Exercise Set 1

Part 1: The Nested Fixedpoint Algorithm

Replicating John Rust (1987)

Zurich Initiative on Computational Economics (ZICE)

January 2017

Bertel Schjerning

Introduction

In this exercise we will be looking at the first real world example of a structural estimation in the Rust's 1987 paper on Engine replacement. The main goal is to replicate the main results from this paper using the Nested Fixed Point Algorith.

This exercise set contains a lot of information. It is meant as a quick, rough introduction to the model and method. Great articles are Rust (1987), Rust (2000), John Rust's manuals on discrete Markov processes, and Aguirregabiria and Mira's (2010) survey on structural discrete choice models in Journal of Econometrics.

The model

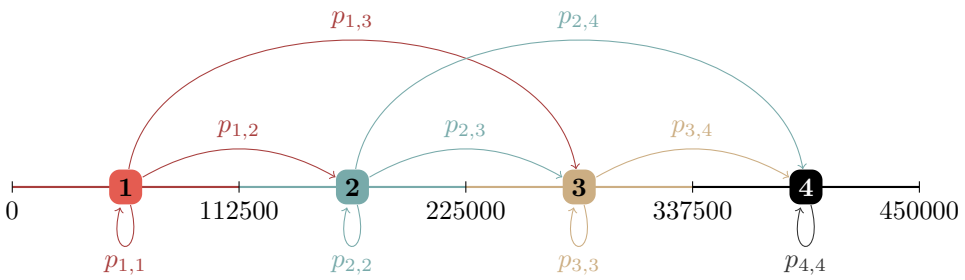
We are going to consider a model of optimal replacement. Madison Metropolitan Bus Company owns a fleet of busses, and their maintainance is managed by Harold Zurcher. Each month, he gets the busses in for maintainance. He registers their odometer readings and makes a binary decision: he can repair the engine of each bus or replace it. The replacement motor is expensive and built from scratch. However, the bus will run as a new one from the factory after a replacement. If he does not replace the engine, he has to do the usual maintainance (repair, oil changes, etc). If he does, he has to pay the replacement cost and do the usual maintainance of a new engine (oil, check everything is running fine, etc). As economists, we would model this using a cost function in two variables: odometer reading and the replacement decision

$$u(x_t, d_t, \theta_1) = \begin{cases} -c(x_t, \theta_c) & \text{if } d_t = 0 \\ -\left[\overline{P} - \underline{P} + c(0, \theta_c)\right] & \text{if } d_t = 1. \end{cases} . \tag{1}$$

Here $RC \equiv \overline{P} - \underline{P}$ is the net replacement cost (scrap value minus engine cost). The binary choice of replacement is d_t, and the state (odometer reading) is x_t. The remaining part of the cost function u is the per period maintainance cost $c(x_t, \theta_c)$. Rust compares a few specifications, and we will use the linear version. The means that

$$c(x_t, \theta_c) = \theta_c \cdot 0.001 \cdot x_t.$$

To characterize the state space we define the set of possible values of x_t. We will assume that it is a bounded interval $[0, 450000]$ measured in miles. To handle the problem on a computer, we will discretize this state space into $N + 1$ nodes, which results in N bins. This means that state 1 means that the odometer is in the interval $[0, \frac{450000}{N}]$, state 2 means that it is in $[\frac{450000}{N}, 2 \cdot \frac{450000}{N}]$, ..., and state N means$[(N-1) \cdot \frac{450000}{N}, N \cdot \frac{450000}{N}]$. Transitions mean moving from state n to m. If we abstract from the decision to replace the engine, the odometer reading can only increase. We model this uncontrolled transition **process** as a finite discrete Markov process. The idea can be seen in the figure below where we illustrate the transition dynamics for $N = 4$. We see that each state covers an interval of readings, not specific values.



Notice a few things. We have written this example such that if s is the current state $s + 2$ is the upper bound on the possible state tomorrow, and state 4 is absorbing (i.e. the bus can't get away from this, unless the engine is replaced). The odometer readings are in principle unbounded and continuous, and these are features that we want to approximate. First of all, the largest odometer reading in the state space should be well larger than the largest observed mileage. Second of all, we need *enough* intervals to capture the difference in transition probabilities. If we keep the four intervals as above we would probably get $p_{j,j}$ very close to 1, and a finer grid would allow for a more flexible process.

Question

Take the above discrete Markov chain, and write the transition matrix.

Controlled Markov Process

Now let us introduce the choice. The previous part was directional (x can only evolve in one direction: forward), and uncontrolled. However, remember that we have a control that enables us to reset x at a cost RC. The question is then: when should we replace the engine?

The Bellman equation

The decision maker chooses a sequence of replacement decisions, d_t, to maximize expected over an infinite horizon discounted by $\beta \in (0, 1)$. In each period, the decision

maker receives a state and choice random ulity which is composed of $u(x_t, d_t, \theta_c)$ from above and an additve choice specific shock, $\epsilon_t(d_t)$, which we assume is iid extreme value type 1 distributed. The latter could represent different unobserved advantages and disadvantages of keeping or replacing.

$$V_\theta(x_t) = \sup_d E\left[\sum_{t=\tau}^{\infty} \beta^{t-\tau}\left(u(x_t, d_t, \theta_c) + \epsilon_t(d_t)\right) \;\middle|\; x_\tau, \epsilon_\tau, \theta_c, \theta_p\right].$$

Here θ_p is the parameter characterizing the transition probabilities. These are markov probabilities in our case, so we simply get a matrix of parameters. The Bellman equation for this is *simply*

$$V_\theta(x, \epsilon) = \max_{d\in\{0,1\}}\left[u(x, d, \theta_c) + \epsilon(d) + \beta\int_y\int_\eta V_\theta(y, \eta)p(\mathrm{d}y, \mathrm{d}\eta|x, \eta, d, \theta_c, \theta_p)\right].$$

Here x and ϵ are the current period variables, and y and η are the corresponding future period variables. Assuming that $\epsilon_t(d_t)$, is conditional independent extreme value type 1 distributed errors, we can rewrite

$$EV_\theta(x, i) = \int_y\left[\log\left\{\sum_{j\in\{0,1\}}\exp\left(u(x, j, \theta_c) + \beta EV_\theta(y, j)\right)\right\}\right]p(\mathrm{d}y|x, i, \theta_p). \qquad (2)$$

The right-hand side is a contraction mapping in EV_θ, and is straight forward to calculate for parameters and EV_θ given. Notice that the discrete state formulation in combination with the Markov nature of the transitions allows us to simply replace the outer integral with a matrix multiplication. To simplify the transition process even further, we simplify the transition probabilities such that

$$\begin{aligned} p_{1,1} &= p_{2,2} = p_{3,3} = \ldots = p_0 \\ p_{1,2} &= p_{2,3} = p_{3,4} = \ldots = p_1 \\ p_{1,3} &= p_{2,4} = p_{3,5} = \ldots = p_{2,} \end{aligned}$$

and so on. So no matter if $x = 1$, $x = 2$ or something different, the probability of staying is the same, the possibility of increasing by one in the state space is the same, and so on. This simplifies the transition matrix and reduces the number of parameters dramatically. Remember each $p_{k,l}$ is a parameter in itself. We could have continuous distributions in stead, parameterized by a few parameters, but this would require integrating over the transition process using quadrature rules or simulation combined with iterpolation of the expected value function.

Noote that the regeneration property of the markov process for x_t implies that for all x we have $EV_\theta(x, 1) = EV_\theta(0, 0)$; i.e., the decision to replace the bus engine regenerates the state of the system to$x = 0$ so that for any $x$the expectation of the value function when $d = 1$ is identical to the expectation of the value function when $x = 0$ and $d = 0$.

This means that we can reduce the number of unknown functions $EV_\theta(x,d)$ in the bus replacement problem from 2 to 1, and hereafter will denote the expected value of not replacing the bus engine, $EV_\theta(x)$. Using this expression it is straightforward to show that the functional equation (2) can be written as:

$$EV_\theta(x,i) = \int_y \left[\log\left\{\exp\left(u(x+y,0,\theta_c) + \beta EV_\theta(x+y)\right) + \exp\left(u(0,1,\theta_c) + \beta EV_\theta(0)\right)\right\}\right] p(\mathrm{d}y|x,i,\theta_p). \tag{3}$$

Since mothly mileage follows a discrete markov process with transition matrix P we can simply write in matrix form

$$EV_\theta(x) = P \times \left[\log\left\{\exp\left(u(x,0,\theta_c) + \beta EV_\theta(x)\right) + \exp\left(u(0,1,\theta_c) + \beta EV_\theta(0)\right)\right\}\right], \tag{4}$$

where the expectation over the mileage process is done by premultyplying a the N-by-N transition matrix to the "log-sum". Here $EV_\theta(x)$ and $u(x,0,\theta_c)$ are N-by-1 vectors (log,Σ and exp are taken element-wise). It is super important that you understand this. This is the functional equation to solve.

With the current model, we simply get logit choice probabilities of d for the probability of replacement

$$P(d=1|x,\theta) = \frac{\exp\left(u(0,1,\theta_c) + \beta EV_\theta(0)\right)}{\exp\left(u(x,0,\theta_c) + \beta EV_\theta(x)\right) + \exp\left(u(0,1,\theta_c) + \beta EV_\theta(0)\right)}. \tag{5}$$

and the keep probability is $P(d=0|x,\theta) = 1 - P(d=1|x,\theta)$

These define the likelihood function together with the transition probabilities (shown for a single bus)

$$l(\{x_\tau, d_\tau\}_{\tau=t}^T | x_\tau, d_\tau, \theta) = \prod_{t=\tau+1}^T P(d_t|x_t,\theta) p(x_t|x_{t-1}, d_{t-1}, \theta_p). \tag{6}$$

Including more observational units simply involve adding a subscript i and multiplying over them as well (since everything is iid conditional on the payoff-relevant states). We are going to take the natural logarithm, and calculate (and optimize) the log-likelihood function instead. This is then simply the sum of log choice probabilities and transition probabilities (at the observed data) over busses and periods. This gives us

$$\begin{aligned} \ell(\mathbf{x}, \mathbf{d} | x_\tau, d_\tau, \theta) &= \sum_t \sum_i \log\left(P(d_{i,t}|x_{i,t},\theta)\right) + \sum_t \sum_i \log\left(p(x_{i,t}|x_{i,t-1}, d_{i,t-1}, \theta_p)\right) & (7) \\ &= \ell^1 + \ell^2 & (8) \end{aligned}$$

Solving the model

So now we know how to solve the model. Solving the model means finding a fixed point in

$$EV_\theta = \Gamma(EV_\theta),$$

where $\Gamma : B \to B$ is a mapping defined by the right hand side of (4). Here B is the Banach space (a particular vector space) of bounded, measurable functions on $[0, \infty)$. Rust showed that this is a contraction mapping, which means that we can apply Γ successively, and as we proceed, our sequence of EV_θ-functions will approach the unique solution to the non-linear equation. We can also use a generalized version of Newton's method to find the zeros of an equivalent problem

$$(I - \Gamma)EV_\theta = 0,$$

where I is the identity operator on B . The identity operator is like a "1" in other spaces. It takes a function as its argument, and returns the same function. Equivalently the "0" on the right hand side is not a zero in the usual sense, but it is the zero element (or zero function) in B. If Γ was linear this would be a straightforward problem, but it is not. Since $(I - \Gamma)$ can be shown to have an invertible (Fréchet) derivative, we use Newton's method. Newton's method for solving non-linear equations is simply to create a Taylor expansion around a point (the point is our current function guess EV_θ^K)

$$0 = (I - \Gamma)EV_\theta^{K+1} \approx (I - \Gamma)EV_\theta^K + (I - \Gamma')(EV_\theta^{K+1} - EV_\theta^K),$$

where $(I-\Gamma')$ is the Fréchet derivative of $(I-\Gamma)$. To calculate Γ' we simply take (2) and differentiate with respect to EV. This turns out to be β times the transition matrix of the controlled process. What is the transition matrix of the controlled process? Well, we don't really develop the theory of Markov decision processes in this course, but it turns out that in this class of dynamic programming problems, the process (d, x) is also Markovian. That is, in our model we assume x followed a discrete Markov process unconditionally, but this then implies that even when we let the agent behave optimally, and therefore affect the evolution of x, we still get a Markov process. The transition matrix for this process in not too difficult to construct. Ask yourself: if $P(0|x_t)$ is the probability of not replacing given x_t, what is the probability of being in the same state tomorrow? It is just $P(0|x_t) \cdot p_0$. What is probability of increasing the state by 1? It is simply $P(0|x_t) \cdot p_1$. Continue this way, and you can fill out the entire controlled transition matrix (see p. 25 in the manual for further explanation). Since $(I - \Gamma')$ is linear and invertable, we get

$$\begin{aligned} -(I - \Gamma)EV_\theta^K &= (I - \Gamma')(EV_\theta^{K+1} - EV_\theta^K) \\ -(I - \Gamma')^{-1}(I - \Gamma)EV_\theta^K &= EV_\theta^{K+1} - EV_\theta^K \\ EV_\theta^{K+1} &= EV_\theta^K - (I - \Gamma')^{-1}(I - \Gamma)EV_\theta^K. \end{aligned} \tag{9}$$

Using this procedure instead of contraction iterations can be dramatically faster than contraction iterations (also called value function iterations or successive approximations). As mentioned before, we need to do this, because the solution enters into the likelihood function. Rust (2000) provides details on further improvements using Werner (1983)'s method.

What	Equation number	Code line(s) in nfxp.m
Cost function (c)	1	240
$\partial c/\partial\theta_c$		239
Bellman operator	4	42
Conditional Choice Probabilities	5	46
log-likelihood	7+8	251 (ℓ^1)and 257 ($\ell^1 + \ell^2$)
$I - \Gamma'$		166
Newton-Kantorovich step	9	167

Table 1: Overview

Optimizing the likelihood

We now know how to solve the model and calculate the likelihood function. In principle we could simply try different values of the parameters, but this is extremely ineffective. Instead we use a quasi-Newton method called BHHH. As you hopefully know, this uses the outer product of the scores as the Hessian-updating formula. This means we do not have to calculate the Hessian, only the first order derivative. This means differentiating the likelihood contributions with respect to a vector of all the parameters: β,θ_C, and all elements of θ_p, and summing over busses and periods. This is a trivial task once we realize that we can use the implicit function theorem to calculate $\partial EV/\partial\theta$

$$\frac{\partial EV}{\partial\theta} = (I - \Gamma')\frac{\partial\Gamma(EV)}{\partial\theta}.$$

We normally assume β constant and leave it out of the vector. What about RC? Well, we simply take Γ evaluated at EV, and differentiate partially wrt. RC and θ_c. This is simply partial differentiation of (4) left-multiplied by $I - \Gamma'$.

Equations and code

It is not always obvious how to go from equations to code. Hopefully, it is possible to implement the above model on your computer, but above I provide a table of equation to line number in nfxp.m. Obviously more lines are used, but they are the main lines, which should resemble the equations.

Question time

Now we have the theory down. Go through the code, try to understand the way run_nfxp.m works. What is called first? Why is it in that order? If there are certain equations your find difficult to translate into code, try looking in the code, and compare to the equations. Every question expects you to run and/or modify run_nfxp.m (script) or nfxp.m (static class).

1. Try changing c to the square root function instead (remember to also edit both dc and cost in lines 242 and 243)

2. Try typing "profile viewer" in Matlab. This opens up the Matlab code profiler. In the "Run this code" field, type "run_nfxp". This gives you a lot of information about the performance of your code.

   (a) Now try changing the optimizer options (optimset), and turn the use of the non-numerical Hessian off . What happens?

   (b) Now also try it with the analytical gradient off, what happens?

3. Why don't we estimate β? Try running the code for different values of β and keep the value of the likelihood function.

4. The problem is, that c and β are very much related (why?). When esitmating with different β, do the changes in the esimtaes of c and/or RC make intuitively sense?

   (a) Can you think of some data/variation, which could allow us to identify β?

5. We use the latest EV guess to start the nfxp.solve-procedure even though we change θ from one likelihood iteration to another. Why do you think we do that?

   (a) What if we started over with $EV = 0$ each iteration? Try that and see what happens with the parameters and the numerical performance. Use the profiler or the output from run_nfxp.m.

6. Try setting the maximum number of miles (odometer reading) to 900. Now the absorbing state is much higher.

   (a) If we adjust the number of grid points as well, so that we have a comparable model (multiply the number of grids by 2), do we get a better fit?

   (b) Try to lower the number of grid points to 175 again. How do the parameters change? Are the changes intuitive?

   (c) What if you change the max to 225 and half the number of grids (hint: what goes wrong?)?

7. Try doing a Monte Carlo study. This is done by changing line 13 in run_nfxp.m to = 1.

   (a) What happens if you change the maximum number of Newton-Kantorovich iterations to 1 (hint: do we get convergence? Do the estimates seem right?). Use the profiler or the output from run_nfxp.m.

(b) What if you set it to 0 (hint: what goes wrong?)?

8. Try focusing on other parts of the Monte Carlo analysis. You could for example compare the standard errors from the two step estimator with the efficient full maximum likelihood version. M-estimator theory tells us that the standard errors should be biased, what do we see?