Solving Dynamic Games with Newton’s Method

Karl Schmedders

University of Zurich

Institute for Computational Economics
University of Chicago

July 31, 2008
Discrete-Time Finite-State Stochastic Games

Central tool in analysis of strategic interactions among forward-looking players in dynamic environments

Example: The Ericson & Pakes (1995) model of dynamic competition in an oligopolistic industry

Little analytical tractability

Most popular tool in the analysis: The Pakes & McGuire (1994) algorithm to solve numerically for an MPE (and variants thereof)
Applications

Advertising (Doraszelski & Markovich 2007)


Consumer learning (Ching 2002)

Firm size distribution (Laincz & Rodrigues 2004)

Learning by doing (Benkard 2004, Besanko, Doraszelski, Kryukov & Satterthwaite 2007)
Applications cont’d

Mergers (Berry & Pakes 1993, Gowrisankaran 1999)


Productivity growth (Laincz 2005)


Technology adoption (Schivardi & Schneider 2005)

International trade (Erdem & Tybout 2003)

Need for better Computational Techniques


“Moreover the burden of currently available techniques for computing equilibria to the models we do know how to analyze is still large enough to be a limiting factor in the analysis of many empirical and theoretical issues of interest.”
This Tutorial

1. Discrete-Time Finite-State Stochastic Games

2. Separable Game

3. Solution Methods for Dynamic Games
Discrete-Time Finite-Space Stochastic Games

Solving Dynamic Games
State Space

Infinite-horizon game in discrete time \( t = 0, 1, 2, \ldots \)

Set of \( N \) players, \( i = 1, \ldots, N \)

At time \( t \) player \( i \) is in one of finitely many states \( x_{t}^{i} \in X^{i} \)

State space of the game \( X = \prod_{i} X^{i} \)

State in period \( t \) is \( x_{t} = (x_{t}^{1}, \ldots, x_{t}^{N}) \)

Notation: \( x_{t}^{-i} = (x_{t}^{1}, \ldots, x_{t}^{i-1}, x_{t}^{i+1}, \ldots, x_{t}^{N}) \)
Player’s Actions and Transitions

Player $i$’s action in period $t$ is $u^i_t \in U^i(x_t)$

Set of feasible actions $U^i(x_t)$ is arbitrary, often $U^i = \mathbb{R}_+^K$

Players’ actions at time $t$: $u_t = (u^1_t, \ldots, u^N_t)$

Law of motion: State follows a controlled discrete-time, finite-state, first-order Markov process with transition probability $\Pr(x'|u_t, x_t)$

Special case of independent transitions:

$$\Pr(x'|u_t, x_t) = \prod_{i=1}^{N} \Pr^i\left((x')^i | u^i_t, x^i_t\right)$$
Objective Function

Player $i$ receives a payoff of $\pi^i(u_t, x_t)$ in period $t$

Objective is to maximize the expected NPV of future cash flows

$$E \left\{ \sum_{t=0}^{\infty} \beta^t \pi^i(u_t, x_t) \right\},$$

with discount factor $\beta \in (0, 1)$
Bellman Equation

\( V^i(x) \) is the expected NPV to player \( i \) if the current state is \( x \)

Bellman equation for player \( i \) is

\[
V^i(x) = \max_{u^i} \pi^i \left( u^i, U^{-i}(x), x \right) + \beta E_{x'} \left\{ V^i(x') \mid u^i, U^{-i}(x), x \right\}
\]

(1)

where \( U^{-i}(x) \) denotes feedback (Markovian) strategies of other players

Player \( i \)'s strategy is given by

\[
U^i(x) = \arg \max_{u^i} \pi^i \left( u^i, U^{-i}(x), x \right) + \beta E_{x'} \left\{ V^i(x') \mid u^i, U^{-i}(x), x \right\}
\]

(2)

System of equations defined by (1) and (2) for each player \( i = 1, \ldots, N \) and each state \( x \in X \) defines a pure-strategy MPE
Example of a Separable Game: Patent Race
Patent Race Between Two Firms

$N$ innovation stages

Firms start race at stage 0

Period $t$ innovation stages: $(x_{1,t}, x_{2,t})$ where 
$x_{i,t} \in X \equiv \{0, ..., N\}, i = 1, 2$

Period $t$ investment: $a_{i,t} \in A = [0, \bar{A}] \subset \mathbb{R}_+, i = 1, 2$

Cost of investment: $C_i(a) = c_i a^\eta, \eta \in \mathbb{N}, c_i > 0, i = 1, 2$

Independent and stochastic innovation technologies
Transition from State to State

Transition from period to period: \( x_{i,t+1} = x_{i,t} \) or \( x_{i,t+1} = x_{i,t} + 1 \)

Markov process (depends on investment levels)

Firm \( i \)'s state evolves according to

\[
x_{i,t+1} = \begin{cases} 
  x_{i,t}, & \text{with probability } p(x_{i,t}|a_{i,t}, x_{i,t}) \\
  x_{i,t} + 1, & \text{with probability } p(x_{i,t} + 1|a_{i,t}, x_{i,t}) 
\end{cases}
\]

Distribution over next period's states

\[
p(x|a, x) = F(x|x)^a
\]

\[
p(x + 1|a, x) = 1 - F(x|x)^a
\]

\( F(x|x) \in (0, 1) \) is probability that there is no change in state if \( a = 1 \)
Firms’ Optimization Problem

First firm to reach state $N$ wins the race and receives prize $\Omega$

Ties are broken by flip of a coin

Firms discount future costs and revenues at common rate $\beta < 1$

Firms’ objective: maximize expected discounted payoffs
Equilibrium I

Restriction to pure Markov strategies

Firm i’s strategy: \( \sigma_i(\cdot) : X \times X \to A \)

Expected discounted payoff: \( V_i(\cdot) : X \times X \to \mathbb{R} \)

Bellmann equation for \( x_i, x_{-i} < N \),

\[
V_i(x_i, x_{-i}) = \max_{a_i \in A} \left\{ -C_i(a_i) + \beta \sum_{x_i', x_{-i}'} p(x_i' | a_i, x_i) p(x_{-i}' | a_{-i}, x_{-i}) V_i(x_i', x_{-i}') \right\}
\]
Equilibrium II

Boundary condition at terminal states

\[ V_i(x_i, x_{-i}) = \begin{cases} 
\Omega, & \text{for } x_{-i} < x_i = N \\
\Omega/2, & \text{for } x_i = x_{-i} = N \\
0, & \text{for } x_i < x_{-i} = N 
\end{cases} \]

Optimal strategies satisfy

\[ \sigma_i(x_i, x_{-i}) = \arg \max_{a_i \in A} \left\{ -C_i(a_i) + \beta \sum_{x'_i, x'_{-i}} p(x'_i|a_i, x_i)p(x'_{-i}|a_{-i}, x_{-i}) V_i(x'_i, x'_{-i}) \right\} \]
Our Equilibrium Equations

\[ 0 = -V_i(x_i, x_{-i}) - c_i a_i^{\eta} + \beta \sum_{x_i', x_{-i}'} p(x_i'|a_i, x_i)p(x_{-i}'|a_{-i}, x_{-i})V_i(x_i', x_{-i}') \]

\[ 0 = -\eta c_i a_i^{\eta-1} + \beta \sum_{x_i', x_{-i}'} \frac{\partial}{\partial a_i} p(x_i'|a_i, x_i)p(x_{-i}'|a_{-i}, x_{-i})V_i(x_i', x_{-i}') \]

Parameter specification: \( c_1, c_2, \eta, F(x_1, x_2) \equiv F, \Omega \)

Unknowns: \( V_1(x_1, x_2), V_2(x_1, x_2), a_1(x_1, x_2), a_2(x_1, x_2) \)

Four equations per stage \((x_i, x_{-i})\)

Backward induction: instead of solving all equations simultaneously

solve each stage game separately
Solving Systems of Nonlinear Equations
Nonlinear Systems of Equations

System $F(x) = 0$ of $n$ nonlinear equations in $n$ variables

$x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$

\[
\begin{align*}
F_1(x_1, x_2, \ldots, x_n) &= 0 \\
F_2(x_1, x_2, \ldots, x_n) &= 0 \\
\vdots \\
F_{n-1}(x_1, x_2, \ldots, x_n) &= 0 \\
F_n(x_1, x_2, \ldots, x_n) &= 0
\end{align*}
\]

Initial guess $x^0 = (x_1^0, x_2^0, \ldots, x_n^0)$

Methods generate a sequence of iterates $x^0, x^1, x^2, \ldots, x^k, x^{k+1}, \ldots$
Solution Methods

Most popular methods in economics for solving $F(x) = 0$

1. Gauss-Jacobi Method

2. Gauss-Seidel Method

3. Newton’s Method

4. Homotopy Methods
Discrete-Time Finite-Space Stochastic Games
Separable Game
Nonlinear Equations
Dynamic Game Application

Gauss-Jacobi Method

Last iterate \( x^k = (x_1^k, x_2^k, x_3^k, \ldots, x_{n-1}^k, x_n^k) \)

New iterate \( x^{k+1} \) computed by repeatedly solving one equation in one variable using only values from \( x^k \)

\[
F_1(x_1^{k+1}, x_2^k, x_3^k, \ldots, x_{n-1}^k, x_n^k) = 0
\]
\[
F_2(x_1^k, x_2^{k+1}, x_3^k, \ldots, x_{n-1}^k, x_n^k) = 0
\]
\[
\vdots
\]
\[
F_{n-1}(x_1^k, x_2^k, \ldots, x_{n-2}^k, x_{n-1}^{k+1}, x_n^k) = 0
\]
\[
F_n(x_1^k, x_2^k, \ldots, x_{n-2}^k, x_{n-1}^k, x_n^{k+1}) = 0
\]

Computer storage: Need to store both \( x^k \) and \( x^{k+1} \)

Interpretation as iterated simultaneous best reply
Gauss-Seidel Method

Last iterate $x^k = (x_1^k, x_2^k, x_3^k, \ldots, x_{n-1}^k, x_n^k)$

New iterate $x^{k+1}$ computed by repeatedly solving one equation in one variable and immediately updating the iterate

\[
F_1(x_1^{k+1}, x_2^k, x_3^k, \ldots, x_{n-1}^k, x_n^k) = 0
\]
\[
F_2(x_1^{k+1}, x_2^{k+1}, x_3^k, \ldots, x_{n-1}^k, x_n^k) = 0
\]
\[\vdots\]
\[
F_{n-1}(x_1^{k+1}, x_2^{k+1}, \ldots, x_{n-2}^{k+1}, x_{n-1}^{k+1}, x_n^k) = 0
\]
\[
F_n(x_1^{k+1}, x_2^{k+1}, \ldots, x_{n-2}^{k+1}, x_{n-1}^{k+1}, x_n^{k+1}) = 0
\]

Computer storage: Need to store only one vector

Interpretation as iterated sequential best reply
Solving a Simple Cournot Game

$N$ firms

Firm $i$’s production quantity $q_i$

Total output is $Q = q_1 + q_2 + \ldots + q_N$

Linear inverse demand function, $P(Q) = A - Q$

All firms have identical cost functions $C(q) = \frac{2}{3}cq^{3/2}$

Firm $i$’s profit function $\Pi_i$ is

$$\Pi_i = q_i P(q_i + Q_{-i}) - C(q_i) = q_i (A - (q_i + Q_{-i})) - \frac{2}{3}cq_i^{3/2}$$
First-order Conditions

Necessary and sufficient first-order conditions

\[ A - Q_{-i} - 2q_i - c\sqrt{q_i} = 0 \]

Firm \( i \)'s best reply \( R(Q_{-i}) \) to a production quantity \( Q_{-i} \) of its competitors

\[ q_i = R(Q_{-i}) = \left( \frac{A - Q_i}{2} + \frac{c^2}{8} \right) - \frac{c}{2} \sqrt{\frac{A - Q_{-i}}{2} + \frac{c^2}{16}} \]

Parameter values: \( N = 2 \) firms, \( A = 145 \), \( c = 4 \)
Solving the Cournot Game with Gauss-Jacobi

| $k$ | $q_i^k$     | $\max_i |q_i^k - q_i^{k-1}|$ |
|-----|-------------|--------------------------|
| 0   | 10          | —                        |
| 1   | 52.9471     | 42.9471                  |
| 2   | 34.3113     | 18.6358                  |
| 3   | 42.3318     | 8.02047                  |
| 4   | 38.8656     | 3.46613                  |
| 5   | 40.3611     | 1.49545                  |
| 6   | 39.7154     | 0.645682                 |
| 7   | 39.9941     | 0.278695                 |
| 15  | 39.9102     | 0.000336014              |
| 16  | 39.9100     | 0.000145047              |
| 20  | 39.910075   | 5.036 ($-6$)             |
| 21  | 39.910078   | 2.174 ($-6$)             |
Solving the Cournot Game with Gauss-Seidel

| $k$ | $q_1^k$  | $q_2^k$  | $\max_i |q_i^k - q_i^{k-1}|$ |
|-----|----------|----------|--------------------------|
| 0   | 10       | 10       | —                        |
| 1   | 52.9471  | 34.3113  | 42.9471                  |
| 2   | 42.3318  | 38.8656  | 10.6153                  |
| 3   | 40.3611  | 39.7154  | 1.97068                  |
| 4   | 39.9941  | 39.8738  | 0.366987                 |
| 5   | 39.9257  | 39.9033  | 0.0683762                |
| 6   | 39.913   | 39.9088  | 0.0127409                |
| 7   | 39.9106  | 39.9098  | 0.00237412               |
| 8   | 39.9102  | 39.91    | 0.000442391              |
| 9   | 39.9101  | 39.9101  | 0.0000824347             |
| 10  | 39.9101  | 39.9101  | 0.0000153608             |
| 11  | 39.91008 | 39.91008 | 2.862 (−6)               |
**Gauss-Jacobi with** $N = 4$ **firms blows up**

Cournot equilibrium $q^i = 25$ for all firms

$x^0 = (24, 25, 25, 25)$

| $k$ | $q_1^k$ | $q_2^k = q_3^k = q_4^k$ | $\max_i |q_i^k - q_i^{k-1}|$ |
|-----|---------|--------------------------|-------------------------|
| 1   | 25      | 25.4170                  | 1                       |
| 2   | 24.4793 | 24.6527                  | 0.7642                  |
| 3   | 25.4344 | 25.5068                  | 0.9551                  |
| 4   | 24.3672 | 24.3973                  | 1.1095                  |
| 5   | 25.7543 | 25.7669                  | 1.3871                  |
| 13  | 29.5606 | 29.5606                  | 8.1836                  |
| 14  | 19.3593 | 19.3593                  | 10.201                  |
| 15  | 32.1252 | 32.1252                  | 12.766                  |
| 20  | 4.8197  | 4.8197                   | 37.373                  |
| 21  | 50.9891 | 50.9891                  | 46.169                  |
Newton’s Method

Foundation of Newton’s Method: Taylor’s Theorem

**Theorem.** Suppose the function $F : X \to \mathbb{R}^m$ is continuously differentiable on the open set $X \subset \mathbb{R}^n$ and that the Jacobian function $J_F$ is Lipschitz continuous at $x$ with Lipschitz constant $\gamma^L(x)$. Also suppose that for $s \in \mathbb{R}^n$ the line segment $x + \theta s \in X$ for all $\theta \in [0, 1]$. Then, the linear function $L(s) = F(x) + J_F(x)s$ satisfies

$$\|F(x + s) - L(s)\| \leq \frac{1}{2} \gamma^L(x) \|s\|^2.$$ 

Taylor’s Theorem suggests the approximation

$$F(x + s) \approx L(s) = F(x) + J_F(x)s$$
Newton’s Method in Pure Form

Initial guess $x^0$

Given iterate $x^k$ choose Newton step by calculating a solution $s^k$ to the system of linear equations

$$J_F(x^k) \cdot s^k = -F(x^k)$$

New iterate $x^{k+1} = x^k + s^k$

Excellent local convergence properties
Solving Cournot Game \((N = 4)\) with Newton’s Method

| \(k\) | \(q_i^k\) | \(\max_i |q_i^k - q_i^{k-1}|\) |
|------|---------|-------------------|
| 0    | 10      | —                 |
| 1    | 24.6208 | 14.6208           |
| 2    | 24.9999 | 0.3791            |
| 3    | 25.0000 | 0.000108          |
| 4    | 25.0000 | 8.67(−12)         |
Shortcomings of Newton’s Method

If initial guess $x_0$ is far from a solution Newton’s method may behave erratically; for example, it may diverge or cycle (!)

If $J_F(x^k)$ is singular the Newton step may not be defined

It may be too expensive to compute the Newton step $s^k$ for large systems of equations

The root $x^*$ may be degenerate ($J_F(x^*)$ is singular) and convergence is very slow

Practical variants of Newton-like methods overcome all these issues
Practical Newton-like Method

General idea: Obtain global (!) convergence by combining the Newton step with line-search or trust-region methods from optimization

Merit function monitors progress towards root of $F$

Most widely used merit function is sum of squares

$$M(x) = \frac{1}{2} \|F(x)\|^2 = \frac{1}{2} \sum_{i=1}^{n} F_i^2(x)$$

Any root $x^*$ of $F$ yields global minimum of $M$

Local minimizers with $M(x) > 0$ are not roots of $F$

$$\nabla M(\tilde{x}) = J_F(\tilde{x})^\top F(\tilde{x}) = 0$$

and so $F(\tilde{x}) \neq 0$ implies $J_F(\tilde{x})$ is singular
Line Search Method

Newton step

\[ J_f(x^k) \ s^k = -F(x^k) \]

yields a descent direction of \( M \) as long as \( F(x^k) \neq 0 \)

\[ \left( s^k \right)^\top \nabla M(x^k) = \left( s^k \right)^\top J_F(x^k)^\top F(x^k) = -\|F(x^k)\|^2 < 0 \]

Given step length \( \alpha^k \) the new iterate is

\[ x^{k+1} = x^k + \alpha^k s^k \]
Step length

Inexact line search condition (Armijo condition)

\[ M(x^k + \alpha s^k) \leq M(x^k) + c \alpha \left( \nabla M(x^k) \right)^\top s^k \]

for some constant \( c \in (0, 1) \)

Step length is the largest \( \alpha \) satisfying the inequality

For example, try \( \alpha = 1, \frac{1}{2}, \frac{1}{2^2}, \frac{1}{2^3}, \ldots \)

This approach is not Newton’s method for minimization

No computation or storage of Hessian matrix
Global Convergence Property

**Theorem.** Suppose that $J_F$ is Lipschitz continuous and both $\|J_F(x)\|$ and $\|F(x)\|$ are bounded above in an open neighborhood of the level set $\{x : M(x) \leq M(x^0)\}$. Under some further mild technical conditions the sequence of iterates $x^0, x^1, \ldots, x^k, x^{k+1}, \ldots$ satisfies

$$\left(J_F(x^k)\right)^\top F(x^k) \to 0$$

as $k \to \infty$.

Moreover, if $\|J_F(x^k)\| \geq \delta > 0$ then

$$F(x^k) \to 0.$$
Cournot Game with Learning and Investment

\( N = 2 \) firms in dynamic Cournot competition

State of the game: production cost of two firms

Each period: Firms engage in quantity competition

Stochastic transition to state in next period depends on three forces

Learning: Current output may lead to lower production cost

Investment: Firms can also make investment expenditures to reduce cost

Depreciation: Shock to efficiency may increase cost
Period Game

Firm $i$’s production quantity $q_i$

Total output is $Q = q_1 + q_2$

Linear inverse demand function, $P(Q) = A - Q$

Firms’ production cost functions are quadratic $CP_i(q) = \frac{1}{2} b_i q^2$

Firms’ profit functions are

$$\Pi_1 = q_1 P(q_1 + q_2) - \theta_1 \left( \frac{1}{2} b_1 q_1^2 \right)$$

$$\Pi_2 = q_2 P(q_1 + q_2) - \theta_2 \left( \frac{1}{2} b_2 q_2^2 \right)$$

Efficiency of firm $i$ is given by $\theta_i$
Dynamic Setting

Each firm can be in one of \( S \) states, \( j = 1, 2, \ldots, S \)

State \( j \) of firm \( i \) determines its efficiency level
\[
\theta_i = \frac{\Theta (j-1)}{(S-1)} \text{ for some } \Theta \in (0, 1)
\]

Total range of efficiency levels \([\Theta, 1]\) for any \( S \)

Possible transitions from state \( j \) to states \( j - 1, j, j + 1 \) in next period

Transition probabilities for firm \( i \) depend on
production quantity \( q_i \)
investment effort \( u_i \)
depreciation shock
Transition Probabilities

Probability of successful learning \((j \text{ to } j+1)\), \(\psi(q) = \frac{\kappa q}{1+\kappa q}\)

Probability of successful investment \((j \text{ to } j+1)\), \(\phi(u) = \frac{\alpha u}{1+\alpha u}\)

Cost of investment for firm \(i\), \(CI_i(u) = \frac{1}{S-1} \left( \frac{1}{2} d_i u^2 \right)\)

Probability of depreciation shock, \(\delta\)

These individual probabilities, appropriately combined, yield transition probabilities
Equilibrium Equations

Bellman equation for each firm

First-order condition w.r.t. quantity $q_i$

First-order condition w.r.t. investment $u_i$

Three equations per firm per state

Total of $6S^2$ equations
V1(m1e,m2e) =e= Q1(m1e,m2e)*(1 - Q1(m1e,m2e)/M - Q2(m1e,m2e)/M) - ((b1*power(Q1(m1e,m2e),2))/2. + a1*Q1(m1e,m2e))*theta1(m1e) - ((d1*power(U1(m1e,m2e),2))/2. + c1*U1(m1e,m2e))/(-1 + Nst) + (beta*((1 - 2*delta + power(delta,2) + Q2(m1e,m2e)*(delta*kappa - kappa*power(delta,2) + alpha*kappa*power(delta,2)*U1(m1e,m2e)) + (alpha*delta - alpha*power(delta,2))*U2(m1e,m2e) + Q1(m1e,m2e)*(delta*kappa - kappa*power(delta,2) + power(delta,2)*power(kappa,2)*Q2(m1e,m2e) + alpha*kappa*power(delta,2)*U2(m1e,m2e)) + U1(m1e,m2e)*(alpha*delta - alpha*power(delta,2) +
power(alpha,2)*power(delta,2)*U2(m1e,m2e))*(V1(m1e,m2e) + (delta - power(delta,2) + kappa*power(delta,2)*Q1(m1e,m2e) + alpha*power(delta,2)*U1(m1e,m2e))*V1(m1e,m2e - 1) + ((alpha - 2*alpha*delta + alpha*power(delta,2)) *U2(m1e,m2e) + (delta*power(alpha,2) - power(alpha,2)*power(delta,2))*U1(m1e,m2e)*U2(m1e,m2e) + Q2(m1e,m2e)*(kappa - 2*delta*kappa + kappa*power(delta,2) + (alpha*kappa - alpha*delta*kappa)*U2(m1e,m2e) + U1(m1e,m2e)*(alpha*delta*kappa - alpha*kappa*power(delta,2) + delta*kappa*power(alpha,2)*U2(m1e,m2e)) + Q1(m1e,m2e)*((alpha*delta*kappa -
alpha*kappa*power(delta,2))*U2(m1e,m2e) +
Q2(m1e,m2e)*(delta*power(kappa,2) - power(delta,2)*power(kappa,2)
+ alpha*delta*power(kappa,2)*U2(m1e,m2e))))*V1(m1e,m2e + 1) +
(delta - power(delta,2) + kappa*power(delta,2)*Q2(m1e,m2e) +
alpha*power(delta,2)*U2(m1e,m2e))*V1(m1e - 1,m2e) +
power(delta,2)*V1(m1e - 1,m2e - 1) + ((alpha*delta -
alpha*power(delta,2))*U2(m1e,m2e) +
Q2(m1e,m2e)*(delta*kappa -
kappa*power(delta,2) +
alpha*delta*kappa*U2(m1e,m2e)))*V1(m1e - 1,m2e + 1) + ((alpha*delta*kappa -
alpha*kappa*power(delta,2))*Q2(m1e,m2e)*U1(m1e,m2e) +
U1(m1e,m2e)*(alpha - 2*alpha*delta + alpha*power(delta,2) +
(delta*power(alpha,2) -
GAMS Code IV

\[
\begin{align*}
\text{power}(\alpha,2)\cdot\text{power}(\delta,2)\cdot\text{U2}(m1e,m2e) & + Q1(m1e,m2e)\cdot(kappa - 2\delta\cdot\kappa + \kappa\cdot\text{power}(\delta,2) + \\
Q2(m1e,m2e)\cdot(\delta\cdot\text{power}(\kappa,2) - \text{power}(\delta,2)\cdot\text{power}(\kappa,2) + \alpha\cdot\delta\cdot\text{power}(\kappa,2)\cdot\text{U1}(m1e,m2e)) & + (\alpha\cdot\delta\cdot\kappa - \alpha\cdot\kappa\cdot\text{power}(\delta,2))\cdot\text{U2}(m1e,m2e) + \\
U1(m1e,m2e) & \cdot(\alpha\cdot\kappa - \alpha\cdot\delta\cdot\kappa + \delta\cdot\kappa\cdot\text{power}(\alpha,2)\cdot\text{U2}(m1e,m2e)))\cdot\text{V1}(m1e+1,m2e) + \\
((\alpha\cdot\delta - \alpha\cdot\text{power}(\delta,2))\cdot\text{U1}(m1e,m2e) & + Q1(m1e,m2e)\cdot(\delta\cdot\kappa - \kappa\cdot\text{power}(\delta,2) + \\
\alpha\cdot\delta\cdot\kappa\cdot\text{U1}(m1e,m2e))\cdot\text{V1}(m1e+1,m2e) - 1) + \\
((\text{power}(\alpha,2) - 2\delta\cdot\text{power}(\alpha,2) + \text{power}(\alpha,2)\cdot\text{power}(\delta,2))\cdot\text{U1}(m1e,m2e)\cdot\text{U2}(m1e,m2e) & + \\
\end{align*}
\]
GAMS Code V

Q2(m1e,m2e)*U1(m1e,m2e)*(alpha*kappa - 2*alpha*delta*kappa + alpha*kappa*power(delta,2) + (kappa*power(alpha,2) - delta*kappa*power(alpha,2))*U2(m1e,m2e)) + Q1(m1e,m2e)*((alpha*kappa - 2*alpha*delta*kappa + alpha*kappa*power(delta,2))*U2(m1e,m2e) + (kappa*power(alpha,2) - delta*kappa*power(alpha,2))*U1(m1e,m2e)*U2(m1e,m2e) + Q2(m1e,m2e)*(power(kappa,2) - 2*delta*power(kappa,2) + power(delta,2)*power(kappa,2) + (alpha*power(kappa,2) - alpha*delta*power(kappa,2))*U2(m1e,m2e) + U1(m1e,m2e)*(alpha*power(kappa,2) - alpha*delta*power(kappa,2) + power(alpha,2)*power(kappa,2)*U2(m1e,m2e)))))*V1(m1e + 1,m2e + 1))/((1 + kappa*Q1(m1e,m2e))*(1 + kappa*Q2(m1e,m2e))*(1 + alpha*U1(m1e,m2e)))*(1 + alpha*U2(m1e,m2e));

And that was just one of 6 equations
Results

<table>
<thead>
<tr>
<th>$S$</th>
<th>Var</th>
<th>rows</th>
<th>non-zero</th>
<th>dense (%)</th>
<th>Steps</th>
<th>RT (m:s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>2400</td>
<td>2568</td>
<td>31536</td>
<td>0.48</td>
<td>5</td>
<td>0 : 03</td>
</tr>
<tr>
<td>50</td>
<td>15000</td>
<td>15408</td>
<td>195816</td>
<td>0.08</td>
<td>5</td>
<td>0 : 19</td>
</tr>
<tr>
<td>100</td>
<td>60000</td>
<td>60808</td>
<td>781616</td>
<td>0.02</td>
<td>5</td>
<td>1 : 16</td>
</tr>
<tr>
<td>200</td>
<td>240000</td>
<td>241608</td>
<td>3123216</td>
<td>0.01</td>
<td>5</td>
<td>5 : 12</td>
</tr>
</tbody>
</table>

Convergence for $S = 200$

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Residual</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1.56(+4)</td>
</tr>
<tr>
<td>1</td>
<td>1.06(+1)</td>
</tr>
<tr>
<td>2</td>
<td>1.34</td>
</tr>
<tr>
<td>3</td>
<td>2.04(−2)</td>
</tr>
<tr>
<td>4</td>
<td>1.74(−5)</td>
</tr>
<tr>
<td>5</td>
<td>2.97(−11)</td>
</tr>
</tbody>
</table>
Extensions

Complementarity problems

Continuous time setting