Heterogeneous Agents in Macro Models

Solving Hetero-Models *With* Aggregate Uncertainty: Basics

Petr Sedláček

Lake Como School of Advanced Studies

July 2018

Plan overall

Basic tools for solving heterogeneous agent models

1. motivation
2. refresher: solution methods for rep agent models
3. Aiyagari model: hetero-model without aggregate uncertainty
4. Krusell-Smith algorithm: introducing aggregate uncertainty
5. alternative solution methods: XPA, hybrid, S-S
6. continuous time: basic techniques

Plan for today

1. refresher: solution methods for rep agent models
   - function approx. $+$ numerical integ. $\rightarrow$ projection
2. Krusell-Smith algorithm: introducing aggregate uncertainty

Perturbation refresher

Function approximation

Why function approximation?

- we are after policy rules
  - these are functions of state variables
  - moreover, closed form solutions rarely exist
  - $\rightarrow$ work with approximations of true functions
- let's think about this problem more generally first
- later we'll talk about how to implement it with DSGE models

Main idea of function approximation

Consider we want to approximate a function

$$y = f(x)$$

1. choose a family of functions to use as interpolants
   - popular choice is the family of polynomials
   - but others also exist
     - trigonometric functions (fourier approximation)
     - rational functions (pade approximation)

2. find coefficients of interpolant
   - such that interpolant and true function agree at certain points

Main idea of function approximation

We've already made enormous progress at this point:

- we have reduced the problem to a finite dimension!

$$y \approx \overline{f}(x) = a_0 T_0(x) + a_1 T_1(x) + ... + a_n T_n(x)$$

- where a_j are coefficients of the polynomial for $j = 0, .., n$
- and $T_j(x)$ are basis functions

a_j's solve the above at each chosen node, $i = 1, ..., n + 1$

$$y_i = f(x_i) = a_0 + \sum_{j=1}^{n} a_j T_j(x_i)$$

$$y = T(x)a$$

Main idea of function approximation

- the above (interpolation) method is a special case
- regression is a method of interpolation!
  - when the number of grid points i
  - is generally larger than the number of basis functions
  - what if you run a regression of n points on n regressors?

Why polynomial approximation?

Weierstrass' theorem (a sloppy version)

- if $f(x)$ is a continuous function in $[a, b]$
- then there exists a polynomial p for which

$$sup_{a \leq x \leq b} |f(x) - p(x)| \leq \epsilon$$

- with $\epsilon > 0$

In other words

- there exists a polynomial
- that approximates any continuous function
- arbitrarily well

Problem with Weierstrass...

- it is useless from a practical point of view
- because it gives no guidance on how to find p

There are (at least) 2 important choices to be made

- what type of polynomial to use
- and where to evaluate it (grid points)

An example is using monomials and equidistant nodes

- turns out to be a bad idea

Why not monomials?

- the choice of monomial basis functions implies

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & \cdots & x_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & \cdots & x_n^{n-1} \end{bmatrix} \begin{bmatrix} a_0 \\ \vdots \\ a_n \end{bmatrix}$$

- the first matrix on the RHS is a Vandermonde matrix
- even though it is non-singular, it is often ill-behaved
- intuition from regression?

Orthogonal polynomials

- monomials often suffer from high correlation
- orthogonal polynomials are constructed
- to have orthogonal basis functions (w.r.t. some measure)

$$\int_a^b T_i(x)T_j(x)w(x) = 0 \quad \forall i,j \quad i \neq j$$

  - $w(x)$ is some weighting function

Popular orthogonal polynomials are Chebyshev polynomials

Chebyshev polynomials

- defined on interval $[-1, 1]$
- weighting function

$$w(x) = \frac{1}{(1 - x^2)^{1/2}}$$

- basis functions

$$T_0(x) = 1$$

$$T_1(x) = x$$

$$T_{j+1}(x) = 2xT_j(x) - T_{j-1}(x) \quad j > 1$$

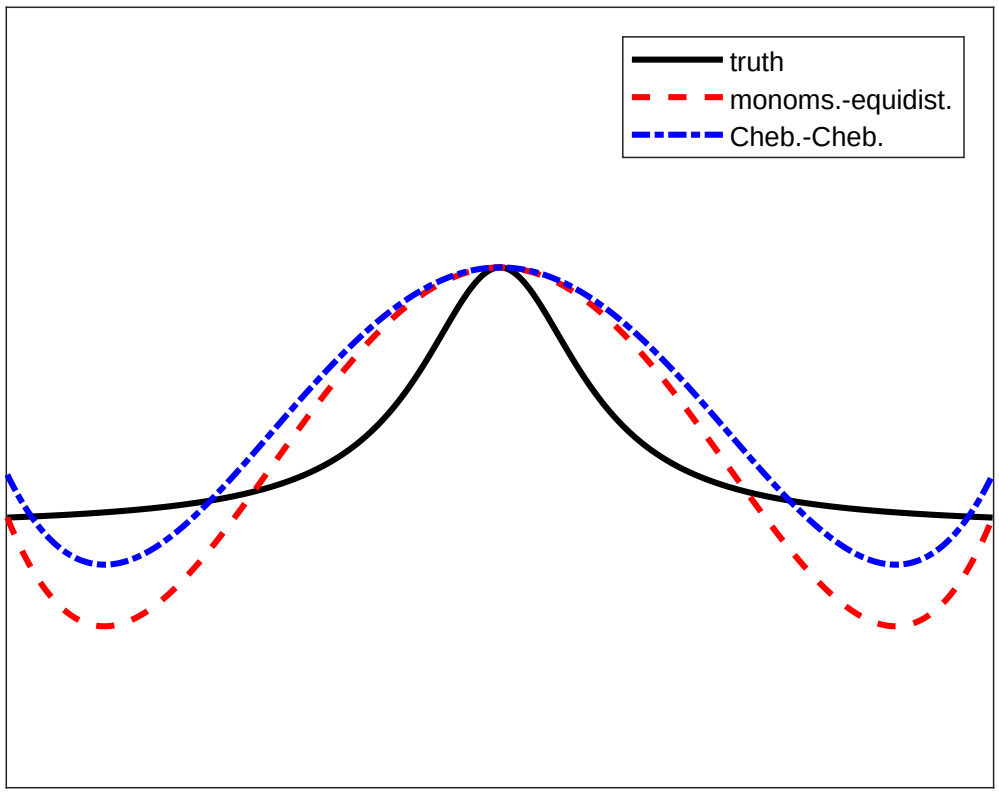
Why not an equidistant grid?

- suppose we have an equidistant grid
- it turns out that the higher the order of polynomial
- the larger the "swings" between grid points
- these oscillations become more dramatic at the end points!

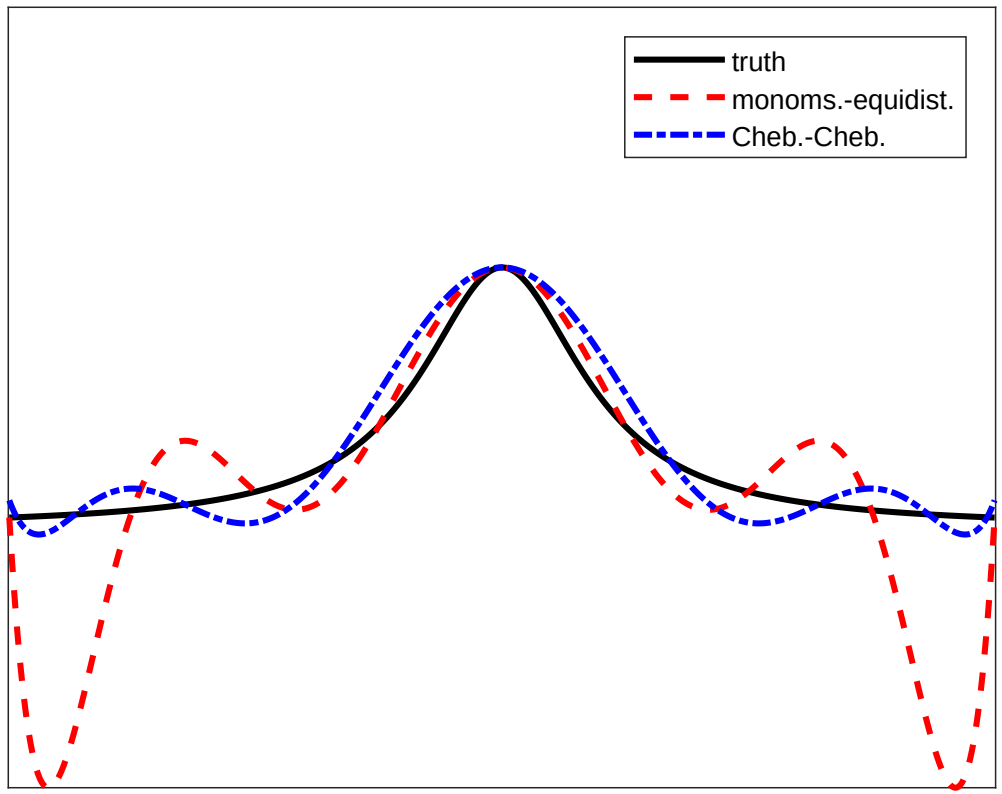
Weierstraas' theorem

- there exists a uniformly converging polynomial approximation
- to find it, however, we have to be smart about the nodes

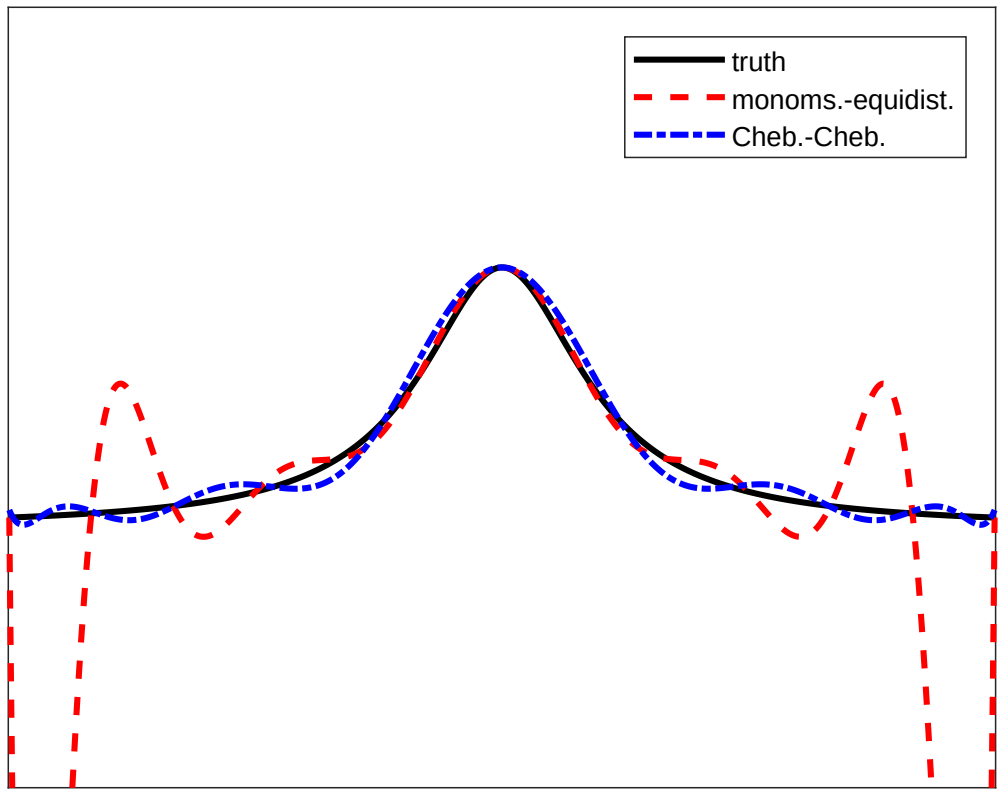
Why not an equidistant grid?



Why not an equidistant grid?



Why not an equidistant grid?



Chebyshev nodes

Chebyshev nodes ensure uniform convergence

- the roots (z_j) at which the basis functions are equal to 0
- e.g. $T_2(x) = 2x^2 - 1 \rightarrow$ nodes of $z_1 = -1/2$ and $z_2 = 1/2$
- i.e. get n Chebyshev nodes by solving the nth basis function
- this is the reason for the popularity of Chebyshev polynomials
- Chebyshev nodes can be computed according to

$$z_j = -cos\left(\frac{(2j-1)\pi}{2n}\right)$$

Interval conversion

- to approximate on an interval $[a, b]$
- we must rescale the Chebyshev nodes
- find α and β for which

$$x = \alpha z + \beta$$

$$\beta - \alpha = a$$

$$\beta + \alpha = b$$

- then if $z \in [-1, 1]$ and $x \in [a, b]$ then

$$x = \frac{b-a}{2} z + \frac{a+b}{2}$$

A popular alternative to polynomials - splines

- polynomials approximate over entire domain
  - spectral method
- splines split support into sections
  - finite element method
- splines can be expressed as linear combinations of basis fces
- but they are not polynomials
  - basis functions are zero for most of the domain

Piece-wise linear splines

- the easiest type is piece-wise linear

$$f(x) \approx \left(1 - \frac{x - x_i}{x_{i+1} - x_i}\right) f_i + \left(\frac{x - x_i}{x_{i+1} - x_i}\right) f_{i+1} \quad x \in [x_i, x_{i+1}]$$

- in general not differentiable at nodes
- could be problematic $\rightarrow$ use higher-order polynomials

Cubic splines

- fit a 3-rd order polynomial in each segment

$$f(x) \approx a_i + b_i x + c_i x^2 + d_i x^3 \quad x \in [x_i, x_{i+1}]$$

$$S(x) = \begin{cases} S_1(x) & x_0 \leq x \leq x_1 \\ S_i(x) & x_{i-1} \leq x \leq x_i \\ S_n(x) & x_{n-1} \leq x \leq x_n \end{cases}$$

- i.e. we have n separate cubic splines for $n+1$ nodes

Cubic splines

- n splines give $4n$ coefficients to determine
- what conditions pin down the $4n$ coefficients?
  - $2 + 2(n-1)$ function values at the nodes
  - $2(n-1)$ smoothness conditions (for $0 < i < n$)

$$S_i'(x_i) = S_{i+1}'(x_i)$$

$$S_i''(x_i) = S_{i+1}''(x_i)$$

  - 2 boundary conditions
    - "natural (simple)" $S_1''(x_0) = 0$ and $S_n''(x_n) = 0$
    - or "clamped" $S_1'(x_0) = 0$ and $S_n'(x_n) = 0$

Before we move on...

- sovling DSGE models means getting policy functions
- so how does function approximation work in DSGE models?

Perturbation refresher
Numerical integration

Why numerical integration?

- in economics, there are plenty of integrals
  - expectations
- evaluating integrals can be a tough problem
  - the functional form may be nasty
  - we may not even have the functional form
  - we may be able to evaluate it, but not draw from it
    - as in e.g. Bayesian estimation
- therefore, we need a way around this...

Intuitive integration method

Consider you want to compute the integral $\int_a^b h(x)dF(x)$

- where x is a random variable with CDF $F(x)$

Monte-Carlo integration uses the following approximation

$$\int_a^b h(x)dF(x) \approx \frac{\sum_{t=1}^{T} h(x_t)}{T}$$

- $\{x_t\}_{t=1}^{T}$ is a series drawn from a random number generator

This procedure is very simple and intuitive but

- it is not very accurate (fast)
- more powerful procedures available $\rightarrow$ numerical integration

Main idea of numerical integration

- we want to calculate

$$I = \int_a^b f(x)dx$$

- the basic idea is to approximate it with

$$I \approx \sum_{i=1}^{n} \omega_i f(x_i)$$

Main idea of numerical integration

$$I \approx \sum_{i=1}^{n} \omega_i f(x_i)$$

- therefore, we face (at least) three choices
  - choice of quadrature weights, ω_i's
  - choice of quadrature nodes, x_i's
  - choice of number of evaluations, n

Types of quadrature methods

Newton-Cotes quadrature methods

- break interval into equidistant intervals
- approximate $f(x)$ with a low-order polynomial
- use integrals of polynomials as the approximations

Gaussain quadrature methods

- same idea as with Newton-Cotes
- more clever in choosing quadrature nodes

Gaussian quadrature

- Newton-Cotes formulas are simple due to equidistant nodes
- moreover, one can show that with Newton-Cotes
  - we get the exact answer when the true function
  - is a polynomial of order $n-1$
- but we can get more accuracy by choosing nodes cleverly
  - we get the exact answer if the true function
  - is a polynomial of order $2n-1$!
  - i.e. 5 nodes give exact (accurate) answers for true functions
  - which are (approximated by) a 9th order polynomial!

Procedure of Gaussian quadrature

Using n nodes, we can approximate $\int_{-1}^{1} f(x)dx$ as

$$\int_{-1}^{1} f(x)dx \approx \sum_{i=1}^{n} \omega_i f(\zeta_i)$$

- i.e. we have $2n$ parameters
- we need $2n$ conditions to pin these parameters down
- $\rightarrow$ make sure we get correct answer for basis functions

$$1, x, x^2, \cdots, x^{2n-1}$$

- $\rightarrow$ get correct answer for *any* polynomial of order $2n-1$!

Procedure of Gaussian quadrature

How to choose weights and nodes?

- to get the basis functions
- we must solve for ω_i and $\zeta_i \quad \forall i$ s.t.

$$\int_{-1}^{1} x^j dx = \sum_{i=1}^{n} \omega_i \zeta_i^j \quad j = 0, 1, \cdots, 2n-1$$

- i.e. solve a system of $2n$ equations in $2n$ unknowns
- note that the solution is independent of f!

Gauss-Hermite quadrature

- when the true function is given by $f(x) = g(x)W(x)$ where
  - $g(x)$ can be approximated well by a polynomial
  - but $f(x)$ cannot
- then adjust the quadrature procedure depending on $W(x)$
- Gauss-Hermite quadrature is used when $W(x) = \exp(-x^2)$
- why is this an interesting case?

Gauss-Hermite quadrature

- nodes and weights chosen s.t.

$$\int_{-\infty}^{\infty} x^j \exp(-x^2)dx = \sum_{i=1}^{n} \omega_i \zeta_i^j \quad j = 0, 1, \cdots, 2n-1$$

- the approximation is then given by

$$\int_{-\infty}^{\infty} g(x) \exp(-x^2)dx \approx \sum_{i=1}^{n} \omega_i^{GH} g(\zeta_i^{GH})$$

Change of variable

- suppose we want to calculate

$$\int_{-\infty}^{\infty} \frac{g(y)}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right) dy$$

- how to use Gauss-Hermite quadrature?
  - define $x = \frac{y-\mu}{\sigma\sqrt{2}}$ or $y = x\sigma\sqrt{2} + \mu$
  - this gives

$$\int_{-\infty}^{\infty} \frac{g(x\sigma\sqrt{2}+\mu)}{\sigma\sqrt{2\pi}} \exp(-x^2)\sigma\sqrt{2}dx$$

  - then use Gauss-Hermite quadrature

$$\int_{-\infty}^{\infty} \frac{g(y)}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right) dy \approx \sum_{i=1}^{n} \omega_i^{GH} \frac{h(\zeta_i^{GH}\sigma\sqrt{2}+\mu)}{\sqrt{\pi}}$$

Change of variable

- could you instead simply do the following?
  - $\overline{g}(y) = \frac{g(y)}{\sigma\sqrt{2\pi}} \frac{\exp\left(-\frac{(y-\mu)^2}{2\sigma^2}\right)}{\exp(-y^2)}$
  - and approximate $\int_{-\infty}^{\infty} \overline{g}(y) \exp(-y^2)$?

Perturbation refresher

Projection in a nutshell

Projection: main idea

Goal is to solve for policy rules of a given model

$$
\begin{aligned}
c_t^{-\gamma} &= \mathbb{E}_t[\beta c_{t+1}^{-\gamma} \alpha Z_{t+1} k_{t+1}^{\alpha-1}] \\
c_t + k_{t+1} &= Z_t k_t^{\alpha} \\
\ln(Z_{t+1}) &= \rho \ln(Z_t) + \epsilon_{t+1} \\
\epsilon_t &\sim N(0, \sigma^2) \\
k_0, Z_0 &\quad \text{given}
\end{aligned}
$$

Projection: main idea

Policy rules are

- (unknown) functions of state variables
- $\rightarrow$ use function approximation and numerical integration

True rational expectations solution given by:

$$c_t = c(k_t, Z_t)$$

$$k_{t+1} = k(k_t, Z_t)$$

Approximate $c(k_t, Z_t)$ with polynomial $P_n(k_t, Z_t; \psi_n)$

- what about $k(k_t, Z_t)$?

Projection: main idea

- what are we solving for?
- how do we do it?

Define error terms

$$e(k_t, Z_t) = -c_t^{-\gamma} + \mathbb{E}_t[\beta c_{t+1}^{-\gamma} \alpha Z_{t+1} k_{t+1}^{\alpha-1}]$$

- substitute c_t with $P_n(k_t, Z_t; \psi_n)$
- there are N_n elements of ψ_n but only one Euler equation...

Details of the setup

Define M grid points $\{k_i, Z_i\}$

$$e(k_i, Z_i; \psi_n) = -P_n(k_i, Z_i; \psi_n)^{-\gamma} + \alpha\beta\mathbb{E}\left[\begin{array}{c} P_n(k', Z'; \psi_n)^{-\gamma} \times \\ Z' \times \\ (k')^{\alpha-1} \end{array}\right]$$

- but what about k' and Z'?

Details of the setup

$$e(k_i, Z_i; \psi_n) = -P_n(k_i, Z_i; \psi_n)^{-\gamma} + \mathbb{E}\left[\begin{array}{c} \alpha\beta \times \\ P_n\left(Z_i k_i^{\alpha} - P_n(k_i, Z_i; \psi_n), \exp(\rho \ln(Z_i) + \epsilon'); \psi_n\right)^{-\gamma} \times \\ \exp(\rho \ln(Z_i) + \epsilon') \times \\ (Z_i k_i^{\alpha} - P_n(k_i, Z_i; \psi_n))^{\alpha - 1} \end{array}\right]$$

- but what about ϵ'?

Details of the setup

$$e(k_i, Z_i; \psi_n) = -P_n(k_i, Z_i; \psi_n)^{-\gamma} +$$

$$\sum_{j=1}^{J} \frac{\omega_j}{\sqrt{\pi}} \left[\begin{array}{c} \alpha\beta \times \\ P_n\left(Z_i k_i^{\alpha} - P_n(k_i, Z_i; \psi_n), \exp(\rho \ln(Z_i) + \sqrt{2}\sigma\zeta_j); \psi_n\right)^{-\gamma} \times \\ \exp(\rho \ln(Z_i) + \sqrt{2}\sigma\zeta_j) \times \\ (Z_i k_i^{\alpha} - P_n(k_i, Z_i; \psi_n))^{\alpha - 1} \end{array} \right]$$

- ω_j and ζ_j are Gauss-Hermite quadrature weights and nodes

How to solve for coefficients of P_n?

- equation solver/minimization routine
- iteration procedures
  - fixed-point iteration
  - time-iteration

How to solve for coefficients of P_n?

How to choose polynomial and grid points?

- use Chebyshev nodes
  - guaranteed uniform convergence
- use Chebyshev polynomials
  - especially useful for iteration procedures
  - rescaling is needed (defined only between -1 and 1)

Solvers and minimization routines

Smart in updating ψ_n, but high-dimensions costly

- Collocation: $M = N_n$
  - use equation solver to obtain ψ_n at which $e(k_i, Z_i; \psi_n) = 0 \quad \forall i$
- Galerkin: $M > N_n$
  - use minimization routine to obtain ψ_n
  - minimize $e(k_i, Z_i; \psi_n)$

Iteration methods

Can deal with high N_n, sometimes guaranteed to converge

- in both fixed-point and time-iteration
  - 1. use latest "guess" of coefficients ψ_n in Eurler equation
  - and compute implied consumption values c_i
  - 2. use c_i values from 1 to get new guess of ψ_n
  - 3. update your guess of coefficients ψ_n
- difference between fixed-point and time-iteration
- is in emplementation of 1

Fixed-point iteration

Define ψ_n^q as value of ψ_n in qth iteration

1. At each grid point calculate c_i using ψ_n^{q-1}

$$c_i^{-\gamma} =$$

$$\sum_{j=1}^{J} \frac{\omega_j}{\sqrt{2}} \left[\begin{array}{c} \alpha\beta\times \\ P_n\left(Z_i k_i^\alpha - P_n(k_i, Z_i; \psi_n^{q-1}), \exp(\rho \ln(Z_i) + \sqrt{2}\sigma\zeta_j); \psi_n^{q-1}\right)^{-\gamma} \times \\ \exp(\rho \ln(Z_i) + \sqrt{2}\sigma\zeta_j)\times \\ \left(Z_i k_i^\alpha - P_n(k_i, Z_i; \psi_n^{q-1})\right)^{\alpha-1} \end{array} \right]$$

Fixed-point iteration

2. Use obtained c_i's to get new guess of ψ_n
   - e.g. for $n = 2$, define

$$X = \begin{bmatrix} 1 & k_1 & Z_1 & k_1^2 & k_1 Z_1 & Z_1^2 \\ 1 & k_2 & Z_2 & k_2^2 & k_2 Z_2 & Z_2^2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & k_M & Z_M & k_M^2 & k_M Z_M & Z_M^2 \end{bmatrix}$$

   - compute $\hat{\psi}_n^q = (X'X)^{-1}X'Y$, where
   - $Y = (c_1, c_2, ..., c_M)$ from step 1

Fixed-point iteration

3. Use past guess and newly estimated values for ψ_n for new guess
   - typically making slower steps is more stable:

$$\psi_n^q = \lambda \psi_n^{q-1} + (1-\lambda)\hat{\psi}_n^q$$

   - $\lambda \in [0, 1)$
     - high values of λ increase chances of convergence
     - but they also slow things down

Time-iteration

Basic idea is the same as with fixed-point iteration

- 1. use latest "guess" of coefficients ψ_n in Eurler equation
- and compute implied consumption values c_i
- 2. use c_i values from 1 to get new guess of ψ_n
- 3. update your guess of coefficients ψ_n

But this time use latest guess of ψ_n only

- for next period's choices
  - makes the solution of c_i trickier
  - guarantees convergence (under conditions similar to VFI)

Time-iteration

There is something slightly inconsistent with fixed-point iteration:

$$c_i^{-\gamma} =$$

$$\sum_{j=1}^{J} \frac{\omega_j}{\sqrt{2}} \left[\begin{array}{c} \alpha\beta\times \\ P_n\left(Z_i k_i^{\alpha} - P_n(k_i, Z_i; \psi_n^{q-1}), \exp(\rho \ln(Z_i) + \sqrt{2}\sigma\zeta_j); \psi_n^{q-1}\right)^{-\gamma} \times \\ \exp(\rho \ln(Z_i) + \sqrt{2}\sigma\zeta_j)\times \\ (Z_i k_i^{\alpha} - P_n(k_i, Z_i; \psi_n^{q-1}))^{\alpha-1} \end{array} \right]$$

Time-iteration

Time iteration uses ψ_n^{q-1} *only for next period's choices*!

$$c_i^{-\gamma} =$$

$$\sum_{j=1}^{J} \frac{\omega_j}{\sqrt{2}} \left[\begin{array}{c} \alpha\beta\times \\ P_n\left(Z_i k_i^{\alpha} - c_i, \exp(\rho \ln(Z_i) + \sqrt{2}\sigma\zeta_j); \psi_n^{q-1}\right)^{-\gamma} \times \\ \exp(\rho \ln(Z_i) + \sqrt{2}\sigma\zeta_j)\times \\ (Z_i k_i^{\alpha} - c_i)^{\alpha-1} \end{array} \right]$$

Interim summary

Pros of perturbation:

- easy to implement (Dynare)
- can handle large state-space (heterogeneity)

Cons of perturbation:

- "local" solution method
  - potentially an issue with low-order approximations
- can't handle certain features (non-differentiabilities)

Interim summary

Pros of projection:

- "global" solution method
- can handle non-differentiabilities

Cons of projection:

- curse of dimensionality
- defining the grid is sometimes tricky
- sometimes not all calculations well-defined in state-space

Introducing aggregate uncertainty into Aiyagari model

Extend Aiyagari model

Assume presence of aggregate uncertainty

- enters production function of representative firm

$$Y_t = Z_t K_t^{\alpha} L_t^{1-\alpha}$$

- otherwise there is no difference from model before

New individual problem

The individual problem is now

$$\max_{\{c_{i,t},k_{i,t+1}\}_{t=0}^{\infty}} \mathbb{E}_t \sum_{t=0}^{\infty} \beta^t \ln(c_{i,t})$$

s.t.

$$c_{i,t} + k_{i,t+1} = r_t k_{i,t} + w_t \epsilon_{i,t} + (1-\delta)k_{i,t}$$

$$k_{i,t+1} \geq 0$$

- how is this different from before?!

New individual problem

Prices are not fixed!

- $r_t = Z_t \alpha K_t^{\alpha-1}$ and $w_t = Z_t(1-\alpha)K_t^{\alpha}$

Moreover, what do agents really care about?

- not just r_t and w_t
- but also all future values of r and w!

What do agents care about?

The need to forecast prices necessitates

- the need for information for forecasting K
- in general, we need to forecast the joint distribution
  - of capital holdings and idiosyncratic productivity levels
- now this is really a tough problem
  - the distribution is infinite-dimensional!

Before moving on

The above discussion makes clear

- that partial equilibrium is much easier
- even with aggregate uncertainty

The individual's problem is easy to solve if

- you know the path of r_t and w_t!
- the above hints at how to solve the GE model

Krusell-Smith algorithm

Main idea of Krusell-Smith algorithm

Principle is similar to that in Aiyagari model

- guess (evolution of) prices
- individual problem solved for given (evolution of) prices
- simulate economy, aggregate, check implied price paths

The difficulty is that to forecast K

- we really need to forecast the entire joint distribution
  - of capital holdings and idiosyncratic productivity

Forecasting the joint distribution

Let $\mathbb{F}_t$ be the beginning-of-period joint distribution

- of capital holdings and idiosyncratic productivity

Its evolution can then be described as

$$\mathbb{F}_{t+1} = \mu(Z_t, \mathbb{F}_t)$$

- given today's joint distribution
- and today's aggregate shock
- can forecast tomorrows joint distribution
  - recall that idiosyncratic shocks are exogenous

Forecasting the joint distribution

However, this is still a crazy difficult problem!

- distribution is infite-dimensional

Krusell and Smith (1998) propose to instead

- approximate the distribution by
- focusing on a limited set of characteristics

Krusell-Smith algorithm

Instead of figuring out $\mu(.,.)$

- assume an approximating "aggregate law of motion"

$$m_{t+1} = \overline{\mu}(Z_t, m_t; \psi_{\overline{\mu}})$$

- m_t is a set of moments of the joint distribution
  - you need to make a stand on m_t beforehand
  - we'll discuss such choices later

Krusell-Smith algorithm

1. guess a value for $\psi_{\overline{\mu}}$
   - implies values for K_t^D and thus r_t and w_t
2. solve individual problem with given aggregate law of motion
3. simulate economy and calculate moments of joint distribution
4. estimate $\hat{\psi}_{\overline{\mu}}$ implied by simulation
5. compare to previous guess
   - if $\hat{\psi}_{\overline{\mu}} = \psi_{\overline{\mu}} \rightarrow$ stop
   - if $\hat{\psi}_{\overline{\mu}} \neq \psi_{\overline{\mu}} \rightarrow$ update and go to 2

What to keep in mind

Algorithm based on idea of "approximate aggregation"

- evolution of prices described well using
  - exogenous shocks
  - a limited number of moments of current distribution
- does not mean that
  - behavior aggregates to rep-agent model
  - or that individual variables behave as aggregates!

Summary

What did we do?

Basic tools for solving heterogeneous agent models

1. motivation
2. quick refresher on perturbation and projection
3. the Aiyagari model
4. introducing aggregate uncertainty
5. Krusell-Smith algorithm

What's next?

Key issues of heterogeneous agent models

- existence/uniqueness of equilibria?

Practical issues with associated solution methods

- choices to be made
- stopping rules
- simulation methods
- accuracy tests

An alternative solution method

- hybrid methods, XPA, S-S

Continuous time