A New Optimization Approach to Maximum Likelihood Estimation of Structural Models

Kenneth L. Judd
Hoover Institution

Che-Lin Su
Northwestern University

March, 2006*

Abstract. Maximum likelihood estimation of structural models is regarded as computationally difficult. This impression is due to a focus on the Nested Fixed-Point approach. We present a direct optimization approach to the problem and show that it is significantly faster than the NFXP approach when applied to the canonical Zurcher bus repair model. The NFXP approach is inappropriate for estimating games since it requires finding all Nash equilibria of a game for each parameter vector considered, a generally intractable computational problem. We reformulate the problem of maximum likelihood estimation of games so into an optimization problem that is qualitatively no more difficult to solve than standard problems. The direct optimization approach is also applicable to other structural estimation problems such as auctions and RBC models, and also to other estimation strategies, such as the methods of moments. It is also easily implemented on standard software.

1. Introduction

Structural estimation of economic models is an important technique in analyzing economic data. However, it is often computationally expensive to implement the most powerful and efficient statistical methods. For example, maximum likelihood estimation is the gold standard in estimation Rust (1987) provided a strategy for doing maximum likelihood estimation of economic models, an approach he dubbed Nested Fixed-Point (NFXP) algorithm. Unfortunately, NFXP is often thought to be computationally infeasible.

While it is clear that NFXP is impractical in many contexts, particularly in the estimation of games, this does not mean that maximum likelihood estimation is intractable. The troublesome feature of NFXP is that it computes a full description of equilibrium at each parameter vector considered while finding the maximum likelihood estimate. We present a direct optimization approach to structural estimation that avoids the necessity of repeatedly solving an economic model; in fact, we only compute the equilibrium associated with the

*THIS IS VERY PRELIMINARY AND INCOMPLETE. DO NOT QUOTE WITHOUT PERMISSION. WE WELCOME COMMENTS.

final estimate. The idea behind the JS[1] approach is simple: choose structural parameters and endogenous economic variables so as to maximize likelihood subject to the constraint that the endogenous economic variables are consistent with the equilibrium defined by the structural parameters. That's it. Nothing more. When formulated in this way, it is clear that all we need to do is to write down the expressions defining the likelihood and the equilibrium constraints, and submit them to one or more of the state-of-the-art constrained optimization programs. With this approach, an economist avoids all the algorithmic details (e.g., BHHH, BFGS, value function iteration, Newton-Kantorovich, policy iteration, etc.) that makes the NFXP and related methods costly to implement.

Of course, NFXP and other procedures are also solving the same constrained optimization problem. However, NFXP insists on proceeding in a "nested" manner. While nesting may sound intuitive to some, it is also clear that nesting is not the usual way we think about constrained optimization problems. Instead, we usually proceed by writing down the Lagrangian, include shadow prices for the constraints, and analyze the resulting set of equations. Also, the numerical analysis literature on constrained optimization does not use nesting except for possibly the very largest problems. The general observation is that nesting is a waste of time because one spends a lot of effort solving the constraints at points far from the solution. Optimization methods avoid this waste.

We show that the JS approach is significantly faster than the NFXP approach when applied to the canonical Zurcher bus repair model. Furthermore, our optimization approach is immediately applicable to other economic models, such as games. The NFXP approach is inappropriate for estimating games since it requires finding all Nash equilibria of a game for each parameter vector considered. The game theory and numerical analysis literatures clearly show that this is an intractable computational problem for all but a few special games. We reformulate the problem of maximum likelihood estimation of games into an optimization problem that is qualitatively no more difficult to solve than standard maximum likelihood problems. Not only will this approach produce the maximum likelihood estimate, but it avoids the difficulties of other methods, such as nested pseudo-maximum likelihood, which implicitly (but without the user's knowledge) makes strong and generally unacceptable equilibrium selection criterion.

The direct optimization approach is also applicable to other structural estimation problems with continuous state spaces, such as auctions and RBC models, and also to other estimation strategies, such as the methods of moments, that can be expressed as solutions

[1] We will tentatively call this method the JS method. This approach fits into some mathematical category, such as bilevel optimization, MPEC, EPEC, or MPCC, but we do not know exactly which one is the best description.

to optimization problems. In particular, the JS approach is also easily implemented on standard software.

2. Current Views

There is much pessimism regarding full maximum likelihood estimation of structural models. For example, Erdem et a. (2004) asserts the following in a section titled "Reducing the Computational Burden of Structural Estimation":

> Estimating structural models can be computationally difficult. For example, dynamic discrete choice models are commonly estimated using the nested fixed point algorithm (see Rust 1994). This requires solving a dynamic programming problem thousands of times during estimation and numerically minimizing a nonlinear likelihood function.....[S]ome recent research ... proposes computationally simple estimators for structural models including auctions, demand in differentiated product markets, dynamic discrete choice and dynamic games. The estimators ... use a two-step approach. In the first step, one flexibly estimates a reduced form for agents' behavior consistent with the underlying structural model. In the second step, the one recovers the structural parameters, by plugging the first-step estimates into the model.....The two-step estimators can have drawbacks. First, there can be a loss of efficiency. The parameters estimated in the second step will depend on a nonparametric first step. If this first step is imprecise, the second step will be poorly estimated. Second, stronger assumptions about unobserved state variables may be required. In a dynamic discrete choice model, accounting for unobserved heterogeneity by using random effects or even a serially correlated, unobserved state variable may be possible using a nested fixed point approach. However, two-step approaches are computationally light, often require minimal parametric assumptions and are likely to make structural models accessible to a larger set of researchers.

The Rust bus repair problem is an example of dynamic discrete choice. Other estimation problems must solve problems over a continuous state space. One such example is auctions. The pessimism about maximum likelihood estimation also appears in the empirical auctions literature. For example Li (2005) considers a basic independent private values model. Suppose n bidders have independent private values drawn from the distribution F. The equilibrium strategy, $s(v)$ is defined by the first-order differential equation

$$s'(v) = (v - s(v))(n-1) f(v) / F(v)$$

for values v above the reservation price, p^0, and below the maximum possible value, $\bar{v}$. We also know that a bidder whose value equals the reservation price will bid his value, implying the initial condition $s\left(p^0\right) = p^0$. The solution is

$$s(v) = v - \frac{1}{F(v)^{n-1}} \int_{p^0}^{v} F(x)^{n-1}\, dx, \text{ if } v \geq p^0 \qquad \text{(bid)}$$

Data from auctions include all or some of the bids. If we parameterize the distribution F, we could find the parameters maximizing the likelihood of the data by solving for s given F. Li states "This is, however, complicated by the high nonlinearity of $s(v)$ in the latent distribution F; whenever the estimation procedure involves computing $s(v)$ within each iteration of an optimization procedure. This is indeed the case for MLE for which $s(v)$ needs to be inverted for each b and y at each iteration. In addition, computation of the jacobian of the bid transformation that is required in the implementation of the MLE makes it more difficult computationally." This pessimism has lead to many alternative procedures, such as Laffont et al. (1995) , Guerre et al. (2000) , Li andVuong (1997) , and Li (2005). These alternatives often involve simulation, a costly and often imprecise process, and are less efficient.

We will argue that this pessimism is mistaken, and that application of nonlinear programming ideas and methods provide an alternative.

Structural estimation in macroeconomics also follows the NFXP approach. For example, Fernandez-Villaverde and Ramirez (2006) use the same nested approach to maximum likelihood estimation of an RBC model. They guess the parameters, solve for the equilibrium, and use that solution in a particle filter simulation to evaluate likelihood.

3. JS Approach

We first formulate the method in generality. Suppose that an economic model has parameters θ. Suppose that equilibrium and optimality imply that the observable economic variables, x, follow a stochastic process parameterized by a finite vector ϕ. The value of ϕ will depend on θ through a set of equilibrium conditions

$$0 = G(\phi, \theta)$$

Denote the likelihood of a data set, x_j, by $L(x, \phi, \theta)$. Therefore, maximum likelihood is the constrained optimization problem

$$\begin{aligned} \max_{\phi,\theta} \quad & L(x, \phi, \theta) \\ \text{s.t.} \quad & 0 = G(\phi, \theta) \end{aligned}$$

This is a very general formulation. We do not require that equilibrium be defined as a solution to a fixed-point equation. We do not need to specify an algorithm for computing ϕ given θ; it is doubtful that we could do better than a good solver would do. Gauss-Jacobi or Gauss-Seidel methods are often used in economics even though they are at best linearly convergent, whereas good solvers are at least superlinearly convergent locally (if not much better) and have better global properties than GJ and GS typically do. Using a direct optimization approach allows one to take advantage of the best available methods from the numerical analysis literature.

This procedure is also similar to the standard methods for solving inverse problems and parameter identification in engineering and science. There is no apparent reason why it should not also do well on economics problems.

4. JS applied to Zurcher

We next apply the JS method to the bus repair model analyzed in Rust (1994). We use the Rust model since it is often used to exposit the ideas of maximum likelihood estimation of dynamic models and to evaluate alternative methods.

The bus repair problem considers the decisions faced by a repairman who must decide whether to perform extensive repairs on a bus when it comes into the bus barn and return it to excellent shape, or to implement less costly activities. The state variable is, x_t, the accumulated mileage at time t since the last engine replacement. Let $c\,(x, \theta_1)$ be the expected per period operating costs, where θ_1 is a vector of parameters of the cost function. Some of these costs are not observed by the econometrician; hence, θ_1 is to be estimated from the observations of when the bus is repaired. Rust assumes that the mileage travelled by a bus during one month is exponentially distributed with parameter θ_2. In each period, Zurcher decides between (i) "normal maintenance" incurring costs $c(x_t, \theta)$, and (ii) replace the bus engine, earning the scrap value P^{scrap}, pay P^{new} for a new engine, and spend $c(0, \theta_1)$ on operating costs. The data is the time series $(x_t, d_t)_{t=1}^{T}$, where x_t is state of the bus examined in period t and d_t is the choice made for that bus. The objective is to find the parameter values that maximize the likelihood of that data..

Zurcher chooses a replacement policy to minimize the expected discounted maintainence costs. d_t is the replacement decision at time $t, d, = 0$ (keep), $d, = 1$ (replace). Given the observed time series data $(x_t, d_t)_{t=1}^{T}$, where x_t is state and d_t is the decision (or choice) associated with x_t, we want to infer the unknown parameter vector θ by maximizing the *likelihood function* $L(\theta)$

$$L(\theta) = \prod_{t=2}^{T} P(d_t|x_t, \theta) p(x_t|x_{t-1}, d_{t-1}, \theta), \tag{1}$$

where the *conditional choice probability*, $P(d|x.\theta)$ is given by the *multinomial formula*

$$P(d|x,\theta) = \frac{exp\{u(x,d,\theta) + \beta EV_\theta(x,d)\}}{\sum_{d' \in D(x)} exp\{u(x,d',\theta) + \beta EV_\theta(x',d)\}}, \tag{2}$$

and the expected value function EV_θ is the fixed point to the contracting mapping $T_\theta(EV_\theta) = EV_\theta$ defined by

$$EV_\theta(x,d) = \int_{x'=0}^{\infty} \log \left[\sum_{d' \in D(x')} exp\{u(x',d',\theta) + \beta EV_\theta(x',d')\} \right] p(dx'|x,d,\theta). \tag{3}$$

We will estimate the unknown parameter vector $\theta = (\theta_1, RC, \theta_3)$ by solving the constrained optimization problem

$$\begin{array}{ll} \max_{\theta, EV_\theta} & L(\theta) = \prod_{t=2}^{T} P(d_t|x_t,\theta) p(x_t|x_{t-1}, d_{t-1}, \theta) \\ \text{s. t.} & EV_\theta(x,d) = T_\theta(EV_\theta(x,d)) \\ & \theta_{30} + \theta_{31} + \theta_{32} = 1 \end{array} \tag{4}$$

4.1. A Numerical Example. We consider an example of Rust's bus engine replacement model. We choose the state space $X = \{1, \ldots, 120\}$ and assume the cost function $c(x,\theta)$ is quadratic, i.e., $c(x,\theta_1) = \theta_{11}x + \theta_{12}x^2$. We do not estimate the discount factor β and let $\beta = 0.95$. The unknown parameters to be estimated are θ_1, RC (scrap value) and θ_3 (the Markov transition probabilities). We simulate a time series data for 1000 time period ($T = 1000$) by using the following parameter values: $\beta = 0.95$, $RC = 1.217$, $\theta_1 = (0.06, 0.0005)$, and $\theta_3 = (0.35, 0.60, 0.05)$. We submitted the problem, coded in AMPL, to the NEOS server. The outputs are given below. The AMPL codes are given in the Appendix.

KNITRO	CPU time (in sec.)		Maj. Iter.	
num. of states	without substitution	substitution	without substitution	substitution
120	0.22	0.23	16	18
240	0.66	0.42	25	21
360	2.14	0.55	29	19
480	9.00	0.98	16	21

A few important points need to be emphasize. First, the problem is solved very quickly. Second, the timing is linear in the number of states if we use a compact way of representing the problem. Third, the likelihood function, the constraints, and their derivatives are evaluated only 16-29 times in this example. In contrast, the Bellman operator in NFXP (the constraints here) is evaluated hundreds of times in NFXP.

4.2. Comparison with Rust Implementation. A computational approach has no value unless it can be implemented using reliable software. Rust describes in detail the software advantages of his NFXP implementation in Gauss. We next run through these points and compare his Gauss implementation with our AMPL implementation.

Ease of use. Rust used Gauss "because: 1. the GAUSS langauge is a high-level symbolic language which enables a nearly 1:1 translation of mathematical formulae into computer code. Matrix operations of GAUSS replace cumbersome do-loops of FORTRAN. 2. GAUSS has built-in linear algebra routines, no links to Lapack needed"

JS: AMPL is also easy to use. All solvers have access to linear algebra routines. AMPL does not have matrix notation, but its approach to matrices, tensors, and indexed sets is very flexible.

Vectorization. Rust: "Efficient use of GAUSS requires the user to "vectorize" a program in a manner very similar to efficient vectorization on large-scale vector supercomputers."

JS: All vectorization is done automatically in AMPL. High performance versions of AMPL are being developed where one uses the same code for desktops and supercomputers.

Optimization Method. Rust: Outer iteration uses BHHH for a while then switches to BFGS, where the user chooses the switch point.

JS: Use solvers far superior to these methods.

Derivatives. Rust: "The NFXP software computes the value of and its derivatives numerically in a subroutine. This implies that we can numerically compute and its derivatives for each trial value encountered in the course of the process of maximizing. In order to do this, we need a very efficient and accurate algorithm for computing the fixed point."

JS: Use true analytic derivatives. This is done automatically by AMPL, and is done efficiently using ideas from automatic differentiation.

Dynamic programming method. Rust: "Inner Fixed Point Algorithm. Contraction mapping fixed point (poly)algorithm. The algorithm combines contraction iterations with Newton-Kantorovich iterations to efficiently compute the functional fixed point." In Rust, contraction iterations are linearly convergent; quadratic convergence is achieved only at final stage.

JS: We use Newton-style methods that are globally faster than contraction mapping ideas. This is particularly important if β is close to 1, representing short, but realistic, time periods.

5. The JS Approach to games

NFXP cannot be used to estimate data from games except for very special cases. Suppose that the game has parameters θ representing payoffs, probabilities, and whatever else is not observed directly by the econometrician. Let σ denote the equilibrium strategy given θ, and that σ is an equilibrium if and only if

$$0 = G(\sigma, \theta)$$

for some function G.[2]. Suppose that likelihood of a data set, x, is $L(x, \phi, \theta)$. Therefore, maximum likelihood is the problem

$$\begin{aligned} \max_{\sigma,\theta} \quad & L(x, \sigma, \theta) \\ \text{s.t.} \quad & 0 = G(\sigma, \theta) \end{aligned}$$

For each θ, NFXP would require one to find all the σ that solves $G(\sigma, \theta)$, compute the likelihood at each equilibrium σ, and report the max. Finding all equilibria is an intractable problem unless one has special structure. Also, the resulting likelihood function will often be discontinuous, if there are multiple equilibria, and possibly nondifferentiable even at points of continuity. Both of these problems will create difficulties for the outer loop in NFXP.

In contrast, JS just sends the problem to good optimization solvers. Multiplicity of equilibria will not create discontinuities or lack of differentiability. Multiple equilibria may produce multiple local solutions, but that is a standard problem in maximum likelinood estimation, and would also be a problem for the NFXP approach.

We are not saying that solving this problem is easy. Of course, it will be more difficult and costly as the size and complexity of the game increases. However, the NFXP approach is impossible to implement for all but the simplest games.

5.1. Comparison with Nested Pseudo-Maximum Likelihood. Aguirregabiria (Victor Aguirregabiria, Economics Letters 84 (2004) 335–340, Pseudo maximum likelihood estimation of structural models involving fixed-point problems) has proposed a nested method for estimating models where the equilibrium is defined by solutions to a fixed-point problem. More precisely, suppose that $y \in Y$ is a vector of discrete random variables over a finite set Y, and suppose that the true distribution of y is P^0. Agguirregabaria assumes P^0 is from a parametric family of probability distributions with parameters y; hence, for some

[2]In some games, equilibrium is defined by a set of complementarity conditions. We do not want to go into the details here, but that case can also be handled by using methods from the MPCC literature.

θ, $P^0 \in P(\theta)$. The distribution $P(\theta)$ is implicitly defined as a solution to the fixed-point problem

$$P(\theta) = \Psi(P(\theta), \theta)$$

for some mapping $\Psi(P, \theta)$. In some cases, such as dynamic programming problems, for any θ there is a unique such $P(\theta)$. However, in many models, such as models of games, there may be many fixed points. Aguirregabiria iterates between guesses for P and guesses for θ. Givne a new guess for θ he obtains a new estimate of P^0 by iterating on the fixed-point mapping Ψ. This detail may cause problems in cases where there are multiple solutions for $P(\theta)$. The key mathematical problem is that Ψ may not be a contraction map in the neighborhood of the true θ and true $P(\theta)$. In fact, in problems with multiple solutions, it is highly unlikely that Ψ is contractive near all equilibria. Therefore, the nested PML method is not consistent because if the data came from a $P(\theta)$ that was not stable under Ψ, there is no chance of converging to the true θ. Also, even when nested PML converges, it will do so at a linear rate at best because it is essentially a Gauss-Seidel scheme.

6. Other applications

We next outline how the JS method could be applied to other estimation problems. While we have not executed these examples yet, the description of how we would proceed might help make clear the key ideas in the JS method.

6.1. Auctions. The JS approach can also be applied to the auction problem. Again, consider the auction problem where n bidders have independent private values drawn from the distribution F, and the equilibrium bidding strategy, $s(v)$ is defined by the first-order differential equation

$$s'(v) = (v - b)(n - 1) f(v) / F(v)$$

for values v above the reservation price, p^0, and below the maximum possible value, $vbar$. While this problem has an easily computed solution, in general one would have to use a numerical method to solve for strategies. The obvious way to proceed is to parameterize the bidding strategy by

$$\hat{S}(v, \alpha) = \sum_{i=1}^{n} \alpha_i \psi_i(v)$$

for a suitable set of n basis functions ψ_i, and have the maximum likelihood procedure choose the coefficients α as well as any structural parameters describing F. The constraint is α is chosen so that $\hat{S}(v, \alpha)$ nearly solves the equilibrium differential equation; this is done by judiciously picking $m \leq n$ nodes and imposing an exact fit for the differential equation on

those nodes. Hence, in the JS method, one would choose a parameterized family for the distribution F, and a flexible approximation scheme for $s(v)$, and choose the parameters for F and s such that the likelihood is maximized and that F and s are consistent with (bid).

6.2. Macroeconometrics. The JS scheme could also be applied in a direct manner. Given the parameterized specification for the law of motion of economic observables, one can compute the likelihood in the same manner as done elsewhere. This applies even to simulation methods such as particle filter. The difference between JS and Fernandez-Villaverde and Ramirez is that they recompute that law of motion many times to evaluate the likelihood and its derivatives, whereas we would use the same specification of the likelihood but just add the equations that define equilibrium. Those equations could be the ones from projection methods or perturbation methods. In fact, almost any algorithm for solving RBC models could be implicitly used in the constraint equations. The key requirement is that the parameterized equilibrium solution must be expressed as a finite set of equations. Simulation methods, such as parameterized expectations, would not be useful in this context since the equilibrium for a parameter vector is not expressed as a manageable set of equations.

7. Conclusion

Maximum likelihood estimation, as well as other methods like methods of moments, indirect inference, partial identification, and calibration are really constrained optimization problems. We suggest that the nonlinear programming approach be applied directly to those problems instead of various two-stage and nested iteration methods that do not take advantage of the best optimization and nonlinear equation solvers. We demonstrate this in the Rust Zurcher bus repair problem, and indicate how to proceed in other cases.

References

[1] John Rust (1987): Optimal Replacement of GMC Bus Engines: An Empirical Model of Harold Zurcher, *Econometrica*, **55**, 999–1033.

[2] John Rust (2000): Nested Fixed Point Algorithm Documentation Manual: Version 6.

[3] Fernández-Villaverde, Jesús, and Juan F. Rubio-Ramírez, "Estimating Macroeconomic Models: A Likelihood Approach" January 24, 2006

[4] Erdem, Tülin, Kannan Srinivasan, Wilfred Amaldoss, Patrick Bajari, Hai Che, Teck Ho, Wes Hutchinson, Michael Katz, Michael Keane, Robert Meyer, and Peter Reiss, "Theory-Driven Choice Models", 2004.

[5] Li, Tong, "Econometrics of first-price auctions with entry and binding reservation prices," Journal of Econometrics 126 (2005) 173–200.

Appendix: AMPL Codes

AMPL Model File

```
# Bus Example:  AMPL model file
#
# An NLP to solve maximum likelihood estimate of bus problem
# K. Judd and C.-L. Su,
# February 21, 2006.
#
   # Define parameters for state space
param N; # number of states
set X := 1..N; # State Space
   # Define data param nT; # number of periods in data
set T := 1..nT; # define vector of time indices
param xt {T}; # state at time t
param dt {T}; # decision at time t
   param beta := 0.95; # diiscount factor - fix beta for usual reasons

   # Define unknown variables to be computed var EV {X}; # Value Function of
each state
var theta1 {1..2}>= 0; # parameter for cost function c(x, theta1) = theta1[1]*x
+ theta1[2]*x^2
var theta3 {1..3}>= 0; # elements in transition matrix
var RC >= 0; # scrap value
var Cost {X} >= 0; # to economize on expressions, create the Cost variable for
c(x)
   # Define objective function
maximize MaxLikelihood_f:  prod {t in 2..nT} log(dt[t]*exp(Cost[xt[t]] - beta*(EV[xt[t]])
- RC - Cost[1]
+ beta*EV[1])/(1+exp(Cost[xt[t]] - beta*(EV[xt[t]]) - RC - Cost[1] + beta*EV[1]))
+ (1-dt[t])*1/(1+exp(Cost[xt[t]] - beta*(EV[xt[t]]) - RC - Cost[1] + beta*EV[1])))
+ sum {t in 2..nT} log(dt[t-1]*(theta3[xt[t]-1+1]) + (1-dt[t-1])*(theta3[xt[t]-xt[t-1]+1]

subject to         # We now list the constraints
   # Bellman equation for states below N-1
FixedPoint_1toNminus2 {i in X: i <= N-2}:  EV[i] = sum {j in 0..2} log(exp(-
Cost[i+j] + beta*(EV[i+j]))+ exp(-RC - Cost[1] + beta*EV[1]))* theta3[j+1];
   # Bellman equation for state N-1
```

```
FixedPoint_Nminus1:  EV[N-1] = log(exp(-Cost[N-1] + beta*(EV[N-1]))+ exp(-RC
- Cost[1] + beta*EV[1]))* theta3[1] + log(exp(-Cost[N] + beta*(EV[N]))+ exp(-RC
- theta1 + beta*EV[1]))* (1-theta3[1]);
   # Bellman equation for state N
FixedPoint_N: EV[N] = log(exp(-Cost[N] + beta*(EV[N]))+ exp(-RC - Cost[1] + beta*EV[1]));

   # probabilities must add to one
Probability:  sum {i in 1..3} theta3[i] = 1;
   # define the cost variable
CostEQ {i in X}:  Cost[i] = sum {j in 1..2} theta1[j]*i^(j);

   ################################
   problem NestedFixedPoint:                    # Define a problem
   MaxLikelihood_f,                                    # the objective function

   EV, RC, theta1, theta3, Cost,             # list the variables
   FixedPoint_1toNminus2,                          # list the constraints
FixedPoint_Nminus1, FixedPoint_N,
Probability, CostEQ;
   ################################
```

AMPL Command File

```
# List true values for our reference
# let RC := 1.2170;
# let theta1 := 0.0851;
# let theta3
# 1 0.35
# 2 0.60
# 3 0.05 ;
# Call solver and give it options
option snopt_options "timing=1 outlev=1";
option snopt_options "iterations_limit=500000 Major_iterations= 50000";
# Initial guesses set at trivial values; probably not good initial guess
let {i in X} EV[i] := 0;
let {i in 1..3} theta3[i] := 1/3;
# Solve command
solve NestedFixedPoint;
# Output commands
option display_round 4;
option display_width 120;
# write the value function
display EV;
# write the structural parameters (remember beta was fixed)
display beta, RC, theta1, theta3;
# write errors in Bellman equations
display {i in X: i <= N-2} FixedPoint_1toNminus2.body[i];
display FixedPoint_Nminus1.body;
display FixedPoint_N.body;
```

AMPL Data File

```
# Choose the size of DP to be approximated
param N := 120;
   # declare length of data
param nT := 1000;
   # declare the value of beta
param beta := 0.95;
   param :  xt dt :=
1            1     0
  .        .        .
  .        .        .
1000     2     0
```

AMPL Output File **********

```
NEOS Server Version 5.0
Solver :  nco:KNITRO:AMPL
Start :  2006-02-26 11:07:57
End :  2006-02-26 11:08:01
Host :  schwinn.mcs.anl.gov (SLOWER THAN OTHERS ON NEOS)
**********
   KNITRO 5.0:  Automatic algorithm selection:  Interior/Direct
   EXIT: LOCALLY OPTIMAL SOLUTION FOUND.
   Final Statistics
----------------
Final objective value = -1.40576716598290e+03
Final feasibility error (abs / rel) = 5.87e-06 / 5.44e-08
Final optimality error (abs / rel) = 5.18e-05 / 9.55e-07
# of iterations (major / minor) = 16 / 16
# of function evaluations = 18
# of gradient evaluations = 17
# of Hessian evaluations = 16
Total program time (secs) = 0.22341 (0.220 CPU time) [WAS 0.8 SECONDS FOR 500
STATES!!!!]

   beta = 0.9500 RC = 1.1442
   :  theta1 theta3 :=
1   0.0545 0.3464
2   0.0034 0.6146
3   .      0.0390 ;
```