PROJECTION METHODS FOR DYNAMIC MODELS

Kenneth L. Judd

Hoover Institution and NBER

June 28, 2006

Functional Problems

- Many problems involve solving for some unknown function
  - Dynamic programming
  - Consumption and investment policy functions
  - Pricing functions in asset pricing models
  - Strategies in dynamic games
- The projection method is a robust method for solving such problems

An Ordinary Differential Equation Example

- Consider the differential equation

$$y' - y = 0, \quad y(0) = 1, \quad 0 \leq x \leq 3. \tag{11.1.1}$$

- Define L

$$Ly \equiv y' - y \ . \tag{11.1.2}$$

  - L is an operator mapping functions to functions; domain is C^1 functions and range is C^0.
  - Define $Y = \{y(x) | y \in C^1, \quad y(0) = 1\}$
  - (11.1.1) wants to find a $y \in Y$ such that $Ly = 0$.

- Approximate functions: consider family

$$\hat{y}(x; a) = 1 + \sum_{j=1}^{n} a_j x^j. \tag{11.1.3}$$

  - An affine subset of the vector space of polynomials.
  - Note that $\hat{y}(0; a) = 1$ for any choice of a, so $\hat{y}(0; a) \in Y$ for any a.

- Objective: find a s.t. $\hat{y}(x; a)$ “nearly” solves differential equation (11.1.1).

- Define *residual function*

$$R(x;a) \equiv L\hat{y} = -1 + \sum_{j=1}^{n} a_j (j x^{j-1} - x^j) \tag{11.1.4}$$

  - $R(x;a)$ is deviation of $L\hat{y}$ from zero, the target value.
  - A projection method adjusts a until it finds a "good" a that makes $R(x;a)$ "nearly" the zero function.
  - Different projection methods use different notions of "good" and "nearly."

- Consider

$$\hat{y}(x;a) = 1 + \sum_{j=1}^{3} a_j x^j$$

- Least Squares:

  - Find a that minimizes the total squared residual

$$\min_a \int_0^3 R(x;a)^2 \, dx. \tag{11.1.5}$$

- Method of moments:
  - Idea: If $R(x;a)$ were zero, then $\int_0^3 R(x;a)\, f(x)\, dx = 0$ for all $f(x)$.
  - Use low powers of x to identify a via projection conditions

$$0 = \int_0^3 R(x;a)\, x^j\, dx\ , \quad j = 0, 1, 2. \tag{11.1.9}$$

- Galerkin
  - Idea: use basis elements, x, x^2, and x^3 in projection conditions
  - Form projections of R against the basis elements

$$0 = \int_0^3 R(x;a)\, x^j\, dx\ , \quad j = 1, 2, 3.$$

- Collocation
  - Idea: If $R(x;a) = 0$ then it is zero at all x.
  - Specify a finite set of X and choose a so that $R(x;a)$ is zero $x \in X$. If $X = \{0, 3/2, 3\}$, the uniform grid, this reduces to linear equations

- Chebyshev Collocation
  - Idea: interpolation at Chebyshev points is best
  - List the zeroes of $T_3(x)$ adapted to [0,3]

$$X = \left\{ \frac{3}{2}\,(\cos\frac{\pi}{6} + 1), \frac{3}{2}, \frac{3}{2}\,(\cos\frac{5\pi}{6} + 1) \right\}$$

- Solutions

Table 11.1: Solutions for Coefficients in (11.1.3)

Scheme:	a_1	a_2	a_3
Least Squares	1.290	-.806	.659
Galerkin	2.286	-1.429	.952
Chebyshev Collocation	1.692	-1.231	.821
Uniform Collocation	1.000	-1.000	.667
Optimal L_2	1.754	-.838	.779

Table 11.2: Projection Methods Applied to (11.1.2): L_2 errors of solutions

	Uniform	Chebyshev	Least		
n	Collocation	Collocation	Squares	Galerkin	Best poly.
3	5.3(0)	2.2(0)	3.2(0)	5.3(-1)	1.7(-1)
4	1.3(0)	2.9(-1)	1.5(-1)	3.6(-2)	2.4(-2)
5	1.5(-1)	2.5(-2)	4.9(-3)	4.1(-3)	2.9(-3)
6	2.0(-2)	1.9(-3)	4.2(-4)	4.2(-4)	3.0(-4)
7	2.2(-3)	1.4(-4)	3.8(-5)	3.9(-5)	2.8(-5)
8	2.4(-4)	9.9(-6)	3.2(-6)	3.2(-6)	2.3(-6)
9	2.2(-5)	6.6(-7)	2.3(-7)	2.4(-7)	1.7(-7)
10	2.1(-6)	4.0(-8)	1.6(-8)	1.6(-8)	1.2(-8)

Continuous-Time Life-Cycle Consumption Models

- Consider life-cycle problem

$$\begin{aligned} &\max_c \int_0^T e^{-\rho t} u(c)\, dt, \\ \dot{A} &= rA + w(t) - c(t) \\ A(0) &= A(T) = 0 \end{aligned} \tag{10.6.10}$$

- Parameters
  - $u(c) = c^{1+\gamma}/(1+\gamma)$
  - $\rho = 0.05, r = 0.10, \gamma = -2$
  - $w(t) = 0.5 + t/10 - 4(t/50)^2$, and $T = 50$.

- The functions $c(t)$ and $A(t)$ must approximately solve the two point BVP

$$\begin{aligned} \dot{c}(t) &= -\tfrac{1}{2}c(t)(0.05 - 0.10), \\ \dot{A}(t) &= 0.1A(t) + w(t) - c(t), \\ A(0) &= A(T) = 0. \end{aligned} \tag{11.4.7}$$

- Approximation: degree 10 Chebyshev polys for $c(t)$ and $A(T)$:

$$\begin{aligned} A(t) &= \sum_{i=0}^{10} a_i T_i\left(\tfrac{t-25}{25}\right), \\ c(t) &= \sum_{i=0}^{10} c_i T_i\left(\tfrac{t-25}{25}\right), \end{aligned} \tag{11.4.6}$$

- Define the two residual functions

$$\begin{aligned} R_1(t) &= \dot{c}(t) - 0.025c(t) \\ R_2(t) &= \dot{A}(t) - \left(.1A(t) + \left(.5 + \tfrac{t}{10} - 4(\tfrac{t}{50})^2\right) - c(t)\right). \end{aligned} \tag{11.4.8}$$

- Choose a_i and c_i so that $R_1(t)$ and $R_2(t)$ are nearly zero and $A(0) = A(T) = 0$ hold.
  - Boundary conditions impose two conditions
  - Need 20 more conditions to determine the 22 unknown coefficients.
  - Use 10 collocation points on $[0, 50]$: the 10 zeros of $T_{10}(t - 25/25)$

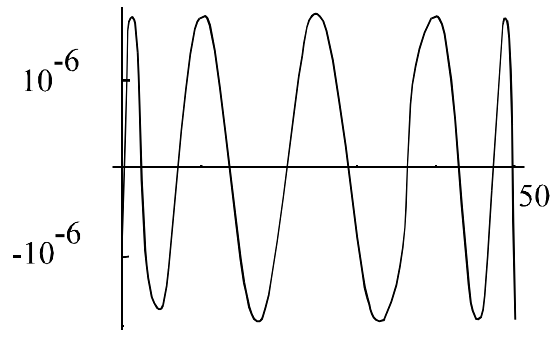
$$\mathcal{C} \equiv \{0.31, 2.72, 7.32, 13.65, 21.09, 28.91, 36.35, 42.68, 47.28, 49.69\}$$

  - Choose the a_i and c_i coefficients, which solve

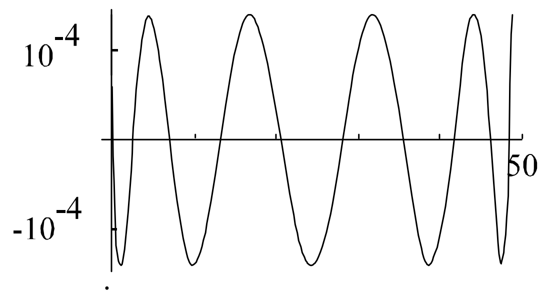
$$\begin{aligned} R_1(t_i) &= 0,\ \ t_i \in \mathcal{C}, i = 1, ..., 10, \\ R_2(t_i) &= 0,\ \ t_i \in \mathcal{C}, i = 1, ..., 10, \\ A(0) &= \textstyle\sum_{i=1}^{10} a_i(-1)^i = 0, \\ A(50) &= \textstyle\sum_{i=1}^{10} a_i = 0. \end{aligned} \tag{11.4.9}$$

  - 22 linear equations in 22 unknowns.
  - The system is nonsingular; therefore there is a unique solution.
- The true solution to the system (11.4.7) can be solved since it is a linear problem.

- Residuals:

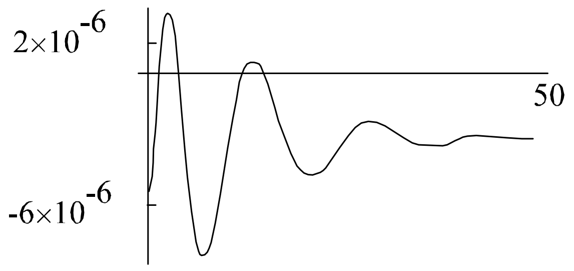


$\dot{c}$ equation residuals

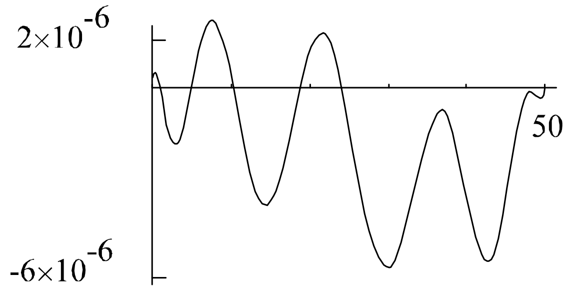


$\dot{A}$ equation residuals

- Errors



relative consumption errors



relative asset errors

- Note: Errors are roughly same size as residuals

Continuous-Time Growth Model

- Consider

$$\max_{c} \int_0^\infty e^{-\rho t}\, u(c)\, dt$$
$$\dot{k} = f(k) - c$$

- Optimal policy function, $C(k)$, satisfies the ODE

$$0 = C'(k)\ (f(k) - C(k)) - \frac{u'(C(k))}{u''(C(k))}\ (\rho - f'(k)) \equiv \mathcal{N}(C)$$
$$\mathcal{N} : C^1 \to C^0$$

together with the boundary condition that $C(k^*) = f(k^*),\ f'(k^*) = \rho$

- Example:
  - $f(k) = \rho k^\alpha/\alpha$, $u(c) = c^{1+\gamma}/(1+\gamma)$
  - $\rho = 0.04$, $\alpha = 0.25$, $\gamma = -2$
  - $k^* = 1$.
- Use Chebyshev polynomials for $k \in [0.25, 1.75]$,

$$\hat{C}(k;a) \equiv \sum_{i=0}^{n} a_i\, T_i\left(\frac{k-1}{0.75}\right)$$

- Define residual

$$\begin{aligned} 0 &= R(k;a) = \mathcal{N}(\hat{C}(\cdot;a))(k) \\ &= \hat{C}'(k)\,\left(f(k) - \hat{C}(k)\right) - \frac{u'(\hat{C}(k))}{u''(\hat{C}(k))}\,(\rho - f'(k)) \end{aligned}$$

- Collocation: compute a by solving

$$R\left(k_i\ ;\ a\right)=0, \quad i=1,\cdots,n+1,$$

where the k_i are the $n+1$ zeroes of $T_{n+1}\left(\frac{k-1}{0.75}\right)$.

- Results: $\hat{E}^n(k)$ is error of degree n approximation

Table 11.3: Projection Methods Applied to (5.1)

k	$\hat{E}^2(k)$	$\hat{E}^5(k)$	$\hat{E}^8(k)$	$\hat{E}^{12}(k)$	$\hat{C}^{12}(k)$
.6	-9(-3)	-2(-3)	4(-6)	-9(-9)	0.159638
.8	-2(-2)	-2(-4)	-2(-6)	-1(-8)	0.180922
1.0	5(-16)	-2(-4)	-5(-16)	5(-16)	0.200000
1.2	1(-2)	1(-4)	1(-6)	7(-9)	0.217543
1.4	4(-3)	-9(-5)	-2(-6)	7(-9)	0.233941

Simple Example: One-Sector Growth

- Consider

$$\max_{c_t} \sum_{t=1}^{\infty} \beta^t u(c_t)$$
$$k_{t+1} = f(k_t) - c_t$$

- Optimality implies that c_t satisfies

$$u'(c_t) = \beta u'(c_{t+1}) f'(k_{t+1})$$

- Problem: The number of unknowns $c_t, \; t = 1, 2, ...$ is infinite.
- **Step 0:** Express solution in terms of an unknown function

$$c_t = C(k_t) : \text{consumption function}$$

  - Consumption function $C(k)$ must satisfy the functional equation:

$$\begin{aligned} 0 &= u'(C(k)) - \beta u'(C(f(k) - C(k))) f'(f(k) - C(k)) \\ &\equiv (\mathcal{N}(C))(k) \end{aligned}$$

  - This defines the operator

$$\mathcal{N} : C^0_+ \to C^0_+$$

  - Equilibrium solves the operator equation

$$0 = \mathcal{N}(C)$$

- **Step 1:** Create approximation:
  - Find
$$\widehat{C} \equiv \sum_{i=0}^{n} a_i k^i$$
    which "nearly" solves
$$\mathcal{N}\ (\widehat{C}) = 0$$
  - Convert an infinite-dimensional problem to a finite-dimensional problem in R^n
    * No discretization of state space
    * A form of discretization, but in spectral domain
- **Step 2:** Compute Euler equation error function:
$$R\ (k;\ \vec{a}) = u'(\widehat{C}(k)) - \beta u'(\widehat{C}(f(k) - \widehat{C}(k)))f'(f(k) - \widehat{C}(k))$$

- **Step 3**: Choose $\vec{a}$ to make $R(\cdot;\vec{a})$ "small" in some sense**:**
  - Least-Squares: minimize sum of squared Euler equation errors
$$\min_{\vec{a}} \int R(\cdot;\vec{a})^2 dk$$
  - Galerkin: zero out weighted averages of Euler equation errors
$$P_i(\vec{a}) \equiv \int R(k;\vec{a})\psi_i(k)dk = 0, \ i = 1, \cdots, n$$
    for n weighting functions $\psi_i(k)$.
  - Collocation: zero out Euler equation errors at $k \in \{k_1, k_2, \cdots, k_n\}$:
$$P_i(\vec{a}) \equiv R(k_i;\vec{a}) = 0 \ , \ i = 1, \cdots, n$$

...

- Details of $\int ...dk$ computation:
  - Exact integration seldom possible in nonlinear problems.
  - Use quadrature formulas – they tell us what are *good* points.
  - Monte Carlo – often mistakenly used for high–dimension integrals
  - Number Theoretic methods – best for large dimension
- Details of solving $\vec{a}$:
  - Jacobian, $\vec{P}_{\vec{a}}(\vec{a})$, should be well-conditioned
  - Newton's method is quadratically convergent since it uses Jacobian
  - Functional iteration and time iteration ignore Jacobian and are linearly convergent.
  - Homotopy methods are almost surely globally convergent
  - Least squares may be ill-conditioned (that is, be flat in some directions).

Bounded Rationality Accuracy Measure

Consider the one-period relative Euler equation error:

$$E(k) = 1 - \frac{(u')^{-1}\left(\beta u'\left(C\left(f(k) - C(k)\right)\right) f'\left(f(k) - C(k)\right)\right)}{C(k)}$$

- Equilibrium requires it to be zero.
- $E(k)$ is measure of optimization error
  - 1 is unacceptably large
  - Values such as .00001 is a limit for people.
  - $E(k)$ is unit-free.
- Define the L^p, $1 \leq p < \infty$, *bounded rationality accuracy* to be

$$\log_{10} \parallel E(k) \parallel_p$$

- The L^∞ error is the maximum value of $E(k)$.

Numerical Results

- Machine: Compaq 386/20 w/ Weitek 1167
- Speed: Deterministic case: < 15 seconds
- Accuracy: Deterministic case: 8th order polynomial agrees with 250,000–point discretization to within 1/100,000.

General Projection Method

- **Step 0:** Express solution in terms of unknown functions

$$0 = \mathcal{N}(h)$$

where the $h(x)$ are decision and price rules expressing equilibrium dependence on the state x

- **Step 1:** Choose space for approximation:
  - Basis for approximation for h:

$$\{\varphi_i\}_{i=1}^{\infty} \equiv \Phi$$

  - Norm:

$$\langle \cdot, \cdot \rangle : C_+^0 \times C_+^0 \to R$$

    basis should be complete in space of C_+^0 functions basis should be orthogonal w.r.t. $\langle \cdot, \cdot \rangle$ norm and basis should be easy to compute norm and basis should be "appropriate" for problem norms are often of form $\langle f, g \rangle = \int_D f(x)g(x)w(x)dx$ for some $w(x) > 0$
  - Goal: Find $\widehat{h}$ which "nearly" solves $\mathcal{N}\ (\widehat{h}) = 0$

$$\widehat{h} \equiv \sum_{i=1}^{n} a_i\ \varphi_i$$

  - We have converted an infinite-dimensional problem to a problem in R^n
    - ∗ No discretization of state space.
    - ∗ Instead, discretize in a functional (spectral) domain.

- Example Bases:
  - $\Phi = \{1, k, k^2, k^3, \cdots\}$
  - $\Phi = \{\sin\ k,\ \sin\ 2k, \cdots\}$: Fourier – (periodic problems)
  - $\varphi_n = T_n(x)$: Chebyshev polynomials – (for smooth, nonperiodic problems)
  - B-Splines (smooth generalizations of step and tent functions).
- Nonlinear generalization
  - For some parametric form, $\Phi(x; a)$

$$\widehat{h}(x; a) \equiv \Phi(x; a)$$

  - Examples:
    - Neural networks
    - Rational functions
- Goal: Find an

$$\widehat{h} \equiv \Phi(x; a)$$

which "nearly" solves $\mathcal{N}\left(\widehat{h}\right) = 0$. Promising direction but tools of linear functional analysis and approximation theory are not available.

- **Step 2:** Compute residual function:

$$R(\cdot, a) = \widehat{\mathcal{N}}(\widehat{h}) \doteq \mathcal{N}(\widehat{h}) \doteq \mathcal{N}(h)$$

- **Step 3:** Choose $\vec{a}$ so that $R(\cdot;\vec{a})$ is "small" in $\langle \cdot, \cdot \rangle$.

  – Alternative Criteria:

    ∗ Least-Squares

    $$\min_{\vec{a}} \ \langle R(\cdot;\vec{a}),\ R(\cdot;\vec{a}) \rangle$$

    ∗ Galerkin

    $$P_i(\vec{a}) \equiv \langle R(\cdot;\vec{a}),\ \varphi_i \rangle \ = 0, i = 1, \cdots, n$$

    ∗ Method of Moments

    $$P_i(\vec{a}) \equiv \left\langle R(\cdot;\vec{a}), k^{i-1} \right\rangle = 0\ ,\ i = 1, \cdots, n$$

    ∗ Collocation

    $$P_i(\vec{a}) \equiv R(k_i;\vec{a}) = 0\ ,\ i = 1, \cdots, n,\ k_i \in \{k_1, k_2, \cdots, k_n\}$$

    ∗ Orthogonal Collocation (a.k.a. Pseudospectral)

    $$P_i(\vec{a}) \equiv R(k_i;\vec{a}) = 0\ ,\ i = 1, \cdots, n,\ k_i \in \{k : \varphi_n(k) = 0\}$$

- Details of $\langle\cdot,\cdot\rangle$ computation:
  - Exact integration seldom possible in nonlinear problems.
  - Use quadrature formulas – they tell us what are *good* points.
  - Monte Carlo – often mistakenly used for high–dimension integrals
  - Number Theoretic methods – best for large dimension
- Details of solving $\vec{a}$:
  - Jacobian, $\vec{P}_{\vec{a}}(\vec{a})$, should be well-conditioned.
  - Newton's method is quadratically convergent since it uses Jacobian; functional iteration (e.g., parameterized expectations) and time iteration ignore Jacobian and are linearly convergent.
  - If Φ is orthogonal w.r.t. $\langle\cdot,\cdot\rangle$, then Galerkin method uses orthogonal projections, helping with conditioning.
  - Least squares uses

    $$\left\langle R, \frac{\partial R}{\partial a_i} \right\rangle = 0$$

    projection conditions, which may lead to ill-conditioning.

Convergence Properties of Galerkin Methods

- Zeidler (1989): If the nonlinear operator $\mathcal{N}$ is monotone, coercive, and satisfies a growth condition then Galerkin method proves existence and works numerically.
- Krasnosel'skii and Zabreiko (1984): If $\mathcal{N}$ satisfies certain degree conditions, then a large set of projection methods (e.g., Galerkin methods with numerical quadrature) converge.
- Convergence is neither sufficient nor necessary
  - Usually only locally valid
  - Convergence theorems don't tell you when to stop.
  - Non-convergent methods are no worse if they satisfy stopping rules

Perfect Foresight Models

- General model
  - $x_t \in R^n$: list of time t values consumption, labor supply, capital stock, output, prices, interest rates, wages, etc.
  - z_t: list of exogenous variables, such as productivity levels, tax rates, monetary growth rates, etc., at time t.
  - Perfect foresight equations

$$
\begin{aligned}
g(t, \overrightarrow{x}, \overrightarrow{z}) &= 0, \ t = 0, 1, 2, ... \\
x_{0,i} &= \overline{x}_{0,i}, \ i = 1, 2, ..., n_I \\
x_t & \quad \text{bounded}
\end{aligned}
$$

- Optimal growth example:

$$\begin{aligned}
&\max_{c_t} \sum_{t=0}^{\infty} \beta^t u(c_t) \\
s.t. \quad & k_{t+1} = F(k_t) - c_t \\
& k_0 = \overline{k}_0
\end{aligned}$$

implies the Euler equation

$$u'(c_t) - \beta u'(c_{t+1}) F'(k_{t+1}) = 0, \; t = 0, 1, 2, ...$$

Eliminate c_t to arrive at equations for k_t

$$\begin{aligned}
g(t, \overrightarrow{k}) \equiv & \; u'(F(k_t) - k_{t+1}) \\
& - \beta u'(F(k_{t+1}) - k_{t+2}) F'(k_{t+1}) = 0, \; t = 0, 1, ... \\
k_0 = & \; \overline{k}_0 \\
\lim_{t \to \infty} k_t \to & \; k^{ss}
\end{aligned} \tag{1}$$

Newton Method

- Canonical model

$$g(t, x_t, x_{t+1}) = 0,\ t = 0, 1, 2, ...$$

- Fair-Taylor (Ecm., 1983)
  - A Gauss-Jacobi scheme
  - Slow, possibly nonconvergent
- L-B-J (see Boucekkine, (JEDC, 1995), and Juillard et al (JEDC, 1998))
  - Sparse Jacobian: time t eq'n depends on only (x_t, x_{t+1})

$$J(x) = \begin{pmatrix} g_2(1,x_1,x_2) & g_3(1,x_1,x_2) & 0 & \cdots \\ 0 & g_2(2,x_2,x_3) & g_3(2,x_2,x_3) & \cdots \\ 0 & 0 & g_2(3,x_3,x_4) & \cdots \\ 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & \ddots \\ \vdots & \vdots & \vdots & \end{pmatrix}$$

  - Use sparse Newton method from large systems literature

$$\begin{aligned} J(x^k)\Delta &= -g(x^k) \\ x^{k+1} &= x^k + \Delta \end{aligned} \tag{2}$$

  - Faster, more accurate than Fair-Taylor

Recursive Models and Dynamic Iteration Methods

- Consider representative agent growth problem

$$\begin{aligned} &\max_{c_t} \textstyle\sum_{t=0}^{\infty} \beta^t\, u(c_t), \\ &\text{s.t.} \quad k_{t+1} = F(k_t) - c_t. \end{aligned} \tag{16.4.1}$$

- Equilibrium consumption rule $C(k)$ satisfies

$$u'(C(k)) = \beta u'(C(F(k) - C(k)))F'(F(k) - C(k)) \tag{16.4.2}$$

  - $C(k)$ is zero of operator

$$\begin{aligned} 0 &= u'(C(k)) - \beta u'(C(F(k) - C(k)))F'(F(k) - C(k)) \\ &\equiv (\mathcal{N}(C))(k) \end{aligned} \tag{16.4.3}$$

  - $\mathcal{N}$ is an operator from continuous functions to continuous functions.

- Consider the four occurrences of C and define the operator $\mathcal{F}$:

$$\begin{aligned} 0 &= u'(C_1) - \beta u'(C_2(F - C_3))\, F'(F - C_4) \\ &\equiv \mathcal{F}(C_1, C_2, C_3, C_4). \end{aligned} \tag{16.4.4}$$

  - We want a function C that solves the equation

$$0 = \mathcal{F}(C, C, C, C) \equiv \mathcal{N}(C). \tag{16.4.5}$$

Time Iteration

- Time iteration implements the iterative scheme

$$0 = u'(C_{i+1}) - \beta u'(C_i(F - C_{i+1}))\, F'(F - C_{i+1}) \tag{16.4.7}$$

  – Intuition: if $C_i(k)$ is tomorrow's consumption policy function, then today's policy, denoted by $C_{i+1}(k)$, must satisfy

$$u'(C_{i+1}(k)) = \beta u'(C_i(F(k) - C_{i+1}(k)))\, F'(F(k) - C_{i+1}(k)). \tag{16.4.9}$$

  – In terms of $\mathcal{F}$, time iteration is the iteration implicitly defined by

$$0 = \mathcal{F}(C_{i+1}, C_i, C_{i+1}, C_{i+1}). \tag{16.4.8}$$

- Convergence

  – Monotonicity property of (16.4.9); that is, if $C_i^{'}(k) > 0$ and $C_i(k) < C_{i-1}(k)$ then $C_{i+1}(k) < C_i(k)$ and C_{i+1} is an increasing function.

  – Monotonicity implies monotone convergence of (16.4.7)

  – However, numerical implementations may introduce numerical error which violates monotonicity.

Fixed-Point Iteration

- Fixed-point iteration applied to (16.4.3) implements the implicit iterative scheme

$$0 = \mathcal{F}(C_{i+1}, C_i, C_i, C_i). \tag{16.4.10}$$

- C_{i+1} is easy to compute since at any k,

$$\begin{aligned} C_{i+1}(k) &= (u')^{-1}\left(\beta u'(C_i(F(k) - C_i(k)))\, F'(F(k) - C_i(k))\right) \\ &\equiv (T_{fp}(C_i))(k) \end{aligned} \tag{16.4.11}$$

- Convergence is not guaranteed

Recursive Models with Nonlinear Equation Methods

- Use nonlinear equations and Chebyshev approximations to solve

$$0 = \mathcal{F}(C, C, C, C). \tag{16.4.10}$$

- No economic "intuition" or "story"; it just works!
- Approximate C with the linear representation

$$\widehat{C}(k\,;\,a) = \sum_{i=1}^{n} a_i \psi_i(k), \tag{16.5.1}$$

  - $\psi_i(k) \equiv T_{i-1}\left(2\frac{k-k_m}{k_M-k_m} - 1\right)$ and n is the number of terms used.
  - Domain D is $[k_m, k_M]$.

- Residual function

$$R(k\,;a) = u'(\widehat{C}(k\,;a)) - \beta u'(\widehat{C}(F(k) - \widehat{C}(k\,;a);a))F'(F(k) - \widehat{C}(k\,;a)). \tag{16.5.2}$$

- Orthogonal collocation chooses k_j and solves

$$R(k_j\,;a) = 0, \;\; j = 1, ..., n. \tag{16.5.3}$$

- Multiple solutions
  - Multiple solutions to first-order conditions exist
  - Only one satisfies global stability
  - If initial guess is close then one typically converges to correct answer
  - Can sometimes avoid bad ones
    - Specify steady state
    - Pick functional form which cannot go bad
    - Alter problem to penalize divergent paths

Coefficients of Solution

- Theoretical predictions
  - Approximation theory says that the Chebyshev coefficients should fall rapidly if $C(k)$ is smooth.
  - Orthogonal basis should imply that coefficients do not change as we increase n.
- Table 16.1 verifies these predictions.

Table 16.1: Chebyshev Coefficients for Consumption Function

k	$n = 2$	$n = 5$	$n = 9$	$n = 15$
1	0.0589755899	0.0600095844	0.0600137797	0.0600137922
2	0.0281934398	0.0284278730	0.0284329464	0.0284329804
3		−0.0114191783	−0.0113529374	−0.0113529464
4		0.0007725731	0.0006990930	0.0006988353
5		−0.0001616767	−0.0001633928	−0.0001634209
6			0.0000427201	0.0000430853
7			−0.0000123570	−0.0000122160
8			0.0000042498	0.0000036367
9			−0.0000011464	−0.0000011212
10				0.0000003557
11				−0.0000001147
12				0.0000000370

Each entry is the coefficient of the k'th Chebyshev polynomial (over the interval [.333, 1.667]) in the n-term approximation of the consumption policy function in (4.3) for the case discussed in Section 4.2.

Errors in Consumption Policy Function

- "Truth" computed by a 1,000,000 state discrete approximation
- "True solution" also has some error because of discretization
- Table 16.2 displays difference between approximations and "truth"

Table 16.2: Policy Function Errors

k	y	c	$n = 20$	$n = 10$	$n = 7$	$n = 4$	$n = 2$
0.5	0.1253211	0.1010611	1(-7)	5(-7)	5(-7)	2(-7)	5(-5)
0.6	0.1331736	0.1132936	2(-6)	1(-7)	1(-7)	2(-6)	8(-5)
0.7	0.1401954	0.1250054	2(-6)	3(-7)	3(-7)	1(-6)	2(-4)
0.8	0.1465765	0.1362965	1(-6)	4(-7)	4(-7)	4(-6)	2(-4)
0.9	0.1524457	0.1472357	1(-6)	3(-7)	3(-7)	5(-6)	2(-4)
1.0	0.1578947	0.1578947	4(-6)	0(-7)	1(-7)	2(-6)	1(-4)
1.1	0.1629916	0.1683016	4(-6)	2(-7)	2(-7)	1(-6)	9(-5)
1.2	0.1677882	0.1784982	3(-6)	2(-7)	2(-7)	4(-6)	7(-6)
1.3	0.1723252	0.1884952	7(-7)	4(-7)	4(-7)	3(-6)	9(-5)

Stochastic Dynamic General Equilibrium

- Canonical RBC Model

$$\begin{aligned} \max_{c_t} E &\left\{ \sum_{t=1}^{\infty} \beta^t u(c_t) \right\} \\ k_{t+1} &= \theta_t f(k_t) - c_t \\ \ln \theta_{t+1} &= \rho \ln \theta_t + \varepsilon_t \end{aligned}$$

- Euler equation

$$u'(c_t) = \beta E\{u'(c_{t+1})\theta_{t+1} f'(k_{t+1}) | \theta_t\}$$

  – Consumption is determined by recursive function

$$c_t = C(k_t, \theta_t)$$

  – $C(k, \theta)$ satisfies functional equation

$$0 = u'(C(k,\theta)) - \beta E \left\{ u' \left(C \left(\theta f(k) - C(k,\theta), \tilde{\theta} \right) \right) \tilde{\theta} f'(\theta f(k) - C(k,\theta)) \mid \theta \right\}$$

- Transform Euler equation into the more linear form

$$\begin{aligned} 0 = C(k,\theta) - (u')^{-1} &\left(\beta E \left\{ u' \left(C(\theta f(k) - C(k,\theta), \tilde{\theta}) \right) \times \tilde{\theta} f' \left(\theta f(k) - C(k,\theta) \right) \mid \theta \right\} \right) \\ &\equiv \mathcal{N}(C)(k,\theta) \end{aligned}$$

but this rewriting is not essential

- Approximate policy function

$$\widehat{C}(k,\theta;\ \mathbf{a}) = \sum_{i=1}^{n_k}\sum_{j=1}^{n_\theta} a_{ij}\psi_{ij}(k,\theta)$$

$$\psi_{ij}(k,\theta) \equiv T_{i-1}\left(2\frac{k-k_m}{k_M-k_m}-1\right) T_{j-1}\left(2\frac{\theta-\theta_m}{\theta_M-\theta_m}-1\right)$$

- Define integrand of expectations

$$I(k,\theta,\ \mathbf{a},z) = u'\left(\widehat{C}\left(\theta f(k)-\widehat{C}(k,\theta;\ \mathbf{a}), e^{\sigma z}\theta^{\rho},\ \mathbf{a}\right)\right) \times e^{\sigma z}\theta^{\rho} f'\left(\theta f(k)-\widehat{C}(k,\theta;\ \mathbf{a})\right)\pi^{-\frac{1}{2}}$$

- $\mathcal{N}\left(\widehat{C}\left(\cdot,\cdot;\ \mathbf{a}\right)\right)(k,\theta)$ becomes

$$\widehat{C}(k,\theta;\ \mathbf{a}) - (u')^{-1}\left(\beta\int_{-\infty}^{\infty} I(k,\theta;\ \mathbf{a},z)\,\frac{e^{-z^2/2}}{\sqrt{2\pi}}\,dz\right)$$

- Use Gauss-Hermite quadrature over z:

$$\int_{-\infty}^{\infty} I(k,\theta,\ \mathbf{a},z)\,\frac{e^{-z^2/2}}{\sqrt{2}}\,dz \doteq \sum_{j=1}^{m_z} I\left(k,\theta,\ \mathbf{a},\sqrt{2}z_j\right)\omega_j$$

  where ω_j, z_j are Gauss-Hermite quadrature weights and points.

- The computable residual function is

$$R(k,\theta;\ \mathbf{a}) = \widehat{C}(k,\theta;\ \mathbf{a}) - (u')^{-1}\left(\beta\sum_{j=1}^{m_z} I\left(k,\theta,\ \mathbf{a},\sqrt{2}z_j\right)w_j\right) \equiv \widehat{\mathcal{N}}\left(\widehat{C}(\cdot,\cdot;\ \mathbf{a})\right)(k,\theta).$$

- Fitting Criteria:
  - Collocation:
    - Choose n_k capital stocks, $\{k_i\}_{i=1}^{n_k}$, and n_θ productivity levels, $\{\theta_i\}_{j=1}^{n_\theta}$
    - Find **a** such that

$$R(k_i, \theta_j;\ \mathbf{a}) = 0,\ i = 1, \cdots, n_k,\ j = 1, \cdots, n_\theta$$

  - Galerkin:
    - Compute the $n_k n_\theta$ projections with Chebyshev weight $w(k, \theta)$ adapted to $[k_m, k_M] \times [\theta_m, \theta_M]$

$$P_{ij}(\ \mathbf{a}) \equiv \int_{k_m}^{k_M} \int_{\theta_m}^{\theta_M} R(k, \theta;\ \mathbf{a}) \psi_{ij}(k, \theta) w(k, \theta)\, d\theta dk$$

    - Approximate projections by Gauss-Chebyshev quadrature

$$\hat{P}_{ij}(\mathbf{a}) \equiv \sum_{\ell_k=1}^{m_k} \sum_{\ell_\theta=1}^{m_\theta} R(k_i, \theta_j;\ \mathbf{a}) \psi_{ij}(k_{\ell_k}, \theta_{\ell_\theta}),$$

      where

$$\begin{aligned} k_{\ell_\theta} &= k_m + \frac{1}{2}(k_M - k_m)\left(z_{\ell_k}^{m_k} + 1\right),\ \ell_k = 1, \ldots, m_k \\ \theta_{\ell_\theta} &= \theta_m + \frac{1}{2}(\theta_M - \theta_m)\left(z_{\ell_\theta}^{m_\theta} + 1\right),\ \ell_\theta = 1, \ldots, m_\theta \\ z_\ell^n &\equiv \cos\left(\frac{(2i-1)\pi}{2n}\right),\ \ell = 1, \ldots, n \end{aligned}$$

    - Coefficients, **a**, are fixed by the system (solved by Newton's method)

$$\hat{P}_{ij}(\mathbf{a}) = 0,\ i = 1, \cdots, n_k,\ j = 1, \cdots, n_\theta$$

- Bounded Rationality Accuracy Measure
  - Consider the computable Euler equation error
    $$E(k,\theta) = \frac{\widehat{\mathcal{N}}(\widehat{C}(\cdot,\cdot;\ \mathbf{a}))(k,\theta)}{\widehat{C}(k,\theta;\ \mathbf{a})}$$
    where $\widehat{\mathcal{N}}$ uses some integration formula for $E\{\cdot\}$; need not be the same as used in computing $R(k,\theta;\mathbf{a})$. In fact, should use better one.
  - Define the L^p, $1 \leq p < \infty$, *bounded rationality accuracy* to be
    $$\log_{10} \parallel E(k) \parallel_p$$
- Verify solution: Accept solution to projection equations, $\mathbf{a}$, only if it passes tests
  - Check stability
    * For example, there should be positive savings at low k, high θ
    * Could simulate capital stock process implied by $\widehat{C}(k,\theta;\ \mathbf{a})$ to see if it has a stationary distribution
  - Check Euler equation errors
    * $E(k,\theta)$ should be moderate for most (k,θ) points in $[k_m,k_M] \times [\theta_m,\theta_M]$
    * $E(k,\theta)$ should be small for most (k,θ) points frequently visited
  - If $\widehat{C}(k,\theta;\ \mathbf{a})$ does not pass these tests, go back and use higher values for n_k and n_θ, and increase m_k,and m_θ

- Numerical Results
  - Basis: Chebyshev polynomials
  - Initial guess: Linear rule through deterministic steady state and zero.
  - $k \in [.333, 2.000]$
  - Method: Collocation and Galerkin.
  - Newton's method solved projection equations, $P_i(\mathbf{a}) = 0$, for $\mathbf{a}$.
  - Machine: Compaq 386/20 (old, but relative speeds are still valid)
  - Speed: Stochastic case: under two minutes for a 60 parameter fit.
  - Errors: 2% for 6 parameter fit, .1% for 60 parameter fit – about a penny loss per $10,000 dollar expenditure
  - Orth. poly. + orthog. collocation + Gaussian quad. + Newton outperforms naive methods by factor of 10 or greater; exceeded Monte Carlo methods by factor of 100+.
  - $\widehat{C}(k, \theta;\ \mathbf{a})$ is an ε-equilibrium with small ε – a bounded rationality interpretation.

Table 17.1: $\log_{10}$ Euler Equation Errors

			$\parallel E \parallel_\infty$	$\parallel E \parallel_1$	$\parallel E \parallel_\infty$	$\parallel E \parallel_1$
γ	ρ	σ	(2, 2, 2, 2)*		(4, 3, 4, 3)	
−15.00	0.80	0.01	−2.13	−2.80	−3.00	−3.83
−15.00	0.80	0.04	−1.89	−2.54	−2.44	−2.87
−15.00	0.30	0.04	−2.13	−2.80	−2.97	−3.83
− 0.10	0.80	0.04	0.01	−1.19	−1.48	−2.22
− 0.10	0.30	0.04	0.18	−1.22	−1.63	−2.65
			(7, 5, 7, 5)		(7, 5, 20, 12)	
−15.00	0.80	0.01	−4.28	−5.19	−4.43	−5.18
−15.00	0.80	0.04	−3.36	−4.00	−3.30	−3.95
−15.00	0.30	0.04	−4.24	−5.19	−4.38	−5.18
− 0.10	0.80	0.04	−2.50	−3.22	−2.60	−3.17
− 0.10	0.30	0.04	−3.43	−4.37	−3.49	−4.39
			(10, 6, 10, 6)		(10, 6, 25, 15)	
−15.00	0.80	0.01	−5.48	−6.43	−5.61	−6.42
−15.00	0.80	0.04	−3.81	−4.38	−3.88	−4.37
−15.00	0.30	0.04	−5.45	−6.43	−5.57	−6.42
−0.10	0.80	0.04	−2.99	−3.68	−3.09	−3.64
−0.10	0.30	0.04	−5.17	−6.12	−5.23	−6.14

*$(n_k,\ n_\theta,\ m_k,\ m_\theta)$

Table 17.2: Alternative Implementations

$n_k = 7, n_\theta = 5, m_k = 7, m_\theta = 5$

γ	ρ	σ	G[a]		P[b]		U[c]		UP[d]	
			error[e]	time	error	time	error	time	error	time
−15	.8	.04	−3.18	1:15	−2.13	:40	−3.06	1:05	−2.19	:44
	.3	.01	−4.35	:11	−4.35	:52	−4.07	:08	−4.07	1:47
−.9	.8	.04	−3.43	:05	−3.43	:19	−3.42	:08	−3.42	:39
	.3	.01	−4.03	:07	−4.03	:30	−3.76	:07	−3.76	1:10

$n_k = 10, n_\theta = 6, m_k = 25, m_\theta = 15$

γ	ρ	σ	G error	G time	P error	P time	U error	U time	UP error	UP time
−15	.8	.04	−3.87	4:20	−3.90	24:44	−3.90	3:41	−3.36	42:15
	.3	.01	−5.68	2:19	−5.14	11:31	−5.49	2:14	−5.30	8:06
−.9	.8	.04	−4.00	1:31	−4.00	5:17	−4.01	1:31	−4.01	5:02
	.3	.01	−5.40	1:23	−4.63	7:13	−5.25	1:20	−5.13	6:01

[a]Chebyshev polynomial basis, Chebyshev zeroes used in evaluating fit

[b]Ordinary polynomial basis, Chebyshev zeroes used in evaluating fit

[c]Chebyshev polynomial basis, uniform grid points

[d]Ordinary polynomial basis, uniform grid points

[e]error measure is $\| E(k) \|_\infty$

Table 17.3: Tensor Product vs. Complete Polynomials[a]

			Tensor Product			Complete Polynomials		
γ	ρ	σ	$n=3$	$n=6$	$n=10$	$n=3$	$n=6$	$n=10$
-15.0	.8	.04	-2.34[b]	-3.26	-3.48	-1.89	-3.10	-4.06
			:01[c]	:13	14:21	:03	:07	1:09
$-.9$	.3	.10	-2.19	-3.60	-5.27	-2.14	-3.55	-5.22
			:01	:08	1:21	:01	:05	:32
$-.1$	.3	.01	-1.00	-2.84	-5.21	-0.99	-2.83	-5.17
			:01	:08	1:24	:01	:05	:35

[b] $\log_{10} \parallel E \parallel_\infty$; [c] Computation time expressed in minutes :seconds.

- Tensor product cases used orthogonal collocation with $n_k = n_\theta = m_k = m_\theta = n$ to identify the n^2 free parameters. Complete polynomial cases used Galerkin projections to identify the $1 + n + n(n+1)/2$ free parameters..

- General Observations:
  - Tensor product of degree n takes more time, but achieves higher accuracy
  - For a specific level of accuracy, complete polynomial method is faster

Fixed-Point Iteration

- Gaussian Quadrature Implementation
  - In fixed-point iteration we execute the iteration

$$\begin{aligned} C_{i+1}(k,\theta) &= (u')^{-1}\left(\beta\, E\,\left\{u'\left(C_i(k^+,\theta^+)\right)\, F_k\left(k^+,\theta^+\right) \mid \theta\right\}\right), \\ k^+ &\equiv F(k,\theta) - C_i(k,\theta), \\ \theta^+ &\sim N(\rho\ \ln\theta, \sigma^2). \end{aligned} \tag{3}$$

  - Note that this is a simple rewriting of Euler equation.
  - Strategy: compute the right hand side for several (k,θ) choices and interpolate to get new C.
  - RHS of (3) is tomorrow's return on saving one more dollar today conditional on today's (k,θ) and if $c = C(k,\theta)$ at all times.
  - RHS of (3) is integral over θ^+.
    * Since integrand is smooth and Gaussian, use four- or five-point Gauss-Hermite rule is adequate.
    * Therefore, to compute the, say, quadratic, solution to this problem, one need only examine a handful of k and θ values. This results in a very rapid way to compute an iterate.
    * May converge slowly if at all; use extrapolation or dampening.
  - This is essentially Miranda-Helmburger (AER, 1988)

Examples: Multiagent Dynamic General Equilibrium

- Model:
  - n types of agents, utility functions, $u_i(c)$, $i = 1, 2, ..., n$,
  - Common discount factor β.
  - Equity is the only asset
  - $c_i = C_i(k)$, wealth distribution is $k = (k_1, k_2, ..., k_n)$
- Approximate $c_i = \widehat{C}_i(k, \theta; \mathbf{a})$.
- Euler equation for type $i = 1, 2, ..., n$

$$\begin{aligned} R_i(k,\theta,C) &= u_i'(C_i(k,\theta)) - \beta\, E\, \{u'(C_i(Y(k,\theta) - C(k,\theta),\tilde{\theta})) \\ &\quad \times F_k(Y(k,\theta) - C(k,\theta),\tilde{\theta}) \mid \theta\} \end{aligned}$$

where

$$\begin{aligned} Y_i(k,\theta) &= k_i F_1(\overline{k},\theta) + w(\overline{k},\theta),\ \ i = 1, .., n \\ w(\overline{k},\theta) &= F(\overline{k},\theta) - \overline{k}F_1(\overline{k},\theta) \\ \overline{k} &\equiv \sum_i k_i. \end{aligned}$$

- Approximate residual function for agents of type $i = 1, 2, ..., n$

$$\begin{aligned} \widehat{R}_i(k, \theta, \widehat{C}(\cdot; \mathbf{a})) &= \widehat{C}_i(k, \theta; \mathbf{a}) - (u_i')^{-1} \left(\beta \, \widehat{E} \left\{ u_i' \left(c^+ \right) F_k \left(k_\cdot^+, \tilde{\theta} \right) \mid \theta \right\} \right) \\ & \quad c_i^+ \equiv \widehat{C}_i(y^+, \tilde{\theta}; \mathbf{a}) \\ & \quad k^+ \equiv Y(k, \theta; \mathbf{a}) - \widehat{C}(k, \theta; a) \end{aligned}$$

  where $\widehat{E}$ is a numerical approximation of the integral. Use product Gaussian quadrature

- Identifying projections are

$$P_{ij}(\mathbf{a}) \equiv \int_{\theta_m}^{\theta_M} \int_{k_m}^{k_M} \cdots \int_{k_m}^{k_M} \widehat{R}_i(k, \theta, \widehat{C}(\cdot; \mathbf{a})) \, \psi_j(k, \theta) \, w(k, \theta) \, dk_1 \cdots dk_n d\theta$$

  where $i = 1, ..., n$, and $j = 1, ..., m$.

- Let $\widehat{P}(\mathbf{a})$ denote a numerical integration approximation of $P(\mathbf{a})$; we will use product Gaussian quadrature

- Solution chooses $\mathbf{a}$ so that $\widehat{P}(\mathbf{a}) = 0$.

Representation: Tensor vs. Complete Polynomials

- Tensor method:

$$\widehat{C}_i(k, \theta; \mathbf{a}) = \sum_{j_1=0}^{n_k} \cdots \sum_{j_n=0}^{n_k} \sum_{\ell=0}^{n_\theta} a^i_{j_1 \cdots j_n \ell} \, \varphi_{i_1}(k_1) \cdots \varphi_{i_n}(k_n) \, \psi_\ell(\theta), \quad i = 1, ..., n$$

  where $\varphi_i(k_j)$ $(\psi_\ell(\theta))$ is a degree $i-1$ $(\ell-1)$polynomial in k_j (θ) from some orthogonal family.

- Complete polynomial method

$$C_i(k, \theta; \mathbf{a}) = \sum_{\substack{0 \le j_1 + \cdots + j_n + \ell \le d \\ 0 \le j_i, \ell \le d}} a^i_{j_1 \cdots j_n \ell} \, \varphi_{j_1}(k_1) \ldots \varphi_{j_n}(k_n) \, \psi_\ell(\theta)$$

- Number of unknown coefficients are far smaller in complete poly case, but not as flexible.

Solution Methods

- Successive Approximation: at grid of (k, θ) points (e.g., Chebyshev zeroes) and given iteration j for $\mathbf{a}$ (denoted $\mathbf{a}^j$), $\widehat{C}_i(k, \theta; \mathbf{a}^j)$, generate data

$$\begin{aligned} \widehat{C}_i(k, \theta; \mathbf{a}^{j+1}) \;&=\; (u')^{-1}\left(\beta\,\widehat{E}\,\left\{u'\left(\widehat{C}_i\left(Y(k,\theta) - \widehat{C}_i(k,\theta;\mathbf{a}), \tilde{\theta}; \mathbf{a}\right)\right)\right.\right. \\ &\left.\left.\times F_k\left(Y(k,\theta) - \widehat{C}_i(k,\theta;\mathbf{a}^j), \tilde{\theta}\right) \mid \theta\right\}\right. \end{aligned} \tag{4}$$

  and set coefficients $\mathbf{a}^{j+1}$ through interpolation or regression

- Time Iteration: same procedure except not generate data for $\widehat{C}_i(k, \theta; \mathbf{a}^{j+1})$ by solving

$$\begin{aligned} \widehat{C}_i(k, \theta; \mathbf{a}^{j+1}) \;&=\; (u')^{-1}\left(\beta\,\widehat{E}\,\left\{u'\left(\widehat{C}_i\left(Y(k,\theta) - \widehat{C}_i(k,\theta;\mathbf{a}^{j+1}), \tilde{\theta}; \mathbf{a}^j\right)\right)\right.\right. \\ &\left.\left.\times F_k\left(Y(k,\theta) - \widehat{C}_i(k,\theta;\mathbf{a}^{j+1}), \tilde{\theta}\right) \mid \theta\right\}\right. \end{aligned} \tag{5}$$

- Newton's Method: just solve nonlinear equations $\widehat{P}(\mathbf{a}) = 0$

...

Table 5: Time and Accuracy Comparisons

				num.	Newt's Method		Succ.Approx:	
agents	γ	deg	basis	coef's	time	acc'cy	time	accuracy
1	-2	1	t	4	:0.05	-2.7	:0.2	-2.7
			c	3	:0.06	-2.6	:0.4	-2.6
		2	t	9	:0.22	-3.4	:01	-3.4
			c	6	:0.17	-3.3	:01	-3.3
		3	t	16	:0.71	-4.1	:01	-4.1
			c	10	:0.49	-4.0	:02	-4.0
		4	t	25	:02	-4.8	:02	-4.9
			c	30	:0.99	-4.7	:03	-4.6
2	-1	1	t	16	:0.66	-3.1	:01	-3.1
	-2		c	6	:0.38	-2.7	:01	-2.7
		2	t	54	:07	-4.1	:08	-4.1
			c	20	:02	-3.4	:06	-3.4
		3	t	128	1:22	-5.0	:33	-4.5
			c	40	:11	-4.1	:21	-4.1
		4	t	250	12:34	-5.9	1:48	-4.5
			c	70	:45	-4.8	:56	-4.7

Note: "inf" means infeasible. "h hrs $n : m.l$" means "h hours n minutes, $m.l$ seconds".

Table 5: Time and Accuracy Comparisons (Continued)

				num	Newt's Method:		Succ. Approx.:	
agents	γ	deg	basis	coef's	time	accuracy	time	accuracy
3	-1	1	t	48	:07	-3.4	:07	-3.4
	-2		c	15	1.48	-2.8	:05	-2.8
	-3	2	t	243	7:07	-4.6	2:11	-4.5
			c	63	:21	-3.6	:36	-3.6
		3	t	768	inf	inf	19:57	-4.6
			c	105	4:05	-4.3	3:09	-4.3
4	-.5	1	t	128	1:09	-3.5	:33	-3.5
	-1		c	24	:5.10	-2.9	:13	-2.9
	-2	2	t	972	inf	inf	24:57	-4.6
	-3		c	84	2:47	-3.7	3:04	-3.7
		3	t	4096	inf	inf	7 hr 13	-4.6
			c	224	52:11	-4.4	26:01	-4.4
5	-.5	1	t	320	8:52	-3.6	2:48	-3.6
	-1		c	35	:17.90	-3.0	:38	-3.0
	-2	2	t	3645	inf	inf	5 hr 16	-4.6
	-3		c	140	12:18	-3.8	10:18	-3.8
	-4	3	t	20,480	inf	inf	inf	inf
			c	420	13 hr	-4.5	3 hr 27	-4.5

Note: "inf" means infeasible. "h hrs $n : m.l$" means "h hours n minutes, $m.l$ seconds".

Table 7: Final Comparisons

Method:	**Basis:**	**Solution Method:**	**Advantages:**	**Disadvantages:**
Taylor Series	Complete	Eigenvalues, linear eq'ns	Fast	Local validity
Projection methods	Tensor or complete	Newton	Quadratic conv.	Infeasible for large problems
	Tensor or complete	Successive approx.	Easy Iterations	possible nonconv.

Summary of Projection Method

- Can be used for problems with unknown functions
- Uses approximation ideas
- Utilizes standard optimization and nonlinear equation solving software
- Can exploit a priori information about problem
- Flexible: users choose from a variety of approximation, integration, and nonlinear equation-solving methods

Table 17.4: Projection Method Menu

Approximation	Integration	Projections	Equation Solver
Piecewise Linear	Newton-Cotes	Galerkin	Newton
Polynomials	Gaussian Rules	Collocation	Powell
Splines	Monte Carlo	M. of Moments	Fixed-pt. iteration
Neural Networks	Quasi-M.C.	Subdomain	Time iteration
Rational Functions	Monomial Rules		Homotopy
Problem Specific	Asymptotics		

- Unifies literature: Previous work can be classified and compared

	Choices		
Authors	**Approximation**	**Integration**	**Sol'n Method**
Gustafson(1959)	piecewise linear	Newt.-Cotes	S.A.-time it.
Wright-W.(1982,4)	poly. (of cond. exp.)	Newt.-Cotes	S.A.-time it.
Miranda-H.(1986)	polynomials	Newt.-Cotes	S.A.-learning
Coleman(1990)	finite element	Gaussian	S.A.-time it.
den Haan-M.(1990)	poly. (of cond. exp.)	Sim. M.C.	S.A.-learning
Judd(1992)	orthogonal poly.	Gaussian	Newton