# Dynamic Discrete Choice Models

## An example in Matlab

Shihang Hou Giulio Schinaia

University of Oxford

November 26, 2021

# Outline

# Today

- Present an example and Matlab code based on Rust (1987)

# Today

- Present an example and Matlab code based on Rust (1987)
- Some concepts are clearly explained on review article by Aguirregabiria and Mira (2010) (very useful reference!)

# Overview

- Based on Rust (1987), Optimal Replacement of GMC Bus Engines: An Empirical Model of Harold Zurcher, Econometrica
- One of the first examples of a dynamic discrete choice model (many online code examples R/Julia/Python/Gauss and *now* Matlab for different solution methods available —check Aguirregabiria's website )
- Code is in: `https://github.com/bzdiop/structuralwg/tree/main/presentations/code/SHGS/rust_code` There are two sets of scripts:
    - finite time horizon (with suffix `_finite`)
    - infinite time horizon (with suffix `_inf`), which we illustrate first, because it is what the paper implements.
        - Maybe obvious point: The model assumes the agent makes decisions indefinitely, but the estimation only requires a finite panel.

# Harold's problem

- In each period $t$ and for each bus $b$, Harold has to make a discrete choice $a$:
  - to replace the engine of the bus, $a = 1$
  - or not replace the engine, $a = 0$.
- To make the decision, Harold observes two (types of) state variables:
  - $x$: Mileage since last replacement (**Observable** to the researcher)
  - $\varepsilon$: Reports from the drivers and mechanics, for example. (**Unobservable** to the researcher)

# How do we model Harold's behaviour?

In each period, we assume the following static utility `function` :

$$U(x_t, a_t, \theta_1) = \begin{cases} - \quad c(x_t, \theta_1) + \varepsilon_t(0) \text{ if } a_t = 0 \\ -[\overline{P} - \underline{P} + c(0, \theta_1)] + \varepsilon_t(1) \text{ if } a_t = 1 \end{cases} \tag{1}$$

- Standard cost of maintenance depends on mileage: $c(x_t, \theta_1)$, e.g. `linear`
- If replace engine, incur (net) replacement cost (RC) : $\overline{P} - \underline{P}$ ...
- ...but incur lower maintenance costs $c(0, \theta_1)$
- Note that we are also making an assumptions about how both the observed and *unobserved* state variables enter the **additively** in the utility function

# (Observable) State variable transition

- We also need to make an assumptions about how the state variables evolve over time.
- Transition of mileage from period to period is summarised by a stochastic process described by $P(x_{t+1}|x_t, a_t, \theta_2)$
- For example, we may *assume* that the gain in mileage is a draw from a `exponential distribution`. But this assumption may not fit Harold's data very well (see section III pf the paper).
- **Instead, we assume a more flexible multinomial distribution** to describe how mileage evolves over time (see next slides) given the choice taken in each period.

# Create matrix of transition probabilities for all possible states

generate_data_inf.m

```
% Create matrix Fx_0 (71 times 71)
% Remember that this is because you may transition from any state to any
% other state, but the probabilities of transitioning to most states are set to zero,
Fx_0 = zeros(length(x_grid),length(x_grid));
for i = 1:length(x_grid)
    Fx_0(i,i) = p_x0;
    if i <= length(x_grid) - 1
        Fx_0(i,i+1) = p_x1;
    end
    if i <= length(x_grid) - 2
        Fx_0(i,i+2) = p_x2;
    end
end
Fx_0(length(x_grid)-1,length(x_grid)) = 1- p_x0;
Fx_0(length(x_grid),length(x_grid)) = 1;

% Create matrix Fx_1 (71 times 71)
Fx_1row = zeros(1,length(x_grid));
Fx_1row(1) = p_x0;
Fx_1row(2) = p_x1;
```

# Matrix of transition probabilities given no replacement

Fx_0

71x71 double

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| 1 | 0.3919 | 0.5953 | 0.0128 | 0 | 0 |
| 2 | 0 | 0.3919 | 0.5953 | 0.0128 | 0 |
| 3 | 0 | 0 | 0.3919 | 0.5953 | 0.0128 |
| 4 | 0 | 0 | 0 | 0.3919 | 0.5953 |
| 5 | 0 | 0 | 0 | 0 | 0.3919 |
| 6 | 0 | 0 | 0 | 0 | 0 |
| 7 | 0 | 0 | 0 | 0 | 0 |

# Matrix of transition probabilities given no replacement

- At zero miles (row 1), there is a 0.3919 probability of only doing 0-5000 miles (cell 1,1), a 0.5953 probability of doing 5001-10000 miles, or a 0.0128 probability of doing 10000-$\infty$ miles.
- **By assumption**, we are bounding the steps to be up to 15000 miles.
- This means that there is zero probability of going from zero to beyond 15000 miles.
- There is a zero probability of going back from 5000-10000 to 5000-10000 (cell 2,1).

# What is the goal of the econometrician?

**Goal:** Estimate the paramaters $\theta$ of the model :

- replacement cost
- maintenance costs parameters
- the transition probabilities for the (observable) state variable

based on the behavioural model and the econometric assumptions needed to solve it.

# generate_inf.m: Initialise the parameters to estimate

generate_data_inf.m

```
    % Parameter values taken from col 2 of table 9 in Rust (1987)
    %beta = 0.999;              % discount factor, could be smaller to speed up convergence
    %p_x0 = 0.3919;             % probability of making [0-5000) miles
    %p_x1 = 0.5953;             % probability of making [5000-10000) miles
    %p_x2 = 1- p_x0 - p_x1;     % probability of making [10000 - \infty) miles

    pars = [10.0750;0.00005293;0.3919;0.5953];   % Starting values for data generation
    rc = pars(1);               % Replacement cost
    theta1_1 = pars(2);         % Maintenance cost paramater with a linear cost function
    p_x0 = pars(3);             % Transition probs
    p_x1 = pars(4);

    beta = 0.8;                 % Discount factor
    p_x2 = 1- p_x0 - p_x1;
    it_tol = 1e-6;             % tolerance paramater to define minimum distance in contraction mapp

    x_grid = transpose(0:5000:350000); %Set up discretised state space of x

    % Vector of costs in each period, for each choice
    u_1 = arrayfun(@(x) cost(1,x,pars),x_grid); % Compute u(a,x) for a = 1 (71 times 1)
    u_0 = arrayfun(@(x) cost(0,x,pars),x_grid); % Compute u(a,x) for a = 0 (71 times 1)
```

# Harold's problem over time

- Optimal value function (by Bellman's principle):

$$V_\theta(x_t, \varepsilon_t) = \max_{a \in \mathcal{C}(x_t)} [u(x_t, a, \theta_1) + \varepsilon_t(a) + \beta EV_\theta(x_t, a)] \tag{2}$$

- Where the "last term" above is denoted as the Emax function:

$$EV_\theta(x_t, a) = \int_{-\infty}^{\infty} \int_{0}^{\infty} V_\theta(y, \varepsilon) p(y|x_t, a, \theta_2) dy dG(\varepsilon) \tag{3}$$

# Rust's assumptions (1)

- Thanks to Rust's assumptions (IID), (CIX) , we can rewrite 3 as:

$$EV_\theta(x_t) = \int \max_{a_t \in \mathcal{C}(x_t)} \left\{ u(a, x_t) + \varepsilon_t + \beta \int EV_\theta(x_{t+1}) P(x_{t+1}|x_t, a) dx_{t+1} \right\} dG_\varepsilon(\varepsilon_t) \tag{4}$$

# Rust's assumptions (1)

- Thanks to Rust's assumptions (IID), (CIX) , we can rewrite 3 as:

$$EV_\theta(x_t) = \int \max_{a_t \in \mathcal{C}(x_t)} \left\{ u(a, x_t) + \varepsilon_t + \beta \int EV_\theta(x_{t+1}) P(x_{t+1}|x_t, a) dx_{t+1} \right\} dG_\varepsilon(\varepsilon_t) \tag{4}$$

- Usually, would need to compute the double integrals in eq 3, which would be computationally intensive.

# Rust's assumptions (1)

- Thanks to Rust's assumptions (IID), (CIX) , we can rewrite 3 as:

$$EV_\theta(x_t) = \int \max_{a_t \in \mathcal{C}(x_t)} \left\{ u(a, x_t) + \varepsilon_t + \beta \int EV_\theta(x_{t+1}) P(x_{t+1} | x_t, a) dx_{t+1} \right\} dG_\varepsilon(\varepsilon_t) \tag{4}$$

- Usually, would need to compute the double integrals in eq 3, which would be computationally intensive.

- Thanks to Rust's assumption (LOGIT), if $\varepsilon$ is distributed according to a type I extreme value distribution, we get:

$$EV_\theta(x_t) = \log \left( \sum_{a \in \mathcal{C}(x_t)} \exp \left( u(a, x_t) + \beta \int EV_\theta(x_{t+1}) P(x_{t+1} | x_t, a) dx_{t+1} \right) \right) \tag{5}$$

# Rust's assumptions (2)

- For tractability, Rust divides the continuous state space into a discrete set of points.

$$EV_\theta(x_t) = \log\left(\sum_{a\in\mathcal{C}(x_t)} \exp\left(u(a, x_t) + \beta \sum_{x_{t+1}} EV_\theta(x_{t+1})P(x_{t+1}|x_t, a)\right)\right) \quad (6)$$

- For a problem with an infinite time horizon and a finite state space, we can solve for the emax functions as the unique solution to a finite system of equations

$$\bar{\boldsymbol{V}} = \log\left(\sum_{a=0}^{J} \exp\left\{\boldsymbol{u}(a, \theta) + \beta \boldsymbol{F}_x(a)\bar{\boldsymbol{V}}\right\}\right) \quad (7)$$

# Deriving the Emax values by contraction mapping

- Rust (1987) shows that equation 6 is a contraction mapping from $\bar{\boldsymbol{V}} \rightarrow \bar{\boldsymbol{V}}$.
- A straightforward, though computationally complicated approach is to iterate values of the policy functions until the difference between iterations is sufficiently small.

$$\bar{\boldsymbol{V}}_{h+1} = \log\left(\sum_{a=0}^{J} \exp\left\{\boldsymbol{u}(a,\theta) + \beta \boldsymbol{F}_x(a)\bar{\boldsymbol{V}}_h\right\}\right) \tag{8}$$

- Note: For finite horizon applications, backward induction is most common solution method (see example later).

# Iterating the value function

iteration.m

```
function [Vit] = iteration(Vbar,Fx_1,Fx_0,u_0,u_1,beta)
    it_0 = exp(u_0 + (beta .* (Fx_0 * Vbar))); %VBar update given no replac
    it_1 = exp(u_1 + (beta .* (Fx_1 * Vbar))); %VBar update given replaceme
    it = it_0 + it_1;
    Vit = log(it);
end
```

# Contraction mapping algorithm to evaluate following 8

inner_algo.m

```
function [Vbar_1] = inner_algo(Fx_1,Fx_0,u_0,u_1,beta,it_tol)
    Vbar_1 = log(exp(u_0) + exp(u_1)); % Start from the static case
    max_itdiff = 1;                    % Start with a value of the difference between iterations
    it_counter = 0;
        while max_itdiff > it_tol
            Vbar_0 = Vbar_1;           % Update the current Vbar with the previously computed t+1
            Vbar_1 = iteration(Vbar_0,Fx_1,Fx_0,u_0,u_1,beta); % Update the value of Vbar
            it_diff = abs(Vbar_1 - Vbar_0);
            max_itdiff = max(it_diff);
            %display(max_itdiff)
            it_counter = it_counter + 1;
            %display(it_counter)
        end
end
```

# Estimation steps of the outer algorithm

1. Start with guess $\hat{\theta}_0$.
2. Evaluate $\bar{\boldsymbol{V}}(\hat{\theta}_0)$, either by iteration until convergence using the contraction mapping (see equation 7 or backwards induction for the finite case).
3. Use $\bar{\boldsymbol{V}}(\hat{\theta}_0)$ and parameter guesses $\hat{\theta}_0$ to compute the choice probability, $P(a|x, \theta)$.
4. Evaluate the log-likelihood and repeat until a local optimum has been reached.

# From the inner to the outer algorithm

`estimate_inf.m` : Uses the dataset to optimise the log-likelihood function in order to estimate the structural parameters, their standard errors, and confidence intervals.

- `rust_loglik_inf.m`
  1. performs similar steps as `generate_data_inf.m` to generate the choice probabilities using the inner algorithm, but using different parameters for each iteration of the optimising algorithm.
     - Uses the same functions to get to the choice probabilities: `cost.m` , `c.m` , `inner_algo.m`, `iteration.m`
  2. Computes the likelihood for the choice probabilities and the transition probability components.

# What Rust's assumptions buys us (1) - factorising the log-likelihood

- Under CIX and IID, the observable state vector $x_{it}$ is sufficient to determine the current choice, and allows the factorisation of the various terms that enter the log-likelihood contribution.

$$l_i(\theta) = \sum_{t=1}^{T_i} \log(Pr(a_{it}|x_{it}, \theta)) + \\ + \sum_{t=1}^{T_i - 1} \log(f_X(x_{i,t+1}|a_{it}, x_{it}, \theta_f)) \tag{9}$$

# Rust's log-likelihood

rust_loglik_inf.m

```
    end
    % Likelihood function for all the structural paramaters
    function [sumloglik] = loglik(data,choiceprob_1,choiceprob_0,Fx_0,Fx_1)
        N = size(data,1);
        loglikvec = zeros(N,1);
        for i = 1:N
            loglikvec(i) = choiceprob(data(i,1),data(i,2),choiceprob_1,choiceprob_0) + ...
                           transprob(data(i,2),data(i,3),data(i,1),Fx_0,Fx_1);
        end
        sumloglik = -sum(loglikvec);
    end
```

Data contains binary choice to replace in column 1, current mileage in column 2, and next period's mileage in column 3.

# Components of the log-likelihood

rust_loglik_inf.m

```
    % Likelihood component for the choice probability
    function [logchoiceprob] = choiceprob(a,x,choiceprob_1,choiceprob_0)
        modx = floor(x./5000)+1;
        if a == 1
            logchoiceprob = log(choiceprob_1(modx));
        elseif a == 0
            logchoiceprob = log(choiceprob_0(modx));
        else
            error('a should be 0 or 1.')
        end
    end
    % Likelihood component for the transition probability
    function [logtransprob] = transprob(x,x_f,a,Fx_0,Fx_1)
        modx = floor(x./5000)+1;
        modxf = floor(x_f./5000)+1;
        if a == 1
            logtransprob = log(Fx_1(modx,modxf));
        elseif a == 0
```

# What Rust's assumptions buys us (2) - functional form for choice probability

- Recall the optimal decision rule is $\alpha(x_{it}, \varepsilon_{it}) = \arg\max_{a \in A}\{v(a, x_{it}) + \varepsilon_{it}(a)\}$, so...

$$\begin{aligned} Pr(a|x,\theta) &\equiv \int I\{\alpha(x,\varepsilon;\theta) = a\} dG_\varepsilon(\varepsilon) \\ &= \int I\{v(a, x_{it}) + \varepsilon_{it}(a) > v(a', x_{it}) + \varepsilon_{it}(a'), \forall a' \neq a\} dG_\varepsilon(\varepsilon_{it}) \qquad (10) \\ &= \frac{\exp(v(a,x))}{\sum_{j=0}^{J} \exp(v(j,x))} \end{aligned}$$

- Note slight abuse of notation

$$v(a, x_t) = u(a, x_t) + \beta \sum_{x_{t+1} q \in X} \bar{V}(x_{t+1}) f_x(x_{t+1}|a, x_t)$$

# Choice probabilities

iteration.m | estimate_inf.m | rust_loglik_inf.m

```
% Write the v(a,x)s
v_0 = u_0 + (beta .* (Fx_0 * Vbar));
v_1 = u_1 + (beta .* (Fx_1 * Vbar));

% Compute choice probabilities
exp_v0 = exp(v_0);
exp_v1 = exp(v_1);
sum_v = exp_v0 + exp_v1;
choiceprob_1 = exp_v1 ./ sum_v;
choiceprob_0 = exp_v0 ./ sum_v;
```

# Let's run the code

...

# Finite horizon introduction

- Suppose that a planner runs a bus service, knowing at the start that his franchise ends in 12 periods.
- This then becomes a finite time horizon problem, which is typically solved by **backwards induction**.

# The final period

- First, consider the last period $\boldsymbol{T}$. The value of each choice j in T is simply $\boldsymbol{U_j}$, given the value of the state at that time $\boldsymbol{x_T}$.
- The expected value at time T given state value $\boldsymbol{x_T}$ is thus the E-max function:

$$EV_T(x_T) = \int \max_{a_T \in \{0,1\}} \{u(x_T, a_T) + \varepsilon_T(a_T)\}\, dG_\varepsilon(\varepsilon_T) \tag{11}$$

$$= \log\left(\sum_{a=0}^{J} \exp(u_T(a, x_T))\right) \tag{12}$$

- In matrix form:

$$\bar{\boldsymbol{V}}_T(\hat{\theta}) = \log\left(\sum_{a=0}^{J} \exp(\boldsymbol{u}_T(a, \hat{\theta})\right) \tag{13}$$

# The penultimate period $(T-1)$

- The value of choosing any option $j$ is now $U_j(a_{T-1}, x_{T-1}) + \beta \mathbf{F}_{x,t}(a, x_{T-1})\bar{\mathbf{V}}_T(\hat{\theta})$.
- The expected value at T-1 given state value $x_{T-1}$ is again the E-max function, this time with the implications for the future period included

$$EV_{T-1}(x_{T-1}) = \int \max_{a_{T-1}\in\{0,1\}} \left\{ U(x_{T-1}, a_{T-1}) + \beta \mathbf{F}_{x,t}(a, x_{T-1})\bar{\mathbf{V}}_T(\hat{\theta}) \right\} dG_{\varepsilon}(\varepsilon_{T-1}) \tag{14}$$

$$= \log\left(\sum_{a=0}^{J} \exp(u_{T-1}(a, x_{T-1}) + \beta \mathbf{F}_{x,t}(a, x_{T-1})\bar{\mathbf{V}}_T(\hat{\theta}))\right) \tag{15}$$

# Backwards induction

- In general, for the Rust assumptions, in matrix form, the values can be solved for recursively using the following expression.

$$\bar{\boldsymbol{V}}_t(\hat{\theta}) = \log\left(\sum_{a=0}^{J} \exp\left\{\boldsymbol{u}_t(a, \hat{\theta}) + \beta \boldsymbol{F}_{x,t}(a)\bar{\boldsymbol{V}}_{t+1}(\hat{\theta})\right\}\right) \tag{16}$$

- These solved values can then be used to evaluate the probability of observing the given choices:

$$Pr_t(a|\hat{\theta}) = \frac{\exp(u_t(a, \hat{\theta}) + \beta \boldsymbol{F}_{x,t}(a)\bar{\boldsymbol{V}}(\hat{\theta}))}{\sum_{j=0}^{J} \exp(u_t(j, \hat{\theta}) + \beta \boldsymbol{F}_{x,t}(j)\bar{\boldsymbol{V}}(\hat{\theta}))} \tag{17}$$

# Rust's assumptions (1)

- (AS) Additive separability

$$U(a, x_{it}, \varepsilon_{it}) = u(a, x_{it}) + \varepsilon_{it}(a)$$

- (CLOGIT) The unobserved state variables $\{\varepsilon_{it}(a) : a = 0, 1, \cdots, J\}$ are independent across alternatives and have an extreme value type I distribution.

- (Discrete support of x) The support of $x_{it}$ is discrete and finite: $x_{it} \in X = \{x^{(1)}, x^{(2)}, \cdots, x^{(|X|)}\}$ with $|X| < \infty$.

# Rust's assumptions (1)

- (IID) Unobserved state variables are i.i.d across agents and time, with common cdf $G_\varepsilon(\varepsilon_{it})$
- (CIX) Next period observable state variables do not depend on unobserved state variables ($\varepsilon_{it}$)

$$CDF(x_{it}|x_{it}, a_{it}, \varepsilon_{it}) = F_x(x_{i,t+1}|a_{it}, x_{it})$$

  Denote parameters of $F_x$ by $\theta_f$

- CIX + IID $\rightarrow$ $F(x_{i,t+1}, \varepsilon_{i,t+1}|a_{it}, x_{it}, \varepsilon_{it}) = G_\varepsilon(\varepsilon_{i,t+1})F_x(x_{i,t+1}|a_{it}, x_{it})$

# Rust's assumptions (3)

- (CIY) Conditional on current values of the decision and observable state variables, the value of the payoff variable, $Y$, is independent of $\varepsilon_{it}$

$$Y(a_{it}, x_{it}, \varepsilon_{it}) = Y(a_{it}, x_{it})$$

  - This assumption is not needed in the Rust (1987) example, because we do not have a payoff variable.

# Some examples of the restrictiveness of Rust's assumptions

- **CIX**: Consider an example an observed state variable $x_t$ is human capital and an unobserved state variable is the worker's health $\varepsilon_t$. CIX implies then that health does not affect the rate of accumulation of human capital.
- **CLOGIT and iid**: Choice probabilities exhibit IIA and does not allow for covariance of unobservable state variables - e.g. in occupation choice, unobserved utility are independent between occupations
- **AS**: Implies that marginal utility with respect to observable state variables does not depend on unobservables. e.g. the marginal effect on wage of human capital does not depend on health shocks

# Alternative assumptions (e.g. in Eckstein-Keane-Wolpin models)

- Relaxing AS by using non-linear functional forms
- Observable $Y$ variables that are choice-censored and do not satisfy CIY
- Permanent unobserved heterogeneity using mixture models
- Unobservables that are correlated across choice options (departing from CLOGIT)

# What Rust's assumptions buys us (1) - evaluating the Emax function

- Fully flexible model

$$V(x_{it}, \varepsilon_{it}) = \max_{a \in A} \{U(a, x_{it}, \varepsilon_{it}) + \beta V(x_{i,t+1}, \varepsilon_{i,t+1}) dF(x_{i,t+1}, \varepsilon_{i,t+1} | a, x_{it}, \varepsilon_{it})\}$$

- Rust's model (by AS, CIX, IID, CLOGIT, discrete support)

$$\bar{V}(x_{it}) = \int \max_{a \in A} \left\{ u(a, x_{it}) + \varepsilon_{it}(a) + \beta \sum_{x_{i,t+1} \in X} \bar{V}(x_{i,t+1}) f_x(x_{i,t+1} | a, x_{it}) \right\} dG_{\varepsilon}(\varepsilon_{it}) \tag{18}$$

$$= \log \left( \sum_{a=0}^{J} \exp \left( u(a, x_{it}) + \beta \sum_{x_{i,t+1} \in X} \bar{V}(x_{i,t+1}) f_x(x_{i,t+1} | a, x_{it}) \right) \right) \tag{19}$$

Equation 18 is known as the Emax function, with a known form for logit errors.

# What Rust's assumptions buys us (2) - factorising the log-likelihood

- Under CIX and IID, the observable state vector $x_{it}$ is sufficient to determine the current choice, and allows the factorisation of the various terms that enter the log-likelihood contribution.

$$\begin{aligned} l_i(\theta) = \sum_{t=1}^{T_i} \log(Pr(a_{it}|x_{it},\theta)) + \sum_{t=1}^{T_i} \log(f_Y(y_{it}|a_{it},x_{it},\theta_Y)) \\ + \sum_{t=1}^{T_i-1} \log(f_X(x_{i,t+1}|a_{it},x_{it},\theta_f)) + \log(Pr(x_{i1}|\theta)) \end{aligned} \tag{20}$$

# What Rust's assumptions buys us (3) - functional form for choice probability

- Recall the optimal decision rule is $\alpha(x_{it}, \varepsilon_{it}) = \arg\max_{a \in A}\{v(a, x_{it}) + \varepsilon_{it}(a)\}$, so...

$$\begin{aligned} Pr(a|x, \theta) &\equiv \int I\{\alpha(x, \varepsilon; \theta) = a\} dG_{\varepsilon}(\varepsilon) \\ &= \int I\{v(a, x_{it}) + \varepsilon_{it}(a) > v(a', x_{it}) + \varepsilon_{it}(a'), \forall a' \neq a\} dG_{\varepsilon}(\varepsilon_{it}) \\ &= \frac{\exp(v(a, x))}{\sum_{j=0}^{J} \exp(v(j, x))} \end{aligned} \tag{21}$$

- Note slight abuse of notation

$$v(a, x_t) = u(a, x_t) + \beta \sum_{x_{t+1} q \in X} \bar{V}(x_{t+1}) f_x(x_{t+1}|a, x_t)$$

# Linear costs

c.m

```
function [out] = c(x,pars)
    theta1_1 = pars(2);
    out = theta1_1 .* x;
end
```

# Static utilities

cost.m

```
function [u] = cost(a,x,pars)
    rc = pars(1);

    if a == 1
        u = -(c(0,pars) +  rc);
    elseif a == 0
         u = -c(x,pars);
    else
         error('a should be 0 or 1.')
    end
end
```

# State transition probabilities assuming exponential distribution

$$P(x_{t+1}|x_t, a_t, \theta_2) = \begin{cases} \theta_2 \exp\{\theta_2(x_{t+1} - x_t)\} \text{ if } a_t = 0 \text{ and } x_{t+1} \geq x_t \\ \theta_2 \exp\{\theta_2(x_{t+1})\} \text{ if } a_t = 1 \text{ and } x_{t+1} \geq 0 \\ 0 \text{ otherwise} \end{cases} \tag{22}$$

- Exponential distribution would buy us a closed form solution to the optimal control decision rule, but it is not supported by the data (see Section III of the paper).