Replication Package for Blevins (2025)

Blevins, J. R. (2025). Identification and Estimation of Continuous Time Dynamic Discrete Choice Games.

Overview

The code in this replication package implements structural estimation and Monte Carlo experiments for two continuous-time dynamic discrete choice models analyzed by Blevins (2025).

The first model (`mc1p` directory) is a continuous-time single-agent renewal model inspired by Rust (1987) that analyzes optimal bus engine replacement decisions. There are two main programs for this model: `rustct` carries out the structural estimation of the model with real data while `mc1p` carries out the Monte Carlo experiments.

The second model (`mcnp` directory) is a continuous-time quality ladder model based on Ericson and Pakes (1995) that analyzes oligopoly dynamics with entry, investment, and exit decisions.

Data Availability

The Monte Carlo experiments in this paper do not involve analysis of external data. The data used are generated via simulation in the code itself.

The NFXP data used in the empirical example is from Rust (1987). It is included in this replication package and is freely available under the MIT License.

NFXP Data

The data for the Nested Fixed Point Algorithm (NFXP) (i.e., bus engine replacement data) used in the single-agent renewal model (`mc1p`) consists of odometer readings and dates of bus engine replacements for 162 buses in the fleet of the Madison Metropolitan Bus Company that were in operation during the period December 1974 to May 1985. This data was originally collected and used by Rust (1987). The data files represent different bus models/vintages and are provided in ASCII format. Each file is named by bus model (e.g., `d309.asc` for Davidson model 309 buses, `g870.asc` for Grumman model 870 buses).

The original data and detailed documentation can be downloaded from John Rust's website at https://editorialexpress.com/jrust/nfxp.html or from the Zenodo repository at https://doi.org/10.5281/zenodo.3374587. The data files are also included in this replication package for convenience.

Data files: `mc1p/data/*.asc`

Computational Requirements

Software Requirements

- Fortran 2008 or later compiler:
  - GNU Fortran (gfortran)
  - Intel Fortran Compiler
- LAPACK and BLAS libraries (or Intel MKL as alternative)
- OpenMP support (included with above compilers)
- GNU Make

The code has been tested with both GNU Fortran and Intel Fortran. The Makefiles will use GNU Fortran by default. To use the Intel Fortran compiler, prefix the usual `make` command with `SYSTEM=intel` as in `SYSTEM=intel make`.

Portions of the code use bash scripting, which may require Linux or macOS.

Controlled Randomness

Random seeds for data generation are set within the Monte Carlo experiment control programs: `mc1p/mc1p.f90` (line 130) and `mcnp/mcnp.f90` (line 128). Differences in compilers, library versions, optimization settings, hardware, etc. may still produce slightly different results.

Hardware Requirements

It is possible to run the single-agent renewal model estimation and Monte Carlo experiments on a standard 2025 desktop computer. The quality ladder model Monte Carlo experiments are more computationally intensive and were carried out on a high performance computing cluster, but with more time it is also possible to run these experiments on a standard desktop computer.

The `rustct` structural estimation code for the single agent model has been tested and easily completes in a few minutes on a 2023 MacBook Pro.

The `mc1p` single agent Monte Carlo experiments and timing exercises have were tested on a 2019 Mac Pro workstation (2.5 GHz 28-Core Intel Xeon W processor), where the full set of experiments completed in around 5 minutes.

The `mcnp` quality ladder model Monte Carlo experiments involve repeatedly solving for the equilibrium of a complex dynamic game with many firms. These simulations are computationally intensive and were carried out on a high-performance computing cluster over the course of several days. We used 1 parallel core for the 2 firm model, 2 parallel cores for the 4 firm model, 4 parallel cores for the 6 firm model, and 8 parallel cores for the 8 firm model.

However, with more time, it is possible to run these experiments on a standard desktop computer. For our largest experiment with 8 firms, a single Monte Carlo replication takes about 5 hours on a 2019 Mac Pro. Completing all 100 replications reported in the paper would therefore take around 21 days. On the other hand, for the smaller 4 firm model all 100 replications can be completed in around 8 hours.

Due to the increased computational requirements of the `mcnp` experiments reported in the paper, we also provide a simple example Monte Carlo experiment with fewer firms and quality states that can be run on a standard laptop computer in a few minutes. Further instructions are provided in the section titled "Simple Example Game" below.

Although this specification is simpler, we note that running this code involves fully computing the equilibrium of the game and computing the matrix exponential for a large matrix to calculate the log likelihood function for each trial parameter value during the optimization. This becomes feasible due to the computational benefits of continuous time games as well as the use of high performance, vectorized Fortran code.

Description of Code

Single-Agent Renewal Model (`mc1p/`)

Main Programs:

- `rustct.f90`: Structural estimation using bus engine replacement data.
- `mc1p.f90`: Monte Carlo experiment control program.

Core Modules:

- `rust_model.f90`: Single-agent continuous-time dynamic discrete choice model.
- `rust_data.f90`: Data loading and processing for bus fleet data.
- `dataset.f90`: Underlying data structure.

Libraries:

- `lbfgsb/`: L-BFGS-B optimization routines.
- `expokit/`: Expokit for dense Matrix exponential calculations.

Control Files:

- `control/mc-<delta>-<nm>.ctl`: Monte Carlo experiment control files.

Quality Ladder Model (`mcnp/`)

Main Program:

- `mcnp.f90`: Monte Carlo experiment control program.

Core Modules:

- `model.f90`: Continuous-time dynamic oligopoly (quality ladder) model with entry, exit, and investment.
- `encoding.f90`: Efficient state space encoding/decoding.
- `dataset.f90`: Data structure for storing simulated observations.
- `sparse.f90`: Sparse matrix operations for computational efficiency.

Libraries:

- `lbfgsb/`: L-BFGS-B optimization routines.

Control Files:

- `control/example.ctl`: A small-scale example control file for testing on a standard laptop or desktop computer.
- `control/mc-<nn>-<delta>.ctl`: Monte Carlo simulation control files for different numbers of firms and data sampling specifications.
- `control/time-<nn>.ctl`: Timing control files for different numbers of firms.

Instructions to Replicators

Setup

1. Ensure you have one of the supported Fortran compilers installed.
2. Ensure BLAS and LAPACK libraries are available (or Intel MKL).
3. Clone or extract this repository to a local directory.

Quick Start

For users with a working Fortran development system looking to quickly run the code:

```
# Build and estimate single-agent model
cd mc1p && make && ./rustct

# Run a single-agent Monte Carlo experiment
./mc1p control/mc-1.00-200.ctl

# Run the simple example quality ladder model Monte Carlo experiment
cd ../mcnp && make
OMP_NUM_THREADS=1 ./mcnp control/example.ctl
```

Installing Dependencies

On Debian/Ubuntu Linux:

```
sudo apt-get install gfortran liblapack-dev libblas-dev
```

On CentOS/RHEL Linux:

```
sudo yum install gcc-gfortran lapack-devel blas-devel
```

On macOS with Homebrew:

```
brew install gcc
```

macOS provides BLAS and LAPACK through the Accelerate framework.

Building the Code

For the single-agent model (`mc1p`), the following commands will build the `rustct` and `mc1p` executables:

```
cd mc1p
make clean
make
```

For the quality ladder model (`mcnp`), the following commands will build the `mcnp` executable:

```
cd mcnp
make clean
make
```

Single Agent Renewal Model Empirical Results (Tables 1 and 2)

Table 1: The `rustct` program reports the sample characteristics of the Rust (1987) data when the raw data is loaded.

```
cd mc1p
./rustct
```

Table 2: Upon completion, the `rustct` program reports the parameter estimates for the default heterogeneous lambda model (last column of Table 2).

This will:

- Load the bus group data.
- Report the sample summary statistics reported in Table 1.
- Estimate parameters of the default heterogeneous lambda model.
- Output results for this specification in LaTeX format.

Running the `rustct` program for each specification will produce the estimates shown in each respective column of Table 2.

To change model specifications, edit lines 38-46 in `rustct.f90`. Uncomment *only* the `model_variant` and `np_theta` for the model specification you wish to use (heterogeneous, homogeneous, or ABBE specification), then recompile the `rustct` executable. Specific settings used to produce Table 2 are given below.

Fixed $\lambda = 1$ (Table 2, Column 1):

```
integer, parameter :: model_variant = RUST_MODEL_ABBE
integer, parameter :: np_theta = np_abbe
```

Variable λ (Table 2, Column 2):

```
integer, parameter :: model_variant = RUST_MODEL_HOMOGENEOUS
integer, parameter :: np_theta = np_restr
```

Heterogeneous λ (Table 2, Column 3):

```
integer, parameter :: model_variant = RUST_MODEL_HETEROGENEOUS
integer, parameter :: np_theta = np_full
```

Single-Agent Monte Carlo Experiments (Table 3)

To execute all of the single-agent model Monte Carlo experiments reported in Table 3, you can use the `mc.sh` shell script:

```
cd mc1p
./mc.sh
```

This script runs all specifications reported in the paper and prints the results as a complete LaTeX table.

Alternatively, you can run a single specification by using the appropriate control file. For example:

```
cd mc1p
./mc1p control/mc-1.00-3200.ctl
```

Individual control files are named as `mc-<delta>-<nm>.ctl` where `delta` is the observation time interval (`0.00` for continuous time) and `nm` is the number of markets simulated.

Quality Ladder Model Timing and Monte Carlo Experiments (Tables 4 and 5)

The quality ladder model experiments are carried out by the `mcnp` program. This program also requires a control file to be specified on the command line:

```
./mcnp control-file.ctl
```

All control files are provided in the `control/` directory.

Table 4:: To produce the model size and computational time results, use the timing control files:

```
./mcnp control/time-02.ctl
./mcnp control/time-04.ctl
./mcnp control/time-06.ctl
./mcnp control/time-08.ctl
...
./mcnp control/time-30.ctl
```

The number of states is reported as `nk` along with the elapsed time to compute the value function.

Table 5: To produce the Monte Carlo results for a given number of firms `nn` and discrete time observation interval `delta`, use the provided control files named `mc-<nn>-<delta>.ctl`. For example, the experiment with `nn = 4` firms and `delta = 1.0` was produced by the following command:

```
./mcnp control/mc-04-1.0.ctl
```

To produce all results reported in the table, you will need to run each specification:

```
./mcnp control/mc-02-0.0.ctl
./mcnp control/mc-02-1.0.ctl
./mcnp control/mc-04-0.0.ctl
./mcnp control/mc-04-1.0.ctl
./mcnp control/mc-06-0.0.ctl
./mcnp control/mc-06-1.0.ctl
./mcnp control/mc-08-0.0.ctl
./mcnp control/mc-08-1.0.ctl
```

Parallel Execution

Both models support OpenMP parallelization. `mc1p` executes individual Monte Carlo replications in parallel across threads while `mcnp` uses multiple threads with a replication to compute the value function, log likelihood function, etc. To set the number of threads, set the `OMP_NUM_THREADS` environment variable.

```
export OMP_NUM_THREADS=8
```

Simple Example Game

As mentioned above, the Monte Carlo experiments for the dynamic game are very computationally intensive we recommended running them on a high performance compute server with a large number of cores. However, we also provide a simplified control file for execution on a standard laptop or desktop computer:

```
export OMP_NUM_THREADS=1
./mcnp control/example.ctl
```

This is a small-scale example with 2 firms, 4 quality states, and a higher continuous-time discount factor, which reduces the computational time required. This example, which runs 25 Monte Carlo replications, completed in under 5 minutes on a 2019 Mac Pro workstation. See `mcnp/logs/example-1.log` and `mcnp/logs/example-2.log` for the results using a 2019 Mac Pro and a 2023 MacBook Pro respectively.

As you may notice above, we recommend running this example without parallelization. Since the dimensionality is very small, using OpenMP in this case will make the program run slower. The overhead cost of the parallelization for this small model exceeds the benefits.

List of Tables and Programs

The provided code reproduces all tables in the paper. Figures in the paper do not require code.

Output	Program	Description	Runtime
Table 1	`rustct`	Rust (1987) Sample Characteristics	30 sec
Table 2	`rustct`	Model Estimates Based on Data from Rust (1987)	30 sec
Table 3	`mc1p control/mc-<delta>-<nm>.ctl`	Single Agent Renewal Model Monte Carlo Results	5 min
Table 4	`mcnp control/time-<nn>.ctl`	Quality Ladder Model Monte Carlo Specifications	1 sec - 10 hr
Table 5	`mcnp control/mc-<nn>-<delta>.ctl`	Quality Ladder Model Monte Carlo Results	5 min - 5 days

`<delta>` denotes the discrete observation time interval:

- `0.00`, `1.00`, and `8.00` for Table 3
- `0.0` and `1.0` for Table 5

`<nm>` denotes the number of markets observed:

- `200`, `800`, and `3200` for Table 3

`<nn>` denotes the number of firms:

- `02` through `30` for Table 4
- `02` through `08` for Table 5

As described above, each of these programs (and for each control file), produces console output corresponding to rows of the tables. Researchers will need to run each program with each specification to reproduce the complete tables. Specific details are given for each table below. The results obtained by the author are provided in the respective `mc1p/logs/` and `mcnp/logs/` directories.

References

- Byrd, R. H., P. Lu, and J. Nocedal (1995). A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific and Statistical Computing* 16, 1190–1208.
- Ericson, R. and A. Pakes (1995). Markov-Perfect Industry Dynamics: A Framework for Empirical Work. *Review of Economic Studies* 62, 53–82.

- Rust, J. (1987). Optimal Replacement of GMC Bus Engines: An Empirical Model of Harold Zurcher. *Econometrica* 55, 999–1033.
- Sidje, R. B. (1998). Expokit: A software package for computing matrix exponentials. *ACM Transactions on Mathematical Software* 24, 130–156.
- Zhu, C., R. H. Byrd, P. Lu, and J. Nocedal (1997). Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization. *ACM Transactions on Mathematical Software* 23, 550-560.

License