Problem set 1: Harold Zurcher and NFXP

Replicating John Rust (1987)

Dynamic Structural Econometrics

November 2021,

Fedor Iskhakov, John Rust and Bertel Schjerning

Introduction

The purpose of this exercise is to replicate the results in engine replacement model of Rust (1987) and to investigate the properties of the NFXP algorithm and the resulting MLE estimator. We begin by introducing the general behavioral framework for dynamic discrete choice models in the context of Zurcher's Bus Engine Replacement Problem. For completeness we will introduce the relevant equations. The next subsections closely follows the presentation slides in the course, so you can skip this part and move directly to the questions if you already is familiar with the theory from the lectures and Rust's paper.

The general behavioral framework.

We consider a decision maker, who makes a sequence of optimal discrete actions to maximize expected discounted utility over a infinite horizon

$$V_\theta\left(s_t\right) = \sup_{\Pi} E\left[\sum_{j=0}^{\infty} \beta^j U\left(s_{t+j}, d_{t+j}; \theta_u\right) | s_t, d_t\right]$$

where $\Pi = \left(f_t, f_{t+1}, .., \right), d_t = f_t\left(s_t, \theta\right) \in C\left(x_t\right) = \{1, 2, .., J\}$ is the sequence of state dependent optimal choices, $\beta \in (0,1)$ is the discount factor, $U\left(s_t, d_t; \theta_u\right)$ is a choice and state specific utility function defined above, and E summarizes expectations of future states given s_t and d_t. Here we subscript the value function with θ to emphasize that it is an implicit function of parameters, θ. Since we consider a stationary infinite horizon problem, we will skip time-subscripts and denote future values of state s with a prime, s'.

By *Bellman Principle of Optimality* we can the express the the maximization problem recursively so that the value function $V_\theta(s)$ constitutes the solution of the following functional (Bellman) equation

$$V(x, \varepsilon) \equiv T(V)(x, \varepsilon) = \max_{d \in C(x)} \Big\{ u(x, \varepsilon, d) + \beta E\Big[V(x', \varepsilon')\Big| x, \varepsilon, d\Big]\Big\}$$

where expectations are taken over the next period values of state $s' = (x', \varepsilon')$ given it's *controlled* motion rule, $p(s' \mid s, d)$

$$E\Big[V(x', \varepsilon')\Big|x, \varepsilon, d\Big] = \int_X \int_\Omega V(x', \varepsilon')p(x', \varepsilon'|x, \varepsilon, d)dx'd\varepsilon'$$

where $\varepsilon = (\varepsilon(1), \dots, \varepsilon(J)) \in \mathbb{R}^J$. Here we divide the state variables $s_t = (x_t, \varepsilon_t)$ into observed states, x_t and unobserved states ε_t. When taking the model to the data, we assume that the researcher only observe ε_t, but the decision make observes the current values of both x_t and ε_t before making his decision.

Solving the model amounts to computing the fixed point V such that $T(V) = V$. Without further assumption, this can be a daunting task since x may be continuous and ε is continuous and J-dimensional. Therefore,

- $V(x, \varepsilon)$ is high dimensional
- Evaluating E may require high dimensional integration
- Evaluating $V(x', \varepsilon')$ may require high dimensional interpolation/approximation
- $V(x, \varepsilon)$ is non-differentiable due to the max operator.

Rust's assumptions

Rust make a number of simplifying assumptions that makes the problem much more tractable

1. *Additive separability in preferences (**AS**)*:

$$U(s_t, d) = u(x_t, d; \theta_u) + \varepsilon_t(d)$$

2. *Conditional independence (**CI**)*:
   State variables, $s_t = (x_t, \varepsilon_t)$ obeys a (*conditional independent*) controlled Markov process with probability density

$$p(x_{t+1}, \varepsilon_{t+1}|x_t, \varepsilon_t, d, \theta_q, \theta_p) = q(\varepsilon_{t+1}|x_{t+1}, \theta_q)p(x_{t+1}|x_t, d, \theta_p)$$

3. *Extreme value Type I (EV1) distribution of ε (**EV**)*
   Each of the choice specific state variables, $\varepsilon_t(d)$ are assumed to be iid. *extreme value distributed* with CDF

$$F(\varepsilon_t(d); \mu, \lambda) = \exp(-\exp(-(\varepsilon_t(d) - \mu)/\lambda)) \text{ for } \varepsilon_t(d) \in \mathbb{R}$$

   with $\theta_q = (\mu, \lambda) = (0, 1)$.

Together, these assumptions greatly simplifies the DP problem. Conditional independence also allow us to reformulate Bellman equation on reduced state space, additive separability allows us to separate out the deterministic part of choice specific values, and the assumption that ε_t is extreme value distributed allows us to compute the expectation of maximum in closed form.

Under CI and AS we have

$$V(x,\varepsilon) = \max_{d\in C(x)} \left\{ u(x,d) + \varepsilon(d) + \beta \int_X \int_\Omega V(x',\varepsilon')p(x'|x,d)q(\varepsilon'|x')dx'd\varepsilon' \right\}$$

Here is is worth noting that additive separability of ε_t in utility allow us to separate out the deterministic part of choice specific value $v(x,d)$, so that

$$V(x',\varepsilon') = \max_{d\in C} \left\{ v(x',d) + \varepsilon'(d) \right\}$$

where

$$v(x,d) = u(x,d) + \beta E\Big[V(x',\varepsilon')\Big|x,d\Big]$$

If we let $EV(x,d) = E\Big[V(x',\varepsilon')\Big|x,d\Big]$ denote the expected value function, we can now express the Bellman equation in *expected value function* space

$$EV(x,d) = \Gamma(EV)(x,d) \equiv \int_X \int_\Omega [V(x',\varepsilon')q(\varepsilon'|x')d\varepsilon']\, p(x'|x,d)dx'$$

where

$$V(x',\varepsilon') = \max_{d'\in C(x)} [u(x',d') + \beta EV(x',d') + \varepsilon'(d')]$$

Note that $EV(x,d)$ only depends on x and d and thus is much lower dimensional than $V(x,\varepsilon)$ since it does not depend on ε. Rust showed that Γ is a separate *contraction mapping* with unique fixed point EV, i.e. $\|\Gamma(EV) - \Gamma(W)\| \leq \beta \|EV - W\|$. This ensures global convergence of VFI.

Alternatively, we can express the Bellman equation in *integrated value function* space. Let $\bar{V}(x) = E\Big[V(x,\varepsilon)\Big|x\Big]$ denote the *integrated value function*

$$\bar{V}(x) = \bar{\Gamma}(\bar{V})(x) \equiv \int_\Omega V(x,\varepsilon)q(\varepsilon|x)d\varepsilon$$

where

$$V(x,\varepsilon) = \max_{d\in C(x)} [u(x,d) + \varepsilon(d) + \beta \int_X \bar{V}(x')p(x'|x,d)dx']$$

Note that $\bar{V}(x)$ is even lower dimensional since does not depend on ε *and d*. Again, it can be shown that $\bar{\Gamma}$ is a *contraction mapping* with unique fixed point $\bar{V}$, i.e. $\left\|\bar{\Gamma}\left(\bar{V}\right) - \bar{\Gamma}(W)\right\| \leq \beta \left\|\bar{V} - W\right\|$ which ensures global convergence using the method of successive approximations.

We can express *expectation of maximum* using properties of EV1 distribution (assumption EV) so that the integrated value function $\bar{V}(x)$, can be expressed as "the log-sum"

$$\bar{V}(x) = E\left[\max_{d\in\{1,\dots,J\}}\{v(x,d)+\lambda\varepsilon(d)\} \mid x\right] = \lambda \log \sum_{j=1}^{J} \exp(v(x,d)/\lambda)$$

Moreover, the *conditional choice probability*, $P(x,d)$, i.e. the probability that the decision maker choose alternative d conditonal on state x, has closed form logit expression

$$P(d \mid x) = E\left[\mathbb{1}\left\{d = \arg\max_{j\in\{1,\dots,J\}}\{v(x,j)+\lambda\varepsilon(j)\}\right\} \mid x\right] \quad = \frac{\exp(v(x,d)/\lambda)}{\sum_{j=1}^{J}\exp(v(x,j)/\lambda)}$$

The benefits of this is HUGE. We avoid J dimensional numerical integration over ε to compute the conditional expectation. Moreover, $P(d \mid x)$, $\bar{V}(x)$ and $EV(x,d)$ are smooth functions so that we can use derivative based algorithms to solve and estimate the model.

The DP problem under AS, CI and EV

Putting all this together we obtain close forms for the conditional choice probability, expectation of the maximum, and a much reduced state space. The conditional choice probabilities (CCPs) are given by

$$P\left(d|x,\theta\right) = \frac{\exp\left\{u\left(x,d,\theta_u\right)+\beta EV_\theta\left(x,d\right)\right\}}{\sum_{j\in C(y)}\exp\left\{u\left(x,j,\theta_u\right)+\beta EV_\theta\left(x,j\right)\right\}}$$

where the expected value function can be found as the unique fixed point to the contraction mapping Γ_θ, defined by

$$\begin{aligned} EV_\theta\left(x,d\right) &= \Gamma_\theta(EV_\theta)\left(x,d\right) \\ &= \int_y \ln\left[\sum_{d'\in D(y)}\exp\left[u(y,d';\theta_u)+\beta EV_\theta\left(y,d'\right)\right]\right] p\left(dy|x,d,\theta_2\right) \end{aligned} \tag{1}$$

Here we have used the subscript θ to emphasize that the Bellman operators, Γ_θ depends on the parameters. In turn, the fixed point, EV_θ, and the resulting CCPs, $P\left(d|x,\theta\right)$ are implicit functions of the parameters we wish to estimate. As before, we can also express the bellman equation in integrated value function space

$$\begin{aligned} \bar{V}_\theta\left(x\right) &= \bar{\Gamma}_\theta(\bar{V}_\theta)\left(x\right) \\ &= \ln\left[\sum_{d'\in D(y)}\exp\left[u(y,d';\theta_u)+\beta\int_y \bar{V}_\theta\left(y'\right) p\left(dy|x,d',\theta_p\right)\right]\right] \end{aligned} \tag{2}$$

Discretizing the state space

What if the state variable x continuous? How to deal with *continuous* state? We either could make a grid over x and interpolate the expected or integrated value function between state points, or we discretize the state variable into n bins indexed with $k = 1, ..., n$: $\hat{X} = \{\hat{x}_1, ..., \hat{x}_n\}$. Discretizing the state variable implies that $\hat{x}$ is now controlled discrete Markov chain with a choice specific transition probability matrix $P(d)$ where the i, j'th element has probability $p_{i,j} = p(\hat{x_j}'|\hat{x}_i, d, \theta_p)$. The discretized Bellman equation becomes

$$EV_\theta(\hat{x}, d) = \hat{\Gamma}_\theta(EV_\theta)(\hat{x}, d) = \sum_j^n \ln\left[\sum_{d'\in D(y)} \exp[u(x', d'; \theta_1) + \beta EV_\theta(x', d')]\right] p(x'_j|\hat{x}, d, \theta_p)$$

When x is discrete we can also express the Bellman equation in matrix form, so that the choice specific expected value function can be found as fixed point on the discretized Bellman operator

$$EV(d) = \hat{\Gamma}(EV) = \Pi(d) \ln\left[\sum_{d'\in D(y)} \exp[u(d') + \beta EV(d')]\right]$$

where $EV(d) = [EV(1, d), .., EV(n, d)]$ and $u(d) = [u(1, d), .., u(n, d)]$ are $n \times 1$ vectors and $\Pi(d)$ is the $n \times n$ state transition matrix conditional on decision d.

Alternatively, we can find the integrated value function as fixed point on the Bellman operator

$$\bar{V} = \hat{\Gamma}(\bar{V}) = \ln\left[\sum_{d'\in D(y)} \exp[u(d') + \beta\Pi(d')\bar{V})]\right]$$

where $\bar{V} = [\bar{V}(1), .., \bar{V}(n)]$ is a $n \times 1$ vector and the vector of conditional expected value function can be obtained subsequently by a simple matrix product $EV(d) = \Pi(d)\bar{V}$.

Harold Zurcher

We now return to Harold Zurcher's replacement problem considered in Rust (1987).[1] Each month, Zurcher gets the buses in for maintenance. He registers their odometer readings, x_t, and makes a binary decision, $d_t \in C(x_t) = \{0, 1\}$ whether to perform ordinary maintenance ($d_t = 0$) or overhaul/engine replacement ($d_t = 1$). Before making his decision, Harold Zurcher observes the state of the bus $s_t = (x_t, \varepsilon_t)$, where x_t is mileage at time t since last engine overhaul/replacement and $\varepsilon_t = [\varepsilon_t(d_t = 0), \varepsilon_t(d_t = 1)]$ is

[1] Harold Zurcher is the superintendent of maintenance in Madison Metropolitan Bus Company that owns a fleet of GMC buses.

a decision specific state variable that for example includes idiosyncratic shocks to the investment decision.

We assume Zurcher minimizes the discounted expected cost of replacing engines and maintaining busses. His current period state and decision dependent pay-off is thus the negative of the per period maintenance and replacement cost

$$U(x_t, \varepsilon_t, d_t; \theta_u) = \ u(x_t, d_t, \theta_u) + \varepsilon_t(d_t) = \begin{cases} -RC - c(0, \theta_c) + \varepsilon_t(1) & \text{if } d_t = 1 \\ -c(x_t, \theta_c) + \varepsilon_t(0) & \text{if } d_t = 0 \end{cases} \tag{3}$$

where $RC \equiv \overline{P} - \underline{P}$ is the net replacement cost (scrap value minus engine cost) and θ_c is the parameters of the operating and maintenance cost function, $c(x_t, \theta_c)$. Together the deterministic part of the pay off function, $u(x_t, d_t, \theta_u)$ has parameters $\theta_u = \{RC, \theta_c\}$. Rust compares a few specifications, and we will for example use the linear specification

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

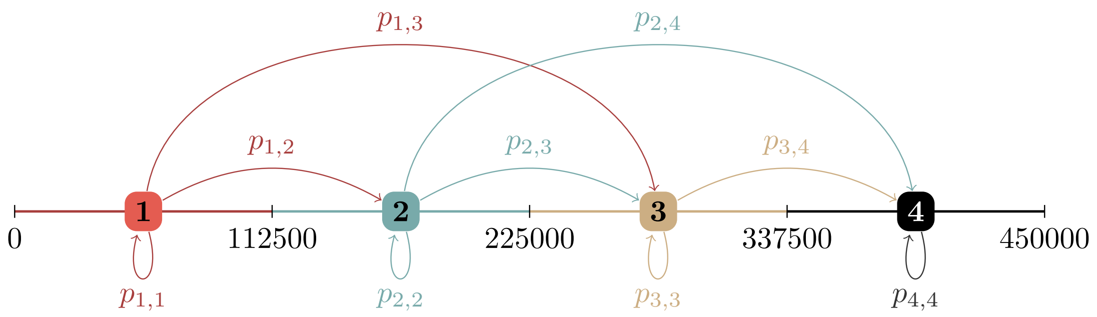
As before, the state variables $s_t = (x_t, \varepsilon_t)$ are divided into observed states, x_t and unobserved states ε_t and obeys a (*conditional independent*) Markov process. The researcher only observe ε_t, but Harold Zurcher observes the current values of both mileage x_t and ε_t before making his replacement decision. Rust assumes that the unobserved states, ε_t, are i.i.d. extreme value distributed and that the observed state, x_t (mileage since last replacement) has the conditional density

$$p(x_{t+1}|x_t, d_t, \theta_2) = \begin{cases} g(x_{t+1} - 0, \theta_p) & \text{if } d_t = 1 \\ g(x_{t+1} - x_t, \theta_p) & \text{if } d_t = 0 \end{cases} \tag{4}$$

In Rust's terminology this is called a *regenerative random walk* to reflect that the mileage since last replacement regenerates to $x_t = 0$ if $d_t = 1$ (engine is replaced), but otherwise continues from its current value x_t if $d_t = 0$ (normal maintenance).

To characterize the state space we define the set of possible values of x_t. Rust assumed that x is a bounded interval $[0, 450000]$ measured in miles and discretized 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$[(n-1) \cdot \frac{450000}{n}, n \cdot \frac{450000}{n}]$.. So here transitions mean moving from state $x = \hat{x}_i$ to $\hat{x}_j$.

If we first 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. We see that each state covers an interval of readings, not specific values.



We have written this example such that if x is the current state $x + 2$ 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, but also more parameters to estimate.

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

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?

Rust discretized the range of travelled miles into $n = 175$ bins, indexed with i so that mileage takes on values $\hat{X} = \{\hat{x}_1, ..., \hat{x}_n\}$ with $\hat{x}_1 = 0$ and the mileage transition probability for $j = 1, ..., \bar{n}$ is

$$p(x'|\hat{x}_k, d, \theta_p) = \begin{cases} Pr\{x' = \hat{x}_{k+j}|\theta_3\} = \theta_{pj} \text{ if } d = 0 \\ Pr\{x' = \hat{x}_{1+j}|\theta_3\} = \theta_{pj} \text{ if } d = 1 \end{cases}$$

Hence, mileage in the next period x' can move up at most $\bar{n}$ grid points where $\bar{n}$ is determined by the distribution of mileage.

We can the express the bellman equation in terms of the discretized choice-specific expected value function for $\hat{x} \in \hat{X}$

$$\begin{aligned} EV_\theta(\hat{x}, d) &= \hat{\Gamma}_\theta(EV_\theta)(\hat{x}, d) \\ &= \sum_j^{\bar{n}} \ln\left[\sum_{d' \in D(y)} \exp[u(x', d'; \theta_1) + \beta EV_\theta(x', d')]\right] p(x'|\hat{x}, d, \theta_p) \end{aligned}$$

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 simple sum, that can be implemented using matrix multiplication. To simplify the transition process even further, we simplify the transition probabilities such that

$$\begin{aligned}\pi_{1,1} &= \pi_{2,2} = \pi_{3,3} = \ldots = \pi_0 \\ \pi_{1,2} &= \pi_{2,3} = \pi_{3,4} = \ldots = \pi_1 \\ \pi_{1,3} &= \pi_{2,4} = \pi_{3,5} = \ldots = \pi_2,\end{aligned}$$

and so on. The corresponding transition matrices are given by

$$\Pi(d=0)_{nxn} = \begin{pmatrix} \pi_0 & \pi_1 & \pi_2 & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & \pi_0 & \pi_1 & \pi_2 & 0 & \cdot & \cdot & 0 \\ 0 & 0 & \pi_0 & \pi_1 & \pi_2 & 0 & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ 0 & \cdot & \cdot & 0 & \pi_0 & \pi_1 & \pi_2 & 0 \\ 0 & \cdot & \cdot & \cdot & 0 & \pi_0 & \pi_1 & \pi_2 \\ 0 & \cdot & \cdot & \cdot & \cdot & 0 & \pi_0 & 1-\pi_0 \\ 0 & \cdot & \cdot & \cdot & \cdot & \cdot & 0 & 1 \end{pmatrix}$$

and

$$\Pi(d=1)_{nxn} = \begin{pmatrix} \pi_0 & \pi_1 & \pi_2 & 0 & \cdot & \cdot & \cdot & 0 \\ \pi_0 & \pi_1 & \pi_2 & 0 & \cdot & \cdot & \cdot & 0 \\ \pi_0 & \pi_1 & \pi_2 & 0 & \cdot & \cdot & \cdot & 0 \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \pi_0 & \pi_1 & \pi_2 & 0 & \cdot & \cdot & \cdot & 0 \\ \pi_0 & \pi_1 & \pi_2 & 0 & \cdot & \cdot & \cdot & 0 \\ \pi_0 & \pi_1 & \pi_2 & 0 & \cdot & \cdot & \cdot & 0 \end{pmatrix}$$

So no matter if $x = \hat{x}_1$, $x = \hat{x}_2$ 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. If the transition matrices were fully unrestricted each $\pi_{k,l}$ is a parameter in itself. We could have continuous distributions in stead, parameterized by a few parameters, but this would instead increase the difficulty of integrating over the transition process. With the specification above, Rust describe the transition process for mileage by only $\bar{n}$ parameters.

Due to the regenerative transitions in the Zurcher problem, the EV(d) simplifies even further

$$\begin{aligned} EV_\theta \equiv EV_\theta(0) &\equiv (EV_\theta(1,0), EV_\theta(2,0), \ldots, EV_\theta(N,0))' \\ EV_\theta(1) &\equiv (EV_\theta(1,0), EV_\theta(1,0), \ldots, EV_\theta(1,0))' \end{aligned}$$

This is because if $d = 1$ we regenerate the state process to a *new* engine, and this means that we should consider the corresponding element of the expected value function. In other words, $EV_\theta(x,1) = EV_\theta(1,0)$ for all x since changing the motor today, means entering tomorrow with a brand new motor, which is the first element in the state space.

$$EV_\theta = P(0) \times \left[\log \left\{ \sum_{j \in \{0,1\}} \exp\left(u(j,\theta_c) + \beta EV_\theta(j)\right) \right\} \right], \tag{5}$$

where the matrix multiplication is over an N-by-N matrix and a N-by-1 vector (log,Σ and exp are taken element-wise).

Solving the model

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

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

where $\Gamma : B \to B$ is a mapping defined by the right hand side of (5). Here B is the Banach space (a particular vector space) of bounded, measurable functions on $[0, \infty)$.

Successive approximations

Rust showed that this is a contraction mapping, which means that we can apply Γ successively

$$EV_\theta^{K+1} = \Gamma(EV_\theta^K) \tag{6}$$

and as we proceed, our sequence of EV_θ-functions will approach the unique solution to the non-linear equation, $EV_\theta = \Gamma(EV_\theta)$.

Newton-Kantorovich iterations

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 $(I-\Gamma)$. Since $(I-\Gamma')$ is linear and invertible, we get

$$\begin{aligned} -(I-\Gamma)EV_\theta^K &= (I-\Gamma')(EV_\theta^{K+1} - EV_\theta^K) \\ -(I-\Gamma')^{-1}(I-\Gamma)EV_\theta^K &= EV_\theta^{K+1} - EV_\theta^K \\ EV_\theta^{K+1} &= EV_\theta^K - (I-\Gamma')^{-1}(I-\Gamma)EV_\theta^K. \end{aligned} \tag{7}$$

We call the iterative scheme implied by 7 the *Newton-Kantorovich iteration*.

Using Newton-Kantorovich iterations instead of contraction iterations can be dramatically faster than contraction iterations (also called value function iterations or successive approximations) which converges linearly and as opposed to Newton-Kantorovich iteration that converges quadratically.[2]

Derivative of Bellman operator

To calculate Γ' we simply take (1) 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 the NFXP manual for further explanation).

[2] Rust (2000) provides details on further improvements using Werner (1983)'s method.

Structural Estimation

Given data on replacement decisions and mileage since last replacement $(d_{i,t}, x_{i,t})$, $t = 1, ..., T_i$ and $i = 1, ..., N$, we want to estimate this model with Maximum Likelihood using the Nested Fixed Point Algorithm. We can obtain the maximum likelihood estimator (ML) by maximizing the sample likelihood

$$\hat{\theta}_{MLE} = \arg\max_{\theta} L(\theta, EV_{\theta})) = \sum_{i=1}^{N} \ell_i^f(\theta, EV_{\theta})$$

where the parameters to be estimated are $\theta = (\theta_u, \theta_p)$, while we usually fix the parameters (β, θ_q) due to lack of identification.

Under the conditional independence assumption (CI) the log-likelihood function for these data ℓ^f takes the particular simple form

$$\ell_i^f(\theta, EV_{\theta}) = \sum_{t=2}^{T_i} log(P(d_{i,t}|x_{i,t}, \theta)) + \sum_{t=2}^{T_i} log\left(p(x_{i,t}|x_{i,t-1}, d_{i,t-1}, \theta_p)\right) \tag{8}$$

where $P\left(d_t|x_t, \theta\right)$ is the conditional choice probability given the observable state variable, x_t. As shown above we can express the CCPs using the logit formula

$$P(d|x, \theta) = \frac{\exp\{u(x, d, \theta_u) + \beta EV_{\theta}(x, d)\}}{\sum_{d' \in \{0,1\}}\{u(x, d', \theta_u) + \beta EV_{\theta}(x, d')\}} \tag{9}$$

where

$$\begin{aligned} EV_{\theta}(x, d) &= \Gamma_{\theta}(EV_{\theta})(x, d) \\ &= \int_y \ln\left[\sum_{d' \in \{0,1\}} \exp[u(y, d'; \theta_u) + \beta EV_{\theta}(y, d')]\right] p(dy|x, d, \theta_p) \end{aligned} \tag{10}$$

So for each evaluation of the likelihood function we need to solve the fixed point equation.

The likelihood contribution that arise for the transition probability is simply the transition probabilities π_j corresponding to the observed monthly mileage, i.e. π_j if mileage moves up j bins. While $p\left(x_t|x_{t-1}, d_{t-1}, \theta_p\right)$ can be expressed in closed form, we need to solve the dynamic program to obtain the likelihood for the discrete choices, because the choice probability $P\left(d_t|x_t, \theta\right)$ depends on the expected value function.

Optimizing the likelihood

As mentioned before, we need to do this, because the solution enters into the likelihood function. 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_u, and all elements of θ_p, and summing over buses 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 (5) left-multiplied by $I - \Gamma'$.

The Nested Fixed Point Algorithm (NFXP)

Since the contraction mapping Γ always has a unique fixed point, the constraint $EV = \Gamma_\theta(EV)$ implies that the fixed point EV_θ is an *implicit function* of θ. Hence, NFXP solves the *unconstrained* optimization problem

$$\max_{\theta} L(\theta, EV_\theta)$$

We can thereby summarize NFXP by the nested algorithm

Outer loop (Hill-climbing algorithm): $L(\theta, EV_\theta)$ is maximized w.r.t. θ

- Quasi-Newton algorithm: Usually BHHH, BFGS or a combination.
- Each evaluation of $L(\theta, EV_\theta)$ requires solution of EV_θ

Inner loop (fixed point algorithm): The implicit function EV_θ defined by $EV_\theta = \Gamma(EV_\theta)$ is solved using a combination of:

- Successive Approximations (SA)
- Newton-Kantorovich (NK) Iterations

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 we 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.

Table 1: Code overview

What	Equation number	function name
Pay-off function (u)	3	zurcher.u
$\partial u / \partial \theta$	You can do it :)	zurcher.u
Bellman operator, Γ	1 or 5	zurcher.bellman_ev
Derivative of Bellman operator, Γ'	You can do it :)	zurcher.bellman_ev
Conditional Choice Probabilities	9	zurcher.bellman_ev
log-likelihood	8	zurcher.ll
Successive approximations algorithm	6	dpsolver.sa
Newton-Kantorovich algorithm	7	dpsolver.nk

Questions

Now we have the theory down. We will first consider how to solve for the fixed point. This is done in the example code run_fxp.m. Go through the code, try to understand the way run_fxp.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.

1. First solve the model using successive approximations for different values of β. How β does the rate of convergence? What is the relative size of the error bound, $tol = ||EV_\theta^{K+1} - \Gamma(EV_\theta^K)||$ between two successive approximations? How is that related to β

2. Solve the model using Newton-Kantorovich iterations. What method is preferred?

3. Solve the model both using both the Bellman equations formulated in expected and integrated value function space (see 1 and 2 respectively). Do you get the same results? Which method is faster? Would this change in applications with continuous states and many alternatives?
   Hint: use the switch mp.bellman_type which can take the character values 'ev' or 'iv' for expected and integrated values respectively. (See zurcher.setup for help)

Now move on to run_busdata.m, which is the code that use NFXP to estimate the engine replacement model using Rust's bus data.

4. Try to replicate the parameter estimates presented in Table 9 and 10 in Rust 1987. Hint: modify mp.n to change the number of grid-points. Try to do this with the formulations of bellman equation in both expected and integrated value function space (i.e. using 1 and 2). Does the results change?

5. Try changing the cost function, c to the square root function instead of a linear function (remember to edit both the utility and it's derivative in zurcher.u)

6. Try another specification that has more parameters, for example a quadratic or s specification that allows RC and parameters in the maintenance cost to be bus type specific.

7. Try typing "profile viewer" in Matlab. This opens up the Matlab code profiler. In the "Run this code" field, type "run_busdata". 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?

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

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

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

10. 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.

11. 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?)?

12. Try doing a Monte Carlo study. Use run_mc_nfxp.m for this.

    (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.

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

13. 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?