Structural Estimation via NFXP: A Step-by-Step Guide to the Rust (1987) Bus Engine Model

Companion Document to `rust_nfxp.jl`

Kyumin Kim

February 22, 2026

Contents

Introduction: Structural vs. Reduced-Form

In empirical economics, there are broadly two approaches to econometric research: reduced-form estimation and structural estimation.

Reduced-form econometrics generally revolves around causal inference, often with quasi-experimental designs. It aims to estimate how a treatment or policy A causally affects outcome B. It is a rapidly growing field with increasingly sophisticated methods for identification—the exploding difference-in-differences (DiD) literature, instrumental variables (IV), regression discontinuity designs (RDD), and so on. Reduced-form work does not explicitly model agents' optimization problems or equilibrium conditions. Instead, it focuses on cleanly identifying causal effects: "If X changes, what happens to Y?"—with as little bias as possible, relying less on heavy economic theory for modeling behavior. Using a fisheries context as an example, a reduced-form question might be: "Did installing marine protected areas (MPAs) improve fish harvest in this region, comparing before and after?"

Structural econometrics goes beyond this by explicitly modeling agents' decision-making processes. It starts with a behavioral model grounded in economic theory and then estimates the model's parameters from data. The approach begins with the question "Why do people make these choices?" and builds a theoretical structure around it.

The workflow looks like this. First, you build a model based on economic theory—for example, consumers maximize utility, or fishers maximize expected profit while choosing where and how much to harvest. Second, you derive equilibrium conditions or first-order conditions from this model. Third, you estimate the "deep parameters" embedded in these conditions—preference parameters, technology parameters, cost parameters, and so on.

I will admit: I did not understand the concept of deep parameters when I was in my first year of the PhD program. One of my dear econometrics professors, Aaron Smith—who is now an endowed professor at UC Berkeley—made this concept much clearer during his office hours. He told me: "Hey Kyumin, just imagine parameters that have specific names like 'elasticities'—names derived from economic theory, parameters tied to theory-based economic behavior." That clicked for me. Structural econometrics is thus often used for industrial organization (IO)-style problems involving very explicit decision-making by agents. In empirical setups, random utility models (RUM) implemented via logit or mixed logit specifications are common in this type of structural modeling. A structural question in the fisheries context would be: "If we increased or decreased the size of MPAs, what would the effect of this size change be on harvest?"

There are many well-established guides and textbooks for reduced-form empirical methods—great books such as *Causal Inference: The Mixtape* by Scott Cunningham and *Mostly Harmless Econometrics* by Joshua Angrist and Jörn-Steffen Pischke. However, I have often noticed that practical, example-driven guides for structural econometric methods are comparatively rare. So I decided to write about one of the core topics in structural estimation.

My First Structural Econometrics Journey: The Agony That Started with Dynamic Discrete Choice

My very first deep dive into structural modeling was during my dynamic optimization course. For the midterm, I had to learn dynamic discrete choice models for the first time, while simultaneously learning the computational algorithms needed to solve them—specifically, the Nested Fixed-Point (NFXP) method.

The midterm problem was Harold Zurcher's bus engine replacement problem—a classic example

of a dynamic discrete choice model in structural econometrics. The problem concerns whether a bus engine should be replaced or not: the optimal decision problem faced by Harold Zurcher, who was responsible for bus engine replacement in the real world. (The economist John Rust later collected Zurcher's engine replacement records and applied his structural model to this real-world data.)

Very frankly, I really struggled with building this model in Julia. Even understanding the concepts and the algorithm was intimidating. I recently took a look at the midterm code I wrote about four years ago. Now I feel like: "Hmm, now I understand this much more clearly... I think I can explain what is going on to other people."

So I decided to present this representative dynamic discrete choice problem for people encountering this class of models for the first time. In this bus engine replacement problem, the "deep structural parameters" we seek to estimate are the **replacement cost** (RC) and the **marginal engine maintenance cost** (θ_1). I will elaborate on these below.

What Is This Document?

This document walks through the Rust (1987) bus engine replacement problem and explains, step by step, how the Nested Fixed-Point (NFXP) algorithm estimates the structural parameters of the model. It is written as a companion to the Julia implementation in `rust_nfxp.jl`.

The goal is not just to explain the code, but to build intuition for *why* each step is necessary and how the pieces connect.

The Problem: What Are We Trying to Do?

Harold Zurcher is a bus maintenance superintendent. Every period, he looks at a bus engine's accumulated mileage and makes a binary decision:

- $a = 0$: **Keep** the current engine (pay maintenance costs that increase with mileage)
- $a = 1$: **Replace** the engine (pay a large one-time cost, mileage resets to zero)

We observe Zurcher's decisions across many buses and many time periods. Our goal is to **recover the cost parameters** that rationalize his behavior:

- RC: the replacement cost (how expensive is a new engine?)
- θ_1: the marginal maintenance cost (how much does mileage hurt?)

Why "Structural"?

A reduced-form approach would simply regress replacement decisions on mileage. Structural estimation goes further: we model Zurcher as a rational, forward-looking agent who solves a dynamic optimization problem. This lets us ask counterfactual questions like "What would happen if engine prices doubled?"

The Model

State and Actions

The state variable is mileage, discretized into bins $x \in \{0, 1, 2, \ldots, n-1\}$ where $n = 90$ in our implementation. The action is binary: $a \in \{0, 1\}$.

Flow Utility

Each period, Zurcher receives utility (really, negative cost):

$$u(x, a = 0; \theta) = -\theta_1 \cdot x \qquad \text{(keep: cost grows with mileage)} \tag{1}$$

$$u(x, a = 1; \theta) = -RC \qquad \text{(replace: fixed cost, mileage resets)} \tag{2}$$

This is the `flow_utility()` function in the code.

Transition Probabilities

After the action, mileage changes stochastically. If Zurcher keeps the engine ($a = 0$), mileage increases by $\Delta x \in \{0, 1, 2\}$ with probabilities $(\theta_{31}, \theta_{32}, 1 - \theta_{31} - \theta_{32})$. If he replaces ($a = 1$), mileage resets to 0 and then increases by the same random increment.

This is captured by the transition matrix F built in `make_transition_matrix()`.

Two Kinds of Probability

The model has two distinct probabilities:

1. $p(x'|x, a; \theta_3)$: **Transition probability**. How the environment evolves. This is about nature, not about Zurcher's choices. We estimate it in Step 1 by counting frequencies.
2. $\mathrm{P}(a|x; \theta)$: **Choice probability**. How Zurcher behaves. This is what the model predicts and what we match to data in Step 2.

Do not confuse them. $p(x'|x, a)$ is an *input* to the model; $\mathrm{P}(a|x)$ is an *output*.

The Error Term: Why Gumbel?

Zurcher's actual utility includes an unobserved shock ε:

$$\tilde{u}(x, a) = u(x, a; \theta) + \varepsilon(a) \tag{3}$$

where $\varepsilon(0)$ and $\varepsilon(1)$ are i.i.d. Type I Extreme Value (Gumbel) distributed.

Rust (1987) chose this distribution because of a remarkable analytical result (McFadden, 1978):

$$\mathbb{E}_\varepsilon \Big[\max_a \big(W_a + \varepsilon(a) \big) \Big] = \log \sum_a \exp(W_a) + \gamma \tag{4}$$

where γ is Euler's constant. The Gumbel assumption turns the max operator into a smooth log-$\sum$-exp, which makes the Bellman equation differentiable and the choice probabilities logit.

The Bellman Equation

Zurcher is forward-looking: he considers not just today's cost but the entire future stream of costs. Define the **expected value function**:

$$\mathrm{EV}(x, a) \equiv \sum_{x'} \mathbb{E}_{\varepsilon'} V(x', \varepsilon') \cdot p(x'|x, a) \tag{5}$$

Applying the Gumbel result, the expectation over ε' yields:

$$\mathbb{E}_{\varepsilon'} V(x', \varepsilon') = \log \left[\sum_{a'} \exp\big(u(x', a'; \theta) + \beta \cdot \mathrm{EV}(x', a') \big) \right] \tag{6}$$

Substituting back:

$$\boxed{\mathrm{EV}(x,a) = \sum_{x'} \log\left[\sum_{a'} \exp\big(u(x',a';\theta) + \beta \cdot \mathrm{EV}(x',a')\big)\right] \cdot p(x'|x,a;\theta_3)} \tag{7}$$

This is a **fixed-point equation**: EV appears on both sides. Solving it is the job of the inner loop.

Where Did ε Go?

The ε terms do not appear in equation (7). They were *integrated out* by taking the expectation. The Gumbel distribution's special property is that this integration yields a clean closed-form (log-$\sum$-exp) rather than requiring numerical simulation.

Choice Probabilities

Given EV, the probability that Zurcher replaces is logit:

$$\mathrm{P}(a=1|x) = \frac{\exp\big(v_1(x)\big)}{\exp\big(v_0(x)\big) + \exp\big(v_1(x)\big)} = \frac{1}{1+\exp\big(v_0(x) - v_1(x)\big)} \tag{8}$$

where:

$$v_0(x) = u(x,0) + \beta \sum_{x'} \mathrm{EV}(x') \cdot F(x,x') \qquad \text{(keep)} \tag{9}$$

$$v_1(x) = u(x,1) + \beta \sum_{x'} \mathrm{EV}(x') \cdot F(1,x') \qquad \text{(replace, restart from 0)} \tag{10}$$

This is the `choice_prob()` function.

Why Is the Bellman Equation Necessary?

In a static discrete choice model, you would simply compute:

$$\mathrm{P}(a=1|x) = \frac{\exp(u_1(x))}{\exp(u_0(x)) + \exp(u_1(x))}$$

No Bellman equation needed. Plug in the flow utilities, get choice probabilities, compute the likelihood, done.

But Zurcher is **forward-looking**. At mileage 30, deciding whether to replace depends on what happens at mileage 40, 50, 60, and so on into the infinite future. The value of keeping depends on the value of keeping next period, which depends on the period after that, ad infinitum. This infinite recursion is captured by the Bellman equation.

The Bellman equation is not an object of interest in itself. It is a **computational bottleneck** that stands between the parameters θ and the likelihood. We need to solve it to evaluate "how good is this θ?"

The NFXP Algorithm

NFXP (Rust, 1987) estimates (θ_1, RC) by maximum likelihood, with the Bellman equation solved as an intermediate step. The name means "Nested Fixed-Point" because fixed-point iteration (solving the Bellman equation) is *nested inside* the optimization loop.

Overview

Step 1 (Pre-estimation): Estimate θ_3 from data by frequency counting.

Step 2 (NFXP): Fix $\hat{\theta}_3$ and solve:

$$\max_{RC,\theta_1} \log L(RC, \theta_1 \mid \text{data}, \hat{\theta}_3)$$

where each likelihood evaluation requires solving the Bellman equation.

Step 1: Transition Probability Estimation

This is the simple part. Look at all periods where Zurcher did *not* replace, and count how much mileage increased:

$$\hat{\theta}_{3k} = \frac{\#\{\text{observations where } \Delta x = k \text{ and } a = 0\}}{\#\{\text{observations where } a = 0\}}, \quad k = 0, 1 \tag{11}$$

This works because of Rust's **conditional independence** assumption: the transition probabilities are independent of the unobserved ε. This is the `estimate_transition()` function.

Step 2: The Nested Loops

7.3.1 Outer Loop (Optimizer)

An optimizer (Nelder-Mead in our code) searches over parameter values (RC, θ_1) to maximize the log-likelihood. At each candidate:

7.3.2 Inner Loop (Bellman Solver)

Given (RC, θ_1), solve the Bellman equation (7) by **contraction mapping iteration**:

1. Initialize $\text{EV}^{(0)} = \mathbf{0}$ (a vector of zeros, one per state)
2. Compute $\text{EV}^{(k+1)}$ from $\text{EV}^{(k)}$ using equation (7)
3. Check convergence: $\|\text{EV}^{(k+1)} - \text{EV}^{(k)}\|_\infty < 10^{-12}$
4. If converged, return EV^*. Otherwise, go to step 2.

The Inner Loop Always Converges

The Bellman operator is a contraction mapping (due to $\beta < 1$), so this iteration is guaranteed to converge regardless of the parameter values. Giving it $(RC = 1000, \theta_1 = 0.001)$ will still converge, it will just find the fixed point for those (wrong) parameters.
Convergence of the inner loop does **not** mean we found the right parameters. It only means "given these parameters, this is what rational behavior looks like." Finding the right parameters is the outer loop's job.

This is the `solve_EV()` function.

7.3.3 From EV* to Likelihood

Once the inner loop converges:

1. Compute choice probabilities $\mathrm{P}(a|x)$ from EV* using equation (8)
2. Evaluate the log-likelihood:

$$\log L = \sum_{i=1}^{N_{\text{buses}}} \sum_{t=1}^{T} \log \mathrm{P}(a_{it}|x_{it}; \theta, \mathrm{EV}^*) \tag{12}$$

3. The optimizer receives this scalar and decides how to update (RC, θ_1)

This is the `neg_log_likelihood()` function.

7.3.4 The Full Loop

```
Outer Loop (optimizer)
  iter 1:                RC=5.0, θ1=0.10
                         → [inner loop:  solve Bellman] → logL = -15432.7
  iter 2:                RC=6.0, θ1=0.10
                         → [inner loop:  solve Bellman] → logL = -14891.2
  iter 3:                RC=6.5, θ1=0.15
                         → [inner loop:  solve Bellman] → logL = -14203.5
  ...
  iter 48:               RC=9.99, θ1=0.300
                         → [inner loop:  solve Bellman] → logL = -12044.1
  iter 49:               logL no longer improving → STOP
```

Why Solve the Bellman Equation Every Time?

When the optimizer changes (RC, θ_1), the entire value function changes. A higher θ_1 means maintenance is more expensive at every mileage level, which changes the optimal replacement threshold, which changes the future value of every state. The old EV* is no longer valid. We must solve from scratch.

Inference: Standard Errors

After finding $\hat{\theta} = (\widehat{RC}, \hat{\theta}_1)$, we need standard errors to assess statistical significance. The asymptotic distribution of MLE is:

$$\sqrt{N}\,(\hat{\theta} - \theta_0) \xrightarrow{d} \mathcal{N}\big(0,\, \mathcal{I}(\theta_0)^{-1}\big) \tag{13}$$

where $\mathcal{I}(\theta) = -\mathbb{E}\left[\frac{\partial^2 \log L}{\partial\theta\,\partial\theta'}\right]$ is the information matrix.

In practice, we compute:

$$\widehat{\mathrm{Var}}(\hat{\theta}) = H^{-1}, \quad \text{where} \quad H = \left.\frac{\partial^2(-\log L)}{\partial\theta\,\partial\theta'}\right|_{\theta=\hat{\theta}} \tag{14}$$

The standard errors are $\mathrm{SE}(\hat{\theta}_j) = \sqrt{(H^{-1})_{jj}}$.

In the code, we use `ForwardDiff.hessian()` to compute H via automatic differentiation. This is more accurate than numerical finite differences and requires no manual derivation.

The t-statistic and p-value for testing $H_0 : \theta_j = 0$ are:

$$t = \frac{\hat{\theta}_j}{\text{SE}(\hat{\theta}_j)}, \quad p = 2[1 - \Phi(|t|)] \tag{15}$$

This is the `compute_se()` and `print_inference()` functions.

Optimal Policy

Once we have $\hat{\theta}$, we can characterize Zurcher's optimal behavior. The replacement probability $\text{P}(a = 1|x)$ at each mileage state reveals the decision rule.

Because of the Gumbel error terms, the policy is *probabilistic* rather than a sharp threshold. However, in practice you will observe:

- At low mileage: P(replace) ≈ 0 (maintenance is cheap, no reason to replace)
- At some threshold: P(replace) rises steeply
- At high mileage: P(replace) ≈ 1 (maintenance is so expensive that replacement is almost certain)

The threshold depends on the ratio RC/θ_1: a higher RC (expensive replacement) delays replacement; a higher θ_1 (expensive maintenance) accelerates it.

NFXP vs. MPEC

NFXP solves the Bellman equation to completion at every parameter guess. An alternative approach, MPEC (Su and Judd, 2012), reformulates the problem as:

$$\max_{\theta, \text{EV}} \ \log L(\theta, \text{EV}) \quad \text{subject to} \quad \text{EV} = \Gamma(\text{EV}; \theta) \tag{16}$$

where Γ is the Bellman operator. Instead of solving the Bellman equation in an inner loop, MPEC treats EV as a decision variable and the Bellman equation as a constraint.

	NFXP	**MPEC**
Strategy	Solve Bellman exactly, then evaluate likelihood	Optimize θ and EV simultaneously
Inner loop?	Yes (full convergence each time)	No
Solver	Any optimizer + custom solver	Constrained optimizer (e.g., Ipopt)
Speed	Can be slow if state space is large	Often faster for large problems
Coding	Modular (solver + optimizer separate)	More complex (gradients, sparsity)

The key insight of MPEC is: intermediate iterations of the outer loop do not need an exactly solved Bellman equation. Only the *final* solution must satisfy both optimality of θ and the Bellman equation. MPEC exploits this by adjusting θ and EV simultaneously.

Code–Theory Mapping

Theory	Code	Section
Flow utility $u(x, a; \theta)$	`flow_utility()`	Section 1
Transition matrix F	`make_transition_matrix()`	Section 1
Bellman eq. fixed point	`solve_EV()`	Section 2
Choice probability $\mathrm{P}(a\|x)$	`choice_prob()`	Section 2
Data generation (DGP)	`generate_data()`	Section 3
Transition estimation $\hat{\theta}_3$	`estimate_transition()`	Section 4a
Log-likelihood $\log L$	`neg_log_likelihood()`	Section 4a
Outer loop (MLE)	`run_nfxp()`	Section 4a
Standard errors H^{-1}	`compute_se()`	Section 4b
Inference table	`print_inference()`	Section 4b

References

- Angrist, J. and Pischke, J.-S. (2009). *Mostly Harmless Econometrics: An Empiricist's Companion*. Princeton University Press.
- Aguirregabiria, V. and Mira, P. (2010). "Dynamic Discrete Choice Structural Models: A Survey." *Journal of Econometrics*, 156(1), 38–67.
- Cunningham, S. (2021). *Causal Inference: The Mixtape*. Yale University Press.
- McFadden, D. (1978). "Modelling the Choice of Residential Location." In *Spatial Interaction Theory and Residential Location*, ed. A. Karlqvist et al.
- Rust, J. (1987). "Optimal Replacement of GMC Bus Engines: An Empirical Model of Harold Zurcher." *Econometrica*, 55(5), 999–1033.
- Su, C.-L. and Judd, K. (2012). "Constrained Optimization Approaches to Estimation of Structural Models." *Econometrica*, 80(5), 2213–2230.