, pricing.
- Labour: search problems when decision is accept-reject.
- DP^2: optimal government unemployment compensation policy when agents solve an accept-reject search problem in the labour market.
- Monetary policy and the zero bound [I'm working on this now].
- Optimal redistributive taxation with aggregate and idiosyncratic uncertainty

Plan=f(speed)

- Quasi-linear, perfect-foresight method. [DSGE at zero lower bound]
- Nonlinear, perfect-foresight using Newton-Raphson [deterministic RBC]
- Parameterised expectations [stochastic RBC]
- A projection method deploying Chebyshev polynomical function approximation. [Deterministic RBC]
- Dynamic programming with value and policy function iteration [Deterministic and stochastic RBC]
- Dynamic programming using collocation and Chebyshev Polynomials [Deterministic RBC]
- Heterogeneous-agent methods using Krussel-Smith [Deterministic RBC]
- DP^2 – some remarks only.

The vehicle: RBC model

- (Initially) representative agent, consumption-savings-investment-output problem, stochastic, neutral-technological progress.
- Useful building-block for many, more modern and realistic applications in growth, finance, monetary economics.
- Many numerical methods textbooks explain using this as an example.
- But to repeat : methods much more widely applicable.

Minimal tools for applied macro theory

- Dynamic optimisation – to solve the problems of the firms and consumers in your model
- Function approximation using sums of polynomials
- Numerical optimisation
- Matrix algebra
- Numerical derivatives, numerical integrals
- Markov chains
- Dynamic programming

Desirable for applied macro, minimal for theoretical macro

- Real analysis – study of existence and convergence theorems on which dp and fa rely
- Most minor modifications of RBC models will be such that required conditions met.
- Many major modifications –eg heterogeneous agents – leave you in territory where there is no possibility of proving existence or uniqueness, so you have to rely on experimentation, homotopy.

Not covering today:

- Peturbation methods for solving RBC/DSGE models
  - Though we are deploying similar idea to solve rootfinding problems our nonlinear model poses…
- Now easy [maybe too easy!] to do this in Dynare, software written for use in Matlab.
- These are 'local' methods. Adequate unless:
  - Large shocks
  - Occasionally binding constraints or other things inducing kinks in policy functions
  - Choice set for agents discontinuous

Good resources

- Stokey and Lucas (1989) – regularity conditions, convergence theorems for tools to work.
- Judd (1993) – excellent discussion of numerical methods for NLDSGE models
- Heer and Maussner (2005) – great 'how to' book, including some good details on tools
- Adda and Cooper (2003) – nice explanations of dynamic programming, simulated method of moments. A read in the bath tour of macro.
- Miranda and Fackler (2002) – includes powerful toolbox of code.

Good resources (2)

- Wouter den Haan's lecture notes on projection methods, + many others
- Den Haan and Marcet (1990) on parameterised expectations method is excellent and very transparent.
- Wouter den Haan manuscript on PEA
- Christiano and Fisher (2000) on unity of PEA and projection methods.

Software

- Pencil and paper indispensible, but insufficient!
- Matlab, Gauss, Python for development, computation, simulation
- Lower level language like Fortran, C++ for computationally intensive tasks
- Mathematica, Maple for debugging ‘pencil and paper’ calculation of derivatives… [perhaps integrated into your Matlab or other programs]

Companion curriculum on empirical macro

- Time series econometrics; VARs, random processes, Markov processes, ergodicity
- Kalman Filter
  - Filtering. Duality with optimal linear regulator.
  - Computing the likelihood of [the ss form of a] DSGE/RBC model
- Markov-Chain-Monte-Carlo methods
  - Characterising numerically posterior densities
- Particle filtering
  - For when you don’t have analytical expressions for the likelihood

2. Piecewise-linear, perfect-foresight solution of a non-linear RE NK model

[Taken from Brendon, Paustian and Yates, 'The pitfalls of speed-limit interest rate rules at the ZLB'.]

Warm-up: piecewise-linear REE solution method

- Perfect foresight, nonlinear REE
- Jung, Teranishi and Watanabe (2005), Eggertson and Woodford (2003)
- Policy rules in NK model with the zero lower bound to interest rates
- Guess period at which ZLB binds; solve resultant, 2 part linear RE system, verify whether we have an REE

An almost linear New Keynesian RE model

$$\widehat{\pi}_t = \frac{(1-\theta)(1-\beta\theta)(\sigma+\varphi)}{\theta}\widehat{y}_t + \beta E_t\widehat{\pi}_{t+1}$$

$$\widehat{y}_t = E_t\widehat{y}_{t+1} - \frac{1}{\sigma}\left(\widehat{R}_t - E_t\widehat{\pi}_{t+1}\right)$$

$$\widehat{R}_t = \max\left\{\alpha_\pi\widehat{\pi}_t + \alpha_y\widehat{y}_t + \alpha_{\Delta y}(\widehat{y}_t - \widehat{y}_{t-1}), \widehat{R}_L\right\}$$

Why ZLB is of interest

- Rates at the ZLB in Japan for 20 years, and in UK, US for 3.5 years
- Woodford's 'forward guidance policy' copied (?) by central banks is optimal policy at the ZLB
- Fiscal multiplier, and benefits of fiscal policy, heavily dependent on ZLB
- Other applications of piecewise linear too.

Why are speed-limit rules of interest?

- Implement/mimic commitment policy
  - Walsh(2003a), Giannoni and Woodford (2003), McCallum and Nelson(2004), Stracca (2007), Leduc and Natal(2011)
- Insurance against measurement error
  - Orphanides and Williams [various])
- Some evidence they fit time series for central bank rates
  - Mehra (2002)

Calibration of simple NK model

σ	elasticity of intertemporal substitution	1
β	discount rate	0.99
θ	Calvo hazard parameter	0.67
φ	inverse Frisch elasticity of labour supply	2
α_π	weight on inflation in policy rule	1.5
α_y	weight on output in policy rule	0
$\alpha_{\Delta y}$	weight on change in output in p.r.	2

$$\widehat{R}_t = \max\left\{\alpha_\pi \widehat{\pi}_t + \alpha_y \widehat{y}_t + \alpha_{\Delta y}(\widehat{y}_t - \widehat{y}_{t-1}), \widehat{R}_L\right\}$$

Solution algorithm

1. Guess ZLB binds at t=0, but not thereafter.

2. Conventional RE solvers give us:

$$\begin{bmatrix} \hat{\pi}_t \\ \hat{y}_t \\ \hat{R}_t \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \hat{y}_{t-1}, t \geq 1$$

3. Now solve for initial period, substituting out expectations using 2.:

$$\hat{\pi}_0 = k\hat{y}_0 + \beta a_1 \hat{y}_0$$
$$\hat{y}_0 = a_2 \hat{y}_0 - \frac{1}{\sigma}\left(\hat{R}_L - a_1 \hat{y}_0\right)$$

4. Given initial values, use 2. to solve recursively for t=1 onwards...

5. Verify:

$$\hat{R}_0^{shadow} < \hat{R}_L$$
$$\hat{R}_t^{shadow} > \hat{R}_L, t \geq 1$$

Solving the linear RE part

- Method of undetermined coefficients [or similar]
- Conjecture:

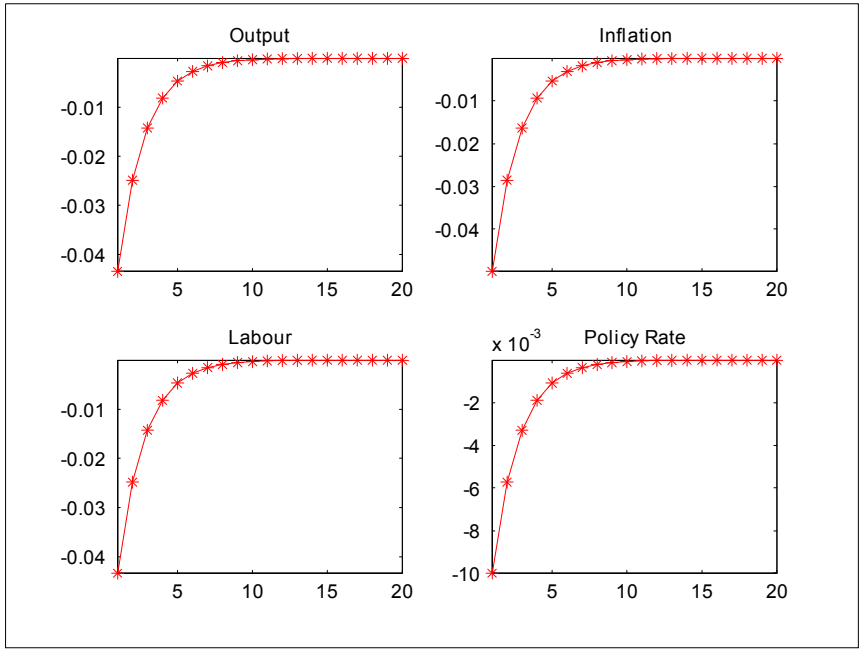
$$\begin{bmatrix} \hat{\pi}_t \\ \hat{y}_t \\ \hat{R}_t \end{bmatrix} = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} \hat{y}_{t-1}$$

- Substitute in where terms in expectations appear
- Solve for the a's.

Recap on nonlinear ZLB solution method in words

- Make a guess at the number of periods for which the zero bound binds.
- Use undetermined coefficients or similar to solve for linear REE in terms of state from this period on.
- Use this solution form to eliminate the expectations terms in the equation system for the initial period.
- This leaves you with a 2-equation 2 unknown system for the initial period, which you can solve. Remember that the interest rate rule is replaced by the assumption that interest rates are zero (from the initial guess).
- Having solved for the initial period, use the solution form for the post zero bound period to simulate forwards step by step.

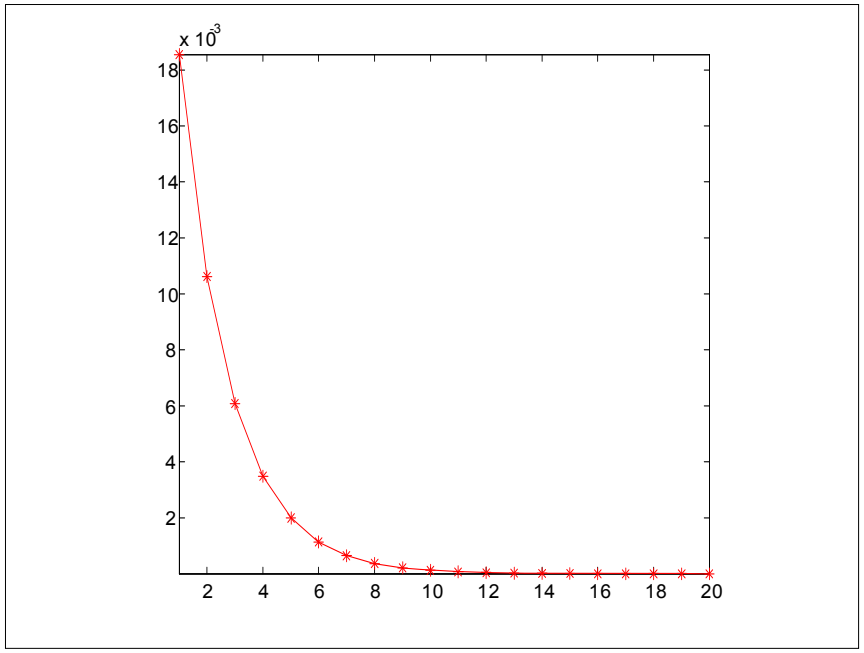
Dynamics under self-fulfilling recession



Self-fulfilling crisis: a simple New Keynesian model

- Inflation and output >4% from ss
- Rates at the ZLB

Real rate under self-fulfilling recession



The real interest rate during the crisis episode

Real rate high in period 2, sustaining forecast of low inflation and output.

General lessons

- Solving nonlinear problems by recasting as a set of linear problem-solving steps.
- Guess and verify.
- NB: this algorithm is a stepping-stone to solving for optimal policy at the zlb.

Why perfect foresight?

- Hopelessly unrealistic?
- May be ok for some questions, eg transition from one SS to another, in response to preannounced policy.
- Useful benchmark.
- Stepping stone to learning other methods.
- Stepping stone to coding other methods, once they are learned.

3. Solving a perfect-foresight, finite-horizon version of the growth model using Newton-Raphson methods

A finite-horizon yeoman-farmer model

$$U(C_0, \ldots C_T) = \left\{ \sum_{t=0}^{T} C_t^{\rho} \right\}^{1/\rho}, \rho \in (-\infty, 1]$$

$$f(K_t) = K_t^{\alpha}, \alpha \in (0, 1)$$

$$K_{t+1} + C_t \leq f(K_t),$$

$$0 \leq C_t,$$

$$0 \leq K_{t+1}, t = 0, \ldots T$$

‘Example 1.1.1, Heer and Maussner, p9.

FONCs for the yeoman-farmer model

$$K_{t+1} = K_t^{\alpha} - C_t, t = 0, 1 \ldots T$$

$$\left(\frac{C_t}{C_{t+1}}\right)^{1-\rho} \alpha K_{t+1}^{\alpha-1} = 1, t = 0, 1 \ldots T-1$$

Euler equation does not hold for period T; no intertemporal choice here, just eat everything.

↑

Use budget constraint to sub out for C terms in Euler equation….

$$C_t = K_t^{\alpha} - K_{t+1}$$

$$\left(\frac{K_t^{\alpha} - K_{t+1}}{K_{t+1}^{\alpha} - K_{t+2}}\right)^{1-\rho} \alpha K_{t+1}^{\alpha-1} = 1$$

$$\Leftrightarrow \left(\frac{K_{t+1}^{\alpha} - K_{t+2}}{K_t^{\alpha} - K_{t+1}}\right)^{1-\rho} - \alpha K_{t+1}^{\alpha-1} = 0$$

The *T* dimensional nonlinear equation system

$$\left(\frac{K_1^\alpha - K_2}{K_0^\alpha - K_1}\right)^{1-\rho} - \alpha K_1^{\alpha-1} = 0,$$

$$\left(\frac{K_2^\alpha - K_3}{K_1^\alpha - K_2}\right)^{1-\rho} - \alpha K_2^{\alpha-1} = 0, ..$$

$$\left(\frac{K_T^\alpha}{K_{T-1}^\alpha - K_T}\right)^{1-\rho} - \alpha K_T^{\alpha-1} = 0$$

- This is a set of *T* nonlinear equations in *T* unknowns, which we are going to solve using a modified Newton-Raphson method, useful in many many contexts.

- This also, like our warm-up example, converts sol of nonlinear equation into sequential solving of linear approximations.

Uni-dimensional Newton-Raphson

$$x^*, s.t. f(x^*) = 0$$

This is our ultimate goal, formalised

$$g^0(x) = f(x_0) + f'(x_0)(x - x_0)$$

We approximate f linearly around some initial point x_0, using first 2 terms of Taylor's theorem.

$$x_1 \ , s.t. g^0(x_1) = 0$$

Then we solve for the x_1 that makes the approximant g(x_1)=0

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$$

The iterative, unidimension NR method

$$x_{s+1} = x_s - \frac{f(x_s)}{f'(x_s)}$$

If we iterate on this equation, then the estimates slowly converge on the roots of the original system.

Can get problem that new guesses lie outside the domain of the function, so we modify by placing bounds on the allowable guesses.

Graphical illustration of uni-dimensional Newton-Raphson

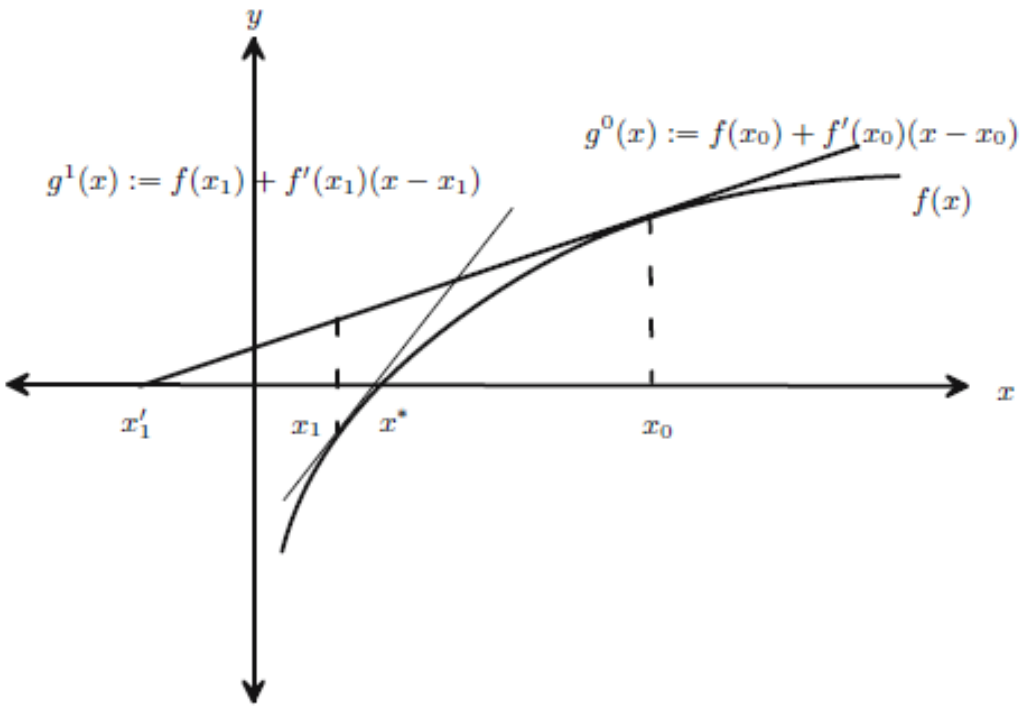


Figure 8.6: Modified Newton-Raphson Method to Locate the Zero of a Single Valued Function

Source: Heer and Maussner, Chapter 8: tools

Multidimensional Newton-Raphson

$$\begin{bmatrix} 0 \\ 0 \\ \dots \\ 0 \end{bmatrix} = \begin{bmatrix} f^1(x_1, \dots x_n) \\ f^2(x_1, \dots x_n) \\ \dots \\ f^n(x_1, \dots x_n) \end{bmatrix} = \mathbf{0} = \mathbf{f}(\mathbf{x})$$

Defining the matrix function f via matrix representation of our system of equations

$$\begin{aligned} f^1() &= \left(\frac{K_1^\alpha - K_2}{K_0^\alpha - K_1}\right)^{1-\rho} - \alpha K_1^{\alpha-1} + 0 * K_3 + 0.K_4 + \dots + 0.K_T \\ f^2() &= \dots \\ &\dots \\ f^{n=T}() &= 0 * K_1 + 0 * K_2 + \dots + \left(\frac{K_T^\alpha}{K_{T-1}^\alpha - K_T}\right)^{1-\rho} - \alpha K_T^{\alpha-1} \end{aligned}$$

In our deterministic yeoman farmer model, the f's will look like this…

Defining the Jacobian, and populating it using the deterministic yeoman farmer eg

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} f_1^1 & f_2^1 & \cdots & f_n^1 \\ f_1^2 & f_2^2 & \cdots & f_n^2 \\ \cdots & \cdots & \cdots & \cdots \\ f_1^n & f_2^n & \cdots & f_n^n \end{bmatrix}, \text{ where } f_j^i = \frac{\partial f^i(\mathbf{x})}{dx_j}$$

Rather than a derivative, we now work with a Jacobian, a matrix of partial derivatives….

$$f_1^1 = \frac{df^1}{dx_1} = \frac{df^1}{dK_1} = \frac{d}{dK_1}\left[\left(\frac{K_1^\alpha - K_2}{K_0^\alpha - K_1}\right)^{1-\rho} - \alpha K_1^{\alpha-1}\right]$$

$$= -(K_1^\alpha - K_2)^{1-\rho}(\rho-1)(K_0^\alpha - K_1)^{(\rho-2)} + (1-\rho)(K_1^\alpha - K_2)^{-\rho}(\alpha K_1^{\alpha-1}) - \alpha(\alpha-1)K_1^{\alpha-2}$$

$$f_2^1 = -(K_0^\alpha - K_1)^{(\rho-1)}(1-\rho)(K_1^\alpha - K_2)^{-\rho}$$

$$\cdots$$

$$f_n^1 = 0, n = 3 \ldots T$$

Defining the first row… most entries in the Jacobian matrix will be zeros.

From linear approximation to our system, to iterative approx to the roots

$$\mathbf{g}(\mathbf{x}) = \mathbf{f}(\mathbf{x}_0) + \mathbf{J}(\mathbf{x}_0)\mathbf{dx}, \mathbf{dx} = \mathbf{x} - \mathbf{x}_0$$

Linear approximation to the multivariate system

$$\mathbf{x}_1 = \mathbf{x}_0 - \mathbf{J}(\mathbf{x}_0)^{-1}\mathbf{f}(\mathbf{x}_0)$$

Solution to g(x)=0 at the point x_0

$$\mathbf{x}_{s+1} = \mathbf{x}_s - \mathbf{J}(\mathbf{x}_s)^{-1}\mathbf{f}(\mathbf{x}_s)$$

Solution to g(x_s+1)=0 at the point x_s; and iterations on this converge globally on the solution x_* to the nonlinear system, with some modifications.

General, multivariate, modified NR algorithm

1. Initialise: choose $\mathbf{x}_0 \in [\underline{\mathbf{x}}, \bar{\mathbf{x}}], i = 0$

2. (i) Compute $\mathbf{J}(\mathbf{x}_i)$,

2. (ii) solve $\mathbf{f}(\mathbf{x}_i) + \mathbf{J}(\mathbf{x}_i)(\mathbf{x}_{i+1} - \mathbf{x}_i) = \mathbf{0}$

2. (iii) if $\mathbf{x}_{i+1} \notin [\underline{\mathbf{x}}, \bar{\mathbf{x}}]$, choose $\lambda \in (0,1)$, s.t. $\mathbf{x}'_{i+1} = \mathbf{x}_i + \boldsymbol{\lambda}(\mathbf{x}_{i+1} - \mathbf{x}_i) \in [\underline{\mathbf{x}}, \bar{\mathbf{x}}]$

2. (iv) set $\mathbf{x}_{i+1} = \mathbf{x}'_{i+1}$

3. check convergence:stop if $\| \mathbf{f}(\mathbf{x}_{i+1}) \| < \epsilon$, else set $i = i + 1$ and go to 2

Modification checks if new guess lies within domain of function.
Note this is HM algorithm 8.5.1, edition 1.

Remarks

- Many solution methods boil down to solving nonlinear optimisation problems like this
- Many alternative methods and refinements.
- Note similarities with before; converting problem of solving nonlinear problem into one of finding ever better solutions to the linear approximations to the nonlinear problem.

Alternative methods and refinements

- Using numerical derivatives based on differences, in this NR method
- Using a ‘secant’ method explicitly based on differences.
- Stochastic modificiations to prevent getting ‘stuck’ at a local. [“simulated annealing”]
- Many pre-coded functions to use, but it helps to know basic method to be able to use them effectively

PS on perfect-foresight and NR

- We used this method in our paper on the zero bound too.
- Instead of linearising the NK model, and then working with the zero bound using a guess and verify method…
- Formulate full, nonlinear NK model, including ZLB, solving system involving finite number of periods, one equation for each period, as you just saw for the RBC model.

What about the infinite horizon RBC model?

- Same method can be used to solve infinite-horizon method, on assumption that we get very close to steady-state in some finite number of periods.

Forward-iteration, perfect foresight with infinite horizon

$$\max_{\{C_t\}} E_0 \sum_{t=0}^{\infty} \beta^t \log(C_t)$$

$$s.t. K_{t+1} = K_t^{\alpha} - C_t$$

$$C_t \geq 0$$

$$K_t \geq 0$$

Now we have discounted utility to make sure the maximand is bounded.

We are going to assume that there is some finite time at which the model should have reached steady state capital, or very close to it.

Then, from some initial conditions, solve for the trajectory to this steady state.

Method the same. Derive EE, combine with resource constraint, formulate system of nonlin equations. Solve root finding problem using NR.

Nonlinear system for ‘forward iteration’

$$0 = \left(\frac{K_1^\alpha - K_2}{K_0^\alpha - K_1} \right) - \alpha\beta K_1^{\alpha-1}$$

$$0 = \left(\frac{K_2^\alpha - K_3}{K_1^\alpha - K_2} \right) - \alpha\beta K_2^{\alpha-1}$$

$$\dots$$

$$0 = \left(\frac{K_T^\alpha - K^*}{K_{T-1}^\alpha - K_T} \right) - \alpha\beta K_T^{\alpha-1}$$

Now we have log utility, and discounting, so slight difference in RHS.

Note before final K was 0; now K_star=steady-state

This is T equations in T unknowns.

K_0 is given.

Iterations needed: if K_T ‘too far’ from ss, then need to lengthen T.

4. Solving the growth model using the parameterised expectations algorithm

Solving RBC model using parameterised expectations

- Den Haan and Marcet (1989)
- Analogy with learning, connection with Marcet's work with Sargent on properties of learning algorithms
- Lots of problems with it, not that widely used now
- But very simple to conceive and program, illustrates the problem nicely, and an introduction into general class of 'projection' methods

The RBC or 'growth' model

$$\max_{\{c_t,k_t\}_{t=0}^{\infty}} E_t[\sum_{t=0}^{\infty} \beta^t \frac{c_t^{1-\gamma}-1}{1-\gamma}]$$

subject to :

$$c_t + k_{t+1} = z_t k_t^{\alpha} + (1-\delta)k_t$$

$$\ln(z_t) = \rho \ln(z_{t-1}) + \sigma e_t, e_t \sim N(0,1)$$

$$c_t = c(k_t, z_t), k_{t+1} = k(k_t, z_t)$$

Solution is a pair of functions linking the choice variables (consumption and capital for use tomorrow) to the state variables [shock z, and capital today]

$$c_t^{-\gamma} = E_t[\beta c_{t+1}^{-\gamma}(\alpha k_{t+1}^{\alpha-1} + 1 - \delta)]$$

$$c_t + k_{t+1} = z_t k_t^{\alpha} + (1-\delta)k_t$$

Solution has to satisfy the Euler equation [in more complex models, the full set of FONCs] and the resource constraint.

In the special case of log utility [gamma=1] and full depreciation [delta=1], these solutions can be calculated analytically and are known to be:

$$c_t = (1-\alpha\beta)z_t k_t^{\alpha}, k_{t+1} = \alpha\beta z_t k_t^{\alpha}$$

Choosing the approximant: expectations in the Euler equation

$$c_t^{-\gamma} = E_t[\beta c_{t+1}^{-\gamma}(\alpha k_{t+1}^{\alpha-1} + 1 - \delta)]$$

We approximate the conditional expectation on the RHS of the Euler equation above with a polynomial in the state variables….

$$g(k_t, z_t) = e^{a_1 \ln k_t + a_2 \ln z_t}$$

NB 1. may be desirable to use higher order polynomial

2. may be a choice about which function to approximate [as here], or more than one function you have to approximate. Eg may be more than one agent forming expectations!

PEA algorithm

1. Draw a long sequence of values for shocks e_t to generate the exogenous technology shock process z_t. (Long=100,000?)
2. Choose a starting value for a_1, a_2 and a starting value for k_0, perhaps the steady state.
3. Simulate the model by:
   3.1 computing $c_0^{-\gamma} = e^{a_1 \ln k_0 + a_2 \ln z_0}$
   3.2 solving for k_1 from the resource constraint, thus: $k_1 = z_0 k_0^{\alpha} + (1-\delta)k_0 - c_0^{-\gamma}$
   3.3. push time forward one period and return to 3.1
4. Compute 'residuals' by:
   4.1 for each t, letting $c_t^{true} = \beta c_{t+1}^{-\gamma}(\alpha k_{t+1}^{\alpha-1} + 1 - \delta), c_t^{app} = e^{a_1 \ln k_t + a_2 \ln z_t}$
   4.1 $R_t = c_t^{true} - c_t^{app}$
5. Update a_1, a_2 by choosing them to min $\sum_{t=0}^{T} R^2$ [nonlinear least squares]
6. Check convergence. If converged, stop, else return to 3.

PEA in words

- 1. Parameterise the expectations function, initialise coefficients
- 2. Draw one very long sequence of shocks
- 3. Simulate the model.
- 4. Find a new set of coefficients for 1. using non linear least squares, by comparing the polynomial prediction for c_t with the simulated one.
- 5. Check for convergence. If not converged, go to 3 and continue

Problems with PEA

- Multicollinearity when you try to get more accurate approximations by including higher order terms in the polynomial.
  - Eg: The square of the shock is correlated with the shock itself.
- Solution accurate only close to steady state, since simulations live mostly there.
- Potential for instability in the updating algorithm
  - Analogy with intertemporal learning models, and projection facilities for solving the problem.

Resolving PEA convergence problems

- Homotopy: use solutions to problems you can solve to inform starting values to problems you haven't yet solved
- Projection facilities and damped updating: slow down updating process in PEA algorithm.

PEA step with a projection facility, or damped updating

$$5.1 : a^{p}_{1j+1}, a^{p}_{2,j+1} = \arg\min(\sum_{t=0}^{T} R^2)$$

$$5.2 : a_{i,j+1} = \lambda a_{i,j} + (1-\lambda) a^{p}_{i,j+1}$$

Trade-off:
Too-slow updating risks getting stuck away from the solution.
Too-fast updating risks inducing instability in the algorithm and not finding the solution.

Example of (not very useful) homotopy algorithm

Objective: solve RBC model defined by…

$$\delta = \delta^f, \gamma = \gamma^f$$

Initialise parameters:
RBC parameters $\gamma_0 =\rightarrow 1, \delta_0 = 1$
PEA parameters:
i) solve model analytically, giving $c_t^0(z_t, k_t), k_{t+1}^0(z_t, k_t)$
ii) simulate model giving sequences c^0, k^0
iii) initialise values of a_1^0 and a_2^0 in $\widehat{c_t^{-\gamma}} = e^{a_1 \ln k_t + a_2 \ln z_t}$ that minimise the gap between c_t simulated in ii) and $\widehat{c_t^{-\gamma}}$
Counter: set $i = 1$

Main loop:
1. Set $\gamma_i, \delta_i = (\gamma_{i-1}, \delta_{i-1}) + \lambda[(\gamma_f, \delta_f) - (\gamma_0, \delta_0)]$
5. Solve for $c_t^1(z_t, k_t), k_{t+1}^1(z_t, k_t)$ using PEA, initialising $(a_1, a_2) = a_1^{i-1}$ and a_2^{i-1}, saving on convergence (a_1^i, a_2^i)
6. If $(\gamma, \delta) \neq (\delta^f, \gamma^f)$, set $i = i + 1$ and go to 1, else you are done.

Homotopy

- General procedure: use solution to a model you know how to solve well, to solve model you don’t.
- Remark: even if not formally doing homotopy, often wise to build up to complex models by solving simpler versions, to gain insight, test code.

Remarks

- Analogy between intra-temporal process of iterative algorithm to find best approximate expectations function…
- …and in*ter*-temporal learning with agents updating expectations functions as data becomes available each period.
- Similarity not just superficial; maths of convergence or lack of it has connections.

Inter and intra temporal learning analogy

- Consumers start with an expectations function.
- Take decisions.
- Data realised.
- Agents compute ‘surprise’. New function updated as function of surprise, and imprecision in estimates.
- Gain ‘damps’ updating.
- Under some conditions, converges to REE.

5. Solving RBC model using collocation, and Chebyshev polynomials

Overview of this section

- Preliminary on approximation using Chebyshev polynomials
- Solving RBC models using Cheb Polys to approximate the expectations function.
- Iterating on the Bellman Equation using Cheb Polys to approximate the value function.

5.1 Function approximation and Chebyshev polynomials

Intro remarks

- Function approximation is a technique widely encountered in theory, econometrics, numerical analysis.
- We encountered it already in using linear approximations in Newton-Raphson to find roots of nonlinear equation system.
- Regression is function approximation.
- Fourier series approximation used to compute the ‘spectrum’.
- Here: we will approximate the expectations function in consumers’ FOC.

Typical function approximation problem, and solution

$$f(x) \in C[a,b]$$

We want to approximate some *f* which is continuous over the a,b interval

$$\widehat{f}(x) = \sum_{i=1}^{n} \alpha_i \varphi_i(x)$$

$$\Pi = \{\varphi_0(x), \varphi_1(x), \ldots\}$$

We do it by taking some weighted combination of basis functions, eg a family of polynomials

$$f(x) \simeq \widehat{f}(x) = \sum_{i=1}^{\infty} \alpha_i \varphi_i(x), \varphi_i(x) = x^i$$

For example, we can represent any continuous function EXACTLY with this infinite sum of polynomials…

$$f(x) \simeq \widehat{f}(x) = \alpha_0 + \alpha_1 x \quad + \alpha_2 x^2 + \ldots \alpha_p x^p$$

Which in practice means using the first p terms.

Chebyshev Polynomials

$$T_i(x) = \cos(i \arccos x)$$

General expression for the C.P. of order i.

$$T_{i+1}(x) = 2xT_i(x) - T_{i-1}(x)$$

They can be defined recursively.

Here are the first four C P's.

$$T_0(x) = \cos(0.\arccos x) = 1$$
$$T_1(x) = \cos(1.\arccos x) = x$$
$$T_2(x) = 2xT_1(x) - T_0(x) = 2x^2 - 1$$
$$T_3(x) = 2xT_2(x) - T_1(x) = 4x^3 - 3x$$

Graph of 1st 3 Chebyshev polynomials

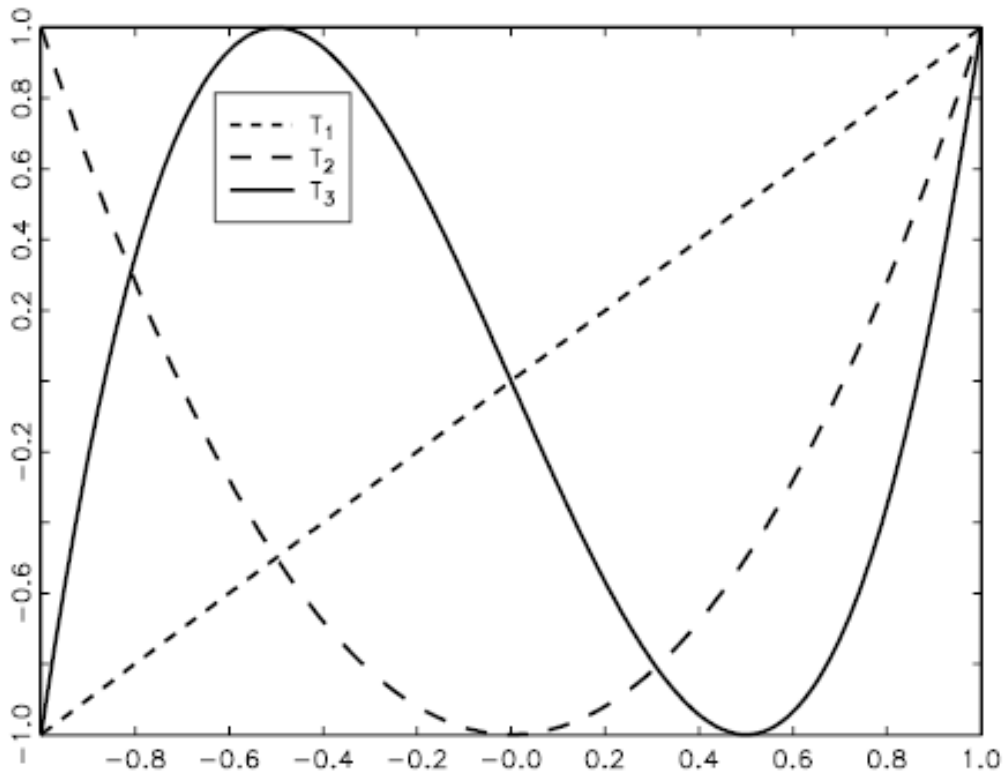


Figure 8.2: Chebyshev polynomials T_1, T_2 and T_3

Source: Heer+Maussner, ch 8, p 434

Domains of the CP and your f(x)

$$f(x) \ : \ [a,b]$$
$$g(x) = f(Z(x)) \ \ : [-1,1]$$

$$X(z) = \frac{2z}{b-a} - \frac{a+b}{b-a}, z \in [a,b]$$
$$Z(x) = \frac{(x+1)(b-a)}{2} + a, x \in [-1,1]$$

CPs only defined on -1,1. X(z) converts values defined on a,b into values defined on -1,1.

Z(x) does the reverse.

$$\widehat{f}(z;\alpha) = \sum_{i=0}^{n} \alpha_i T_i(X(z))$$

This will be our approximating function.
The CPs will take as an input transformations of the original points in our function.

Digression: deriving the Euler equation in the deterministic RBC model

$$\max_{\{c_0, c_1 \ldots\}} U = \sum_{t=0}^{\infty} \beta^t \frac{c_t^{1-\eta} - 1}{1-\eta}, \beta \in (0,1), \eta > 0$$

subject to $k_{t+1} + c_t \leq k_t^{\alpha} + (1-\delta)k_t, \alpha \in (0,1)$

$$0 \leq c_t$$

$$0 \leq k_{t+1}$$

k_0 given

Non-logarithic preferences, and the presence of only partial depreciation, will make an analytical solution impossible.

But as a first step we have to derive the Euler equation anyway; though that still leaves the task of solving for c(k), the policy function.

Forming and differentiating the Lagrangian in the RBC model

$$L = \sum_{t=0}^{\infty} \beta^t \left[\frac{c_t^{1-\eta}-1}{1-\eta} - \lambda_t [k_{t+1} + c_t - k_t^{\alpha} - (1-\delta)k_t] \right]$$

Lagrangian for the consumer's problem.

$$\begin{aligned} \frac{dL}{dc_t} &= 0 = \beta^t [(1-\eta)\frac{c_t^{-\eta}}{1-\eta} - \lambda_t] \\ &\Leftrightarrow 0 = \beta^t [c_t^{-\eta} - \lambda_t] \\ &\Leftrightarrow c_t^{-\eta} = \lambda_t \end{aligned}$$

Which we differentiate wrt today's consumption…

and tomorrow's capital…

FOC wrt capital to the Euler equation

$$\frac{dL}{dk_{t+1}} = \frac{d}{dk_{t+1}}\left[\begin{array}{c}\beta^t\left[\frac{c_t^{1-\eta}-1}{1-\eta} - \lambda_t[k_{t+1} + c_t - k_t^{\alpha} - (1-\delta)k_t\right] \\ +\beta^{t+1}\left[\frac{c_{t+1}^{1-\eta}-1}{1-\eta} - \lambda_{t+1}[k_{t+2} + c_{t+1} - k_{t+1}^{\alpha} - (1-\delta)k_{t+1}\right]\end{array}\right]$$

$$= \beta^t[-\lambda_t] + \beta^{t+1}[\lambda_{t+1}\alpha k_{t+1}^{\alpha-1} + \lambda_{t+1}(1-\delta)] = 0$$

$$\Leftrightarrow 0 = [-\lambda_t] + \beta \quad \lambda_{t+1}[\alpha k_{t+1}^{\alpha-1} + (1-\delta)]$$

$$\Leftrightarrow \frac{\lambda_t}{\lambda_{t+1}} = \beta[\alpha k_{t+1}^{\alpha-1} + (1-\delta)]$$

$$\Leftrightarrow \text{[sub in foc wrt } c_t\text{]} \frac{c_t^{-\eta}}{c_{t+1}^{-\eta}} = \beta[\alpha k_{t+1}^{\alpha-1} + (1-\delta)]$$

$$0 = \beta[\alpha k_{t+1}^{\alpha-1} + (1-\delta)]\left(\frac{c_{t+1}}{c_t}\right)^{-\eta} - 1$$

We differentiate *L* wrt *k_t+1*, set=0, then substitute in FOC wrt consumption to eliminate the L. multipliers, arriving at the Euler equation.
This is really a system of Euler equations for every t, which the policy function *c(k)* has to satisfy. This policy function is what we will approximate using CP's.

Solving for the steady state for capital

$$0 = \beta[\alpha k^{\alpha-1} + (1-\delta)]\left(\frac{c}{c}\right)^{-\eta} - 1$$

$$1 - \delta + \alpha k^{\alpha-1} = 1/\beta$$

$$k^{\alpha-1} = \frac{1}{\beta\alpha} - \frac{1}{\alpha} + \frac{\delta}{\alpha}$$

$$k^{\alpha-1} = \frac{1-\beta(1-\delta)}{\beta\alpha}$$

$$k^{1-\alpha} = \frac{\beta\alpha}{1-\beta(1-\delta)}$$

$$k^* = \left[\frac{\beta\alpha}{1-\beta(1-\delta)}\right]^{\frac{1}{1-\alpha}}$$

Start with the Euler equation.

Drop the time subscripts, since in the steady state all variables are constant.

Then solve for k-star in terms of the model's primitive parameters.

This value is going to help us define good bounds for capital when solving the dynamic RBC model.

Bounding the state space

$$X = [\underline{k}, \bar{k}]$$

$$\bar{k} = 1.5k^*, \underline{k} = 0.5k^*$$

Set bounds experimentally, bracketing the steady state solution for capital.

The residual function

$$\min_{\gamma} \int_{\underline{k}}^{\bar{k}} R(\gamma, k)^2 \, dk$$

Like all projection methods, we start with the objective of minimising the integral of a residual function, evaluated across the state space, as a function of the params that define the approximating function.

$$S(\gamma) = \frac{\pi(\bar{k}-\underline{k})}{2L} \sum_{l=1}^{L} R(\gamma, k(\tilde{k}_l))^2 \sqrt{1-\tilde{k}_l^2}$$

The k_tilde_ls are mapped into k_up, k_down.

This integral is going to be approximated using a sum, evaluated at points corresponding to the zero's of the Chebyshev Polynomial. An example of 'quadrature', a form of numerical integration.

Remarks

- Note there are TWO approximations going on.
  - We are approximating the policy function *c(k)* using Chebyshev polynomials
  - And, in order to find the best such approximation…….
  - we are approximating the integral of the residual function defined over the capital space with a sum, using quadrature.

Computing the residual function for some arbitary k_0

$$R(\gamma, k_0) = \left[\frac{\hat{c}_1}{\hat{c}_0} \right]^{-\eta} (1 - \delta + \alpha k_1^{\alpha-1}) - 1$$

Idea is that if we have got the gamma's right in the approximating function, ie if we have a good approximation to the policy function….

…then the Euler Equation should be close to holding. If it does then the LHS=0.

But then we need an algorithm for computing c_hat_0,c_hat_1 and k_1….

Computing the residual function

$$R(\gamma, k_0) = \left[\frac{\hat{c}_1}{\hat{c}_0} \right]^{-\eta} (1 - \delta + \alpha k_1^{\alpha - 1}) - 1$$

The EE-based residual function we are trying to compute.

1. compute $\hat{c}_0 = \hat{c}(\gamma, k_0)$
2. compute $k_1 = k_0^{\alpha} + (1 - \delta) k_0 - \hat{c}_0$
3. compute $\hat{c}_1 = \hat{c}(\gamma, k_1)$

Given c_hat_0(k_0), we get k_1 from the resource constraint.

Then we evaluate c_hat_1(k_1).

$$\hat{c}(\gamma, k_0) = \sum_{j=0}^{p} \gamma_j T_j(\tilde{k}(k_0))$$

…Using the CP formula here. Where k_tilde(k_0) maps the latter into the -1,1 interval over which the CP is defined.

Not quite done!

- We have to minimise that approximate integral of the [function of the] residual function, over the capital space.
- ….Using some minimisation routine!
- We have seen elements of how to do this when we used NR methods to find the zeros of a nonlinear equation system.
- Can write a function that computes the sum, then use fminsearch, or csminwel to minimise it.

Remarks

- The output of this procedure is a function that specifies what *c* is, given some inherited level of *k*.
- Remember this was the deterministic growth model.
- If we want shocks, we then have a two dimensional state space, and need to do CP in two dimensions.

6. Intro to dynamic programming, value and policy function iteration

Sargent+L: ‘Imperialism of recursive methods’

- DP very useful.
- Details of conditions under which its lessons hold, and numerical methods to operationalise it work, are hard. Avoided here.
- Implementation in simple settings easy!
- Essential if eg agents’ choices are discontinuous, when Lagrangian methods break down. [accept/reject, enter or not, exit or not, bus or train]

Overview of Dynamic Programming section

- 1. Deterministic, finite-element dynamic programming.
- 2. Stochasic, finite element DP
- 3. Continuous state methods using collocation.
- Along the way: Markov-Chain approximations to continuous random processes.
- Few words about DP^2

6.1 Dynamic programming: general setting

General setting

Choose $\{u_t\}_{t=0}^{\infty}$ to

$$\max \sum_{t=0}^{\infty} \beta^t r(x_t, u_t)$$

s.t. $x_{t+1} = g(x_t, u_t); x_0 \in R^n$ given

Choose a sequence of control settings (u), such that a discounted sum of returns (r) is maximised, subject to law of motion for the state (x)

We could be thinking of a consumer, or a firm, [and!/] or a large agent like a policymaker setting policy subject to laws of motion generated by the solutions to the small agents problem [aggregated].

Musings on *h* and *V*

$$u_t = h(x_t)$$
$$x_{t+1} = g(x_t, u_t)$$

Dynamic programming turns problem into search for V ['value function'] and h ['policy function'] such that if we iterate on these 2 equations…… the term on RHS of RHS is maximised.

$$V(x_0) = \max_{\{u_s\}_{s=0}^{\infty}} \sum_{t=0}^{\infty} \beta^t r(x_t, u_t)$$

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

If we knew V, we could compute the RHS for different u's and therefore find h.

Of course, we don't.

The Bellman equation

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

V(today's state) and V(tomorrow's) are linked by the Bellman Equation above. Remember g is the transition law that produces tomorrow's state from today.

$$V(x) = r(x,h(x)) + \beta V(g(x,h(x)))$$

The policy function *h* is a function that satisfies the max operater above, in which case if we substitute in for u, we can get rid of the max operator.

‘Cake-eating’ problem

- Can’t have your cake and eat it. Rather, can’t eat your cake and have it!
- How quickly should I eat my cake, given that it rots, that I don’t want to go hungry, but tomorrow may never come.
- Bellman Equation: suppose that from tomorrow, I have an optimal cake eating plan. How much should I eat now?

‘Contraction mapping’ yielded by [iterations on] the Bellman equation

$$V_{j+1} = \max_u \{r(x,u) + \beta V_j(\tilde{x})\}$$

Under some conditions, starting from any guess of *V*_0, even *V*_0=zeros, iterating on the Bellman equation will converge to give the value function. This is called ‘value function iteration’. As a bi product of the max on the RHS it produces the policy function *h.*

As we will see we can also iterate on the Policy function. ‘Policy function iteration’.

6.2 Deterministic, analytical DP in the growth model

Deterministic VFI in an RBC model we can solve analytically

$$\max \sum_{t=0}^{\infty} \beta^t u(c)$$

$$\text{s.t. } c_t + k_{t+1} = y_t + (1-\delta)k_t$$

$$y_t = f(k_t)$$

$$k_0 \text{ given}$$

$$u = \ln(c)$$

$$\delta = 1$$

$$y_t = Ak^{\alpha}$$

Deterministic: note no expectations term, and no random productivity proess.

Cobb-Douglas production.

Log utility.

Full depreciation of capital k.

Deterministic VFI in the RBC model

$$V_0(k) = 0, \forall k$$

$$V_1(k) = \max_{c,k'} \{\ln(c \quad) + \beta.0\}$$

$$c(k) = Ak^{\alpha}$$

$$V_1(k) = \ln(Ak^{\alpha}) = \ln A + \alpha \ln k$$

We start with any old guess at V, V_0

We plug it into the Bellman Equation.

As the BE instructs, we maximise it wrt to choice variables c,k'

This gives us a new guess at the value function V_1. [Now quite different from V_0.

We repeat the process over and over until convergence.

2nd iteration on the Bellman Equation

$$
\begin{aligned}
V_2(k) &= \max_{c,k'}\{\ln(c) + \beta[\ln A + \alpha \ln k']\} \\
&= \max_{c,k'}\{\ln(Ak^\alpha - k') + \beta[\ln A + \alpha \ln k']\} \\
&\Rightarrow \frac{-1}{Ak^\alpha - k'} + \frac{\beta\alpha}{k'} = 0 \\
&\Rightarrow Ak^\alpha - k' = \frac{1}{\beta\alpha}k' \\
&\Rightarrow Ak^\alpha = k'[1 + \frac{1}{\beta\alpha}] \\
k' &= \frac{Ak^\alpha}{1 + \frac{1}{\beta\alpha}} \\
&= \frac{\beta\alpha Ak^\alpha}{1 + \beta\alpha} \\
&\Rightarrow c = Ak^\alpha - \frac{\beta\alpha Ak^\alpha}{1 + \beta\alpha} \\
&\Rightarrow c = \frac{Ak^\alpha}{1 + \beta\alpha}
\end{aligned}
$$

We have substituted in our V_1 guess.

Now we have to maximise this expression wrt c,k'

Evaluated at these maximised values, we have our next guess at the value function, V_2

As a bi-product, we have the guess at the policy functions (expressions for c and k' in terms of k

Completing the 2nd iteration on the Bellman Equation

$$V_2(k) = \ln\frac{Ak^{\alpha}}{1+\beta\alpha} + \beta[\ln A + \alpha\ln\frac{\beta\alpha Ak^{\alpha}}{1+\beta\alpha}]$$

$$= \ln\frac{A}{1+\alpha\beta} + \beta\ln A + \alpha\beta\ln\frac{\beta\alpha A}{1+\beta\alpha} + \alpha(1+\alpha\beta)\ln k$$

This is our V_2 guess once we have completed the maximisation, [hence no max operator now, since this is done!].

Note that it is quite different from V_1

In a computer program, we would use the difference between each iteration's guess to decide whether we should keep going or not.

A 3rd iteration?!

$$V_3(k) = \max_{c,k'} \{\ln c + \beta V_2(k')\}$$

$$= \max_{c,k'} \{\ln Ak^{\alpha} - k') + \beta[\ln \frac{A}{1+\alpha\beta} + \beta \ln A + \alpha\beta \ln \frac{\beta\alpha A}{1+\beta\alpha} + \alpha(1+\alpha\beta)\ln k']\}$$

Continuing, we would : differentiate wrt k', solve for c, k' that maximise; plug in the maximised values, to produce V_3. Then substitute in the BE again to get an expression for V_4…. And so on.

Final expression for value and policy functions

$$V(k) = (1-\beta)^{-1}\{\ln[A(1-\beta\alpha)] + \frac{\beta\alpha}{1-\beta\alpha}\ln(A\beta\alpha)\} + \frac{\alpha}{1-\beta\alpha}\ln k$$

$$c(k) = (1-\beta\alpha)Ak^{\alpha}$$
$$k'(k) = \beta\alpha Ak^{\alpha}$$

We would need ininitely many iterations to converge, but use of the algebra of geometric series that emerge can produce these expressions.

There is also a guess and verify way to solve for V, but doing it this way illustrates some of the aspects of numerical iterations on the BE.

Remarks on VFI

- Only have convergence under certain conditions. [See Stokey and Lucas].
- Can iterate on the policy function instead, under more restrictive conditions.
- Can use combination of the two to speed computation.
- In general won't have analytical expressions for derivatives we used at each step [and of course no analytical expressions for V,c,k']

6.3 Policy function iteration

Policy function iteration algorithm

1. compute feasible $u = h_0(x)$
2. compute value of pursuing this forever $V_{h_j}(x) = \sum_{t=0}^{\infty} \beta^t r(x_t, h_j(x_t))$

   with $j = 0, x_{t+1} = g[x_t, h_j(x_t)]$
3. Generate new policy $u = h_{j+1}(x)$ that solves the 2 period problem

   $\max_u \{r[x,u] + \beta V_{h_j}[g(x,u)]$
4. Iterate on $1-3$ ro convergence.

Remarks on policy function iteration

- Conditions under which it converges are more restrictive.
- But when these conditions are satisfied, it may converge faster.

6.4 Numerical, deterministic dynamic programming in the growth model

Numerical Dynamic Programming in Matlab or similar

$$k = \begin{bmatrix} k_l \\ k^l + 1/d * (k^u - k^l) \\ k^l + 2/d * (k^u - k^l) \\ \dots \\ k_u \end{bmatrix}$$

$$k^l = 0.5k^{ss}, k^u = 1.5k^{ss}$$

$$V_0(k_i) = [0, 0, 0 \dots]$$

$$g(k_i) = [1, 1, 1, \dots]$$

We start out by defining a vector of grid points for capital, *k*, and initialising vectors for the value function and the policy functions. The policy function vectors will contain numbers 'pointing' us to different elements of the capital grid. Ie *g(1)=5* would mean if we inherit the first element of the capital grid, the best choice for capital tomorrow, *k'* would be the fifth one.

A loop to compute the first new iteration on the Bellman Equation in Matlab or similar

$$V_1(k_i) = \max\{u(c_j) + \beta V_0(k_j)\}$$

1. set $k = k(1) = k_l \Rightarrow y = Ak_l^{\alpha}$
2. evaluate $u(c_j) + \beta V_0(k_j)$ for each possible c_j, k_j'
3. choose highest [in this case $c = Ak^{\alpha}, k' = 0$], assign this number to $V_1(1)$
4. go back to 1. and repeat for next value of $k, k(2) = k^l + 1/d * (k^u - k^l) \ldots$

Subsequent loops are the same – with exception that once we have dispensed with *V=0* the V is going to influence the search for the maximising values of *k',c*.

And we will keep going until we have measured and decided on convergence.

Stopping criterion for a value function iteration loop

$$\text{stop if } \max V_{diff} = \| V_k - V_{k+1} \| < \epsilon$$

After each new iteration, we will compare the old and new *V*s element by element, and compute the maximum difference, and stop provided the absolute value of this difference is less than a pre-specified tolerance value.

This is done by using a 'while abs(max(vdiff))<e …..end' loop in Matlab.

Remarks on numerical DP

- The maximisation step can be very time consuming. Computer has to enumerate and check each candidate Value for each policy choice.
- Convergence greatly speeded up by good starting values.
- Or even missing out the max step. [‘Accelerator’].
- Syntax of loops can be important: ‘vectorising’

Remarks on numerical DP

- Alternative stopping criteria is when the policy functions [vector (in our example) of pointers] does not change.
- That is, if you aren't interested in V itself.
- Simulating the model just requires the pointers.
- The eg we worked out allows us to trace trajectory from initial k, to steady state.

6.5 Stochastic dynamic programming in the growth model

Stochastic dynamic programming

- So far we have assumed there are no shocks. So we can't solve and simulate the classic RBC model, driven by technology shocks.
- To do this we have to enlarge the state space to include a dimension for the shocks.
- So our $V=V(k,A)$
- Finite-element approximation of a continuous random process using Markov-chains.

Digression on Markov chains

- Some economic contexts best described with finite element random processes [regimes?]
- Continuous random processes well approximated by Markov chains [Tauchen (1986)]

Markov chains

$$prob(x_{t+1}|x_t, x_{t-1}, \ldots x_{t-k}) = prob(x_{t+1}|x_t)$$

Markov property

π_0

Z

P

3 objects needed to define a Markov chain. Initial probabilties; vector storing possible values for the chain; *matrix* defining transition probabilities

$\pi_0' P^k$

Computing probabilities at t+k

Ergodic, or stationary distribution of a Markov chain

$$\pi'_k = \pi'_{k-1} P$$

We can iterate on this equation to find the stationary distribution

$$\pi' = \pi' P \Leftrightarrow (I - P)\pi = 0$$

..which happens to solve this equation.

$$p_{ij} > 0 \forall i,j$$

$$p_{ij}^k > 0 \forall i,j; P^k = P * P * \ldots * P \text{ [}k\text{ times], some } k \geq 1$$

Such a distribution exists, and is independent of pi_0 if either of these two conditions holds. Basically requires can't randomly get stuck in one of the states.

Defining a Markov chain for technology

$$A = \begin{bmatrix} 2 \\ 1 \\ 0.5 \end{bmatrix}$$

Technology can take three values: high, medium and low.

$$P = \begin{bmatrix} 0.3 & 0.3 & 0.4 \\ 0.8 & 0.1 & 0.1 \\ 0.2 & 0.75 & 0.05 \end{bmatrix}$$

Prob of going from high to low is 0.4;
Prob of staying in medium if you start in medium is 0.1….

$$\pi_0 = [0.1, 0.1.0.8]'$$

The initial period probabilities of high, medium and low.

Simulating a Markov chain for technology

1. draw x as uniform random number, ie $0 \leq x \leq 1$

2. if $0 \leq x \leq \pi_0(1), A_1^o = A(1);$

if $\pi_0(1) < x \leq \pi_0(2), A_1^o = A(2);$

if $\pi_0(2) < x \leq \pi_0(3), A_1^o = A(3)$

Proc for simulating initial period technology

1. set $t = t + 1$

2. draw x uniform

3. let $s =$ technology state for previous period

3 if $0 \leq x \leq P_{s1}, A_{t+1}^o = A(1);$

if $P_{s1} < x \leq P_{s2}, A_{t+1}^o = A(2);$

if $P_{s2} < x \leq P_{s3}, A_{t+1}^o = A(3).$

4. go to $1\ldots$

Proc for simulating subsequent realisations for technology

Stochastic version of the Bellman equation

$$V_{j+1} = \max_u \{r(x,u) + \beta E[V_j(\tilde{x}|x)\}$$

Our general case modified to include shocks.

$$V_{j+1}(k,A) = \max_{c,k'} \{\ln c + \beta E[V_j(k',A'|A)\}$$

Our RBC case with shocks.

$$V_{j+1}(k,A_m) = \max_{c,k'} \left\{ \ln c + \beta \sum_{i=1}^{3} p_{mi}[V_j(k',A_i'|A_m) \right\}$$

Expanding the expectations operator in our case of a discrete state Markov chain used to approximate the shocks.

The new two-dimensional value function with random, 3-state technology

$$V_0(k,A) = \begin{bmatrix} V_0(k = k^l, A = A(1)) & V_0(k = k^l, A = A(2)) & V_0(k = k^l, A = A(3)) \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots \\ V_0(k = k^u, A = A(1)) & V_0(k = k^u, A = A(2)) & V_0(k = k^u, A = A(3)) \end{bmatrix}$$

Value function is now a matrix storing values corresponding to inherited outcomes for k from k_l to k_h, and in each of these cases, for inherited productivity levels from A(high=1_ to A(low=3).

Filling one element of the Bellman Equation with a 3-state Markov chain for technology

$$V_{j+1}(k^l, A(3)) = \max_{c,k'} \left\{ \begin{array}{c} \ln c + \beta[0.2 * [V_j(k', A' = A(1)|A = A(3))] + \\ 0.75 * [V_j(k', A' = A(2)|A = A(3))] + \\ 0.05 * [V_j(k', A' = A(3)|A = A(3))] \end{array} \right\}$$

Remarks on approximating AR(1) for technology with a Markov chain

- Before you implement your value function iteration, you will have to approximate technology with the MC.
- Original method due to Tauchen (1986).
- The more states, the more accurate the approximation.
- In the limit, the approximant can be made equal to the continuous counterpart.
- But, with more elements, the more time consuming will be the VFI for a given capital grid size.
- Or, to hold computing time constant, you will have to make do with a coarser grid for capital.

6.6 ‘Dynamic programming squared’

- Policymaker devising optimal unemployment compensation; agents solving accept/reject search problem.
- Nested value functions. Each agent’s VF is a function of the others.
- These problems can be solved. LQ examples of optimal policy in optimising RBC/NK models.
- Solved iteratively. Guess VF for one agent; iterate on the other agents’ BE. Then swap.

6.6 Solving growth model using Chebyshev collocation to solve the Bellman equation.

Basic strategy

- Approximate unknown [in this case value] function with finite combination of *n* known basis functions, coefficients on which to be determined…
- Require this approximant to satisfy the functional equation [in this case the Bellman Equation] at *n* prescribed points of the domain, known as the collocation nodes.

Functional equations and collocation: remarks

$$V(k,a) = \max_{k'} \{u(c) + \beta EV(k', a')\}$$

This is a *functional equation*.

Regular equation problem gives us a known function, an asks us to find a value such that some condition is met, eg find x, such that f(x)=0

A functional equation problem asks us to find an *unknown* function [here V(.)] such that some condition is met [here the condition stipulated in the Bellman Equation]

Collocation here converts functional equation problem into a regular equation problem.

From the functional equation to the collocation equation

$$V(k) \approx \sum_{j=1}^{n} c_j \phi_j(k)$$

$$\sum_{j=1}^{n} c_j \phi_j(k_i) = \max_k \left\{ u(k') + \beta E \sum_{j=1}^{n} c_j \phi_j(k') \right\}, i = 1 \ldots n$$

Approximate *V* with the weighted sum of Chebyshev Polynomials

Then substitute in on both sides of the Bellman Equation.

Now we have the collocation equation, a regular but nonlinear equation *system*.

The collocation equation

Collocation equation

$$\Phi c = v(c),$$

Collocation matrix

$$\Phi_{ij} = \phi_j(k_i)$$

$$v_i(c) = \max_{k' \in K} \left\{ u(k_i, k') + \beta E \sum_{j=1}^{n} c_j \phi_j(, k_i k') \right\}$$

Conditional value function at a particular capital value k_i

2 ways to solve the collocation equation

$$c = \Phi^{-1} v(c)$$

Write in fixed point form, then iterate…

$$c_{s+1} = \Phi^{-1} v(c_s)$$

$$\Phi c - v(c) = 0,$$

$$c_{s+1} = c_s - (\Phi - v'(c_s))^{-1} (\Phi c_s - v(c_s))$$

Or, pose as a root finding problem, and update using Newton's method. Here v'() is the Jacobian of the value function at a particular value for k

NB this is a system, not just one equation. One eq for every node.

7. Heterogeneous agents

Why heterogeneous agent models

- Some RBC like models have borrowers and lenders, consumers and entrepreneurs.
- But may not be enough for some problems.
- Can’t study dynamics of income and wealth distribution, nor their impacts.
- Existence of representative agent rules out interesting problems like behaviour viz uninsurable idiosyncratic risk.

Heterogeneity step by step

- Start with no aggregate uncertainty, but uninsurable idiosyncratic risk.
- Basic method is to
  - take aggregate prices as given
  - solve agents decision problem using stochastic DP
  - Simulate individual C's to get distribution
  - Compute aggregate quantities
  - Check to see if market cleared at assumed prices; if not, find price that does clear market, then repeat.
- Move on to aggregate uncertainty.

Choices

- Checking market clearing.
- Solving individual agents' decision problem [we have seen some of these]
  - Finite element DP?
  - Continuous state DP via collocation?
- Computing the stationary distribution of asset holdings
  - Monte carlo simulation
  - Approximating distribution function….

History of methods[to be completed]

- Huggett (1993) Pure exchange economy; endowments, no production, no aggregate uncertainty. Idiosyncratic endowment risk.
- Aiyagari (1994)
- Krussell Smith (1998); aggregate uncertainty. Means of distributions are sufficient statistics.
- Early results on existence and uniqueness for simple cases. Generally not available for later, more realistic models.

Interesting recent papers

- Heathcote et al (2009); Guvenen. Surveys.
- Mackay and Reiss (2012). Countercyclical tax policy in sticky price het agent model.
- Reiter (2006) Computation of het agent model using function approximation to model the distribution.

A het agent model

$$\max E_0 \sum_{t=0}^{\infty} \beta^t u(c_t)$$

Agents maximise discounted stream of utility from consumption

$$u(c_t) = \frac{c_t^{1-\eta}}{1-\eta}, \eta > 0$$

$$w_t \text{ if } \epsilon = e$$

$$b_t \text{ if } \epsilon = u$$

Wages if employed, benefits if not.

$$\pi(\epsilon'|\epsilon) = prob\{\epsilon_{t+1} = \epsilon'|\epsilon_t = \epsilon\} = \begin{bmatrix} P_{u.u} & P_{ue} \\ P_{eu} & P_{ee} \end{bmatrix}$$

Employment status is exogenous, and Markov, with known transition law

$$
\begin{aligned}
a_{t+1} &= (1 + (1-\tau)r_t)a_t + (1-\tau)w_t - c_t, \epsilon = e \\
&= (1 + (1-\tau)r_t)a_t + b_t - c_t, \epsilon = u
\end{aligned}
$$

State-contingent budget constraint. Return on assets and wages taxed at rate tau.

$$
\frac{u'(c_t)}{\beta} = E_t[u'(c_{t+1})(1 + (1-\tau)r_{t+1})]
$$

Euler equation for consumption.

Firms and production

$$Y_t = F(K_t, N_t) = K_t^{\alpha} N_t^{1-\alpha}, \alpha \in (0.1)$$

Competitive firms owned by households, maximise profits, subject to this technology.

$$r_t = \alpha\left(\frac{N_t}{K_t}\right)^{1-\alpha} - \delta$$

$$w_t = (1-\alpha)\left(\frac{K_t}{N_t}\right)^{\alpha}$$

In equilibrium, factors are paid their marginal products

Government

$$B_t = T_t$$

For simplicity…
Governments balance total spending on benefits and total tax revenues from capital income and wages, each period.

$$T_t = \int_{a_{\min}} (\tau r_t K_t + \tau w_t N_t) da$$

$$B_t = \int_{a_{\min}} (b_t) da$$

Objects comprising staionary eqm

$$V(\epsilon, a), c(\epsilon, a), a'(\epsilon, a)$$

$$f(e, a), f(u, a)$$

$$w, r$$

$$K, N, T, B$$

Value and policy functions for agents
Time-invariant density functions for employment status
Constant factor prices – wages and interest rates
Constant capital, labour input, taxes and benefits

Such that…..

Aggregate quantities obtained by summing across agents…

$$K = \sum_{\epsilon \in \{e,u\}} \int_{a_{\min}}^{\infty} a f(\epsilon, a) da$$

$$N = \int_{a_{\min}}^{\infty} f(e, a) da$$

$$C = \sum_{\epsilon \in \{e,u\}} \int_{a_{\min}}^{\infty} c(\epsilon, a) f(\epsilon, a) da$$

$$T = \tau(wN + rK)$$

$$B = (1 - N)b$$

More conditions on the stationary equilibrium of the het agent economy

$$c(\epsilon, a), a'(\epsilon, a)$$

Policy functions solve household max problem

$$\begin{aligned} r &= \alpha\left(\frac{N}{K}\right)^{1-\alpha} - \delta \\ w &= (1-\alpha)\left(\frac{K}{N}\right)^{\alpha} \end{aligned}$$

Factors paid marginal products

$$T = B$$

Government budget balanced

$$F(a', \epsilon') = \sum_{\epsilon \in \{e,u\}} \pi(\epsilon'|\epsilon) F(a'^{-1}(a', \epsilon), \epsilon)$$

Distribution function is time-invariant, ie if take product of it with transition matrix, get back same distribution

Basic stylised steps to compute the stationary distribution

- Computation of individual policy functions, given aggregate q's and p's
  - This step we have already done, 2 different ways
- Computation of distribution given individual policy functions.
  - This is the new step that you haven't seen.

Steps for computing the stationary eq'm of the het agent model in more detail

1. Compute stationary employment N
2. Make initial guesses at K and τ
3. Compute the factor prices w and r
4. Compute household decision rules $c(\epsilon, a), a'(\epsilon, a)$
5. Compute $F(a', \epsilon')$ stationary distribution of assets for emp and unemp
6. Compute K and T that solve aggregate consistency
7. Compute τ that balances the budget
8. Update K and τ if necessary and go to 2.

1. Computing stationary employment

$$N_t = p_{ue}(1 - N_{t-1}) + p_{ee}N_{t-1}$$

Todays employment is yesterday's, times the chance that they stay in employment, plus yesterday's unemployed times the chance they leave unemployment.

Given any initial N_0 we can simply iterate on this equation to produce the stationary N

We can also compute stationary N analytically.

5. Computing stationary distribution

- Three methods
  - **Monte carlo simulation**
  - Function approximation
  - Discretisation of the density function

5. Computing stationary d'n using Monte Carlo simulation

1. Choose sample size N [not to be confused with N for employment]$=x * 10k$
2. Initialise asset holdings a_0^i and employment status ϵ_0^i.
3. Using the policy function already computed, compute $a^{i\prime}\ (\epsilon^i, a^i)$ for each of the N agents.
4. Use random number generator to draw $\epsilon^{i\prime}$ for each agent.
5. Compute summary moments of distribution of asset holdings, eg $\bar{a}, \sigma_a$
6. Iterate until moments converge.

Remarks

- No guarantee that convergent solution exists, or, if it does, that it is unique
- No guarantee that even if you have a unique stationary distribution, that this algorithm, even if correctly coded, will find it.
- Much trial and error needed!

Krussel-Smith

- Next logical step is to include aggregate uncertainty.
- Recall we had only idiosyncratic uncertainty.
- This is for next time.
- That paper contained methodological insight that agents could make do with just mean of moments of distributions.
- Contains within it the hint that heterogeneity doesn't matter.

Recapping on all the different methods

Recap: quasi-linear, perfect foresight method to solve the zero bound problem

- Guess when the nonlinearity stops binding, and the model is linear.
- Solve the linear model.
- Use that linear solution to solve ‘backwards’ for the initial period, when the model is ALSO linear, and with 1 less equation, and 1 fewer unknowns.
- Verify that the guess holds by checking the ‘max’ policy rule implies ZLB binding.

Recap: perfect-foresight, nonlinear Newton Raphson method

- RBC example.
- Solve for steady-state. Problem: to solve for trajectory given some initial value.
- Derive the FOCs; substitute out for C using the resource constraint.
- Formulate system of nonlinear equations, one for each unknown period.
- Solve using multidimensional NR.
- Linearise around a point, then find the root of the linear system. Use that root [provided it falls in the domain] as the next point around which to approximate.

Recap: Parameterised expectations algorithm

- RBC example.
- Derive Euler equation. Substitute in approximate function, involving shocks and states.
- Draw a large time series of shocks.
- Choose new parameters in forecast function to minimise gap between simulated ‘forecasts’ and what consumption actually turns out to be in the simulation.

Recap: Projection using Chebyshev Polynomials

- Another example of function approximation.
- Approximate at the Chebyshev nodes, gaining global accuracy for a given computational time.
- Approximate policy functions using CP's. Define residual function using the Euler Equation.
- Minimise the integral of this residual function, approximating that with a sum of residuals at the nodes.

Recap: deterministic dynamic programming

- Formulate the Bellman equation: recursive definition of the value function.
- Magic: starting from any degenerate guess, iterate on the Bellman equation to get the solution.
- Discretise the state [in this case, just capital] space.
- Essential when choices are discontinuous.

Recap: stochastic dynamic programming

- RBC: approximate technology shock with finite-state Markov chain.
- Now expectation of value function next period is probability weighted sum of different choices under the different outcomes for the shock.

Recap: using Chebyshev collocation to solve the Bellman equation

- Approximate the value function with CP.
- Alternative to discretising the capital/shock space.

Recap: Aiyagari

- Guess real interest rate that clears the capital market.
- At that real rate, solve the individual's problem using dynamic programming.
- Sum up individual capital demand, and see what interest rate clears the market.
- Repeat until convergence.

Recap: Krussel-Smith