1Noter til dynamic programming

# Structural estimation of Markov decision process

## Introduction

The general idea of the Markov decision process is to establish to optimal way in which to make sequential decisions under uncertainty.

It has been extensively used in theoretical studies, as it is able to solve most economic problems involving choices made over time.

Before considering if it is possible to estimate the agents beliefs and preferences, it should be considered if it is logically possible.

It then goes through the different limitations and possibilities of the Markov decision process and the discrete decision process.

## Solving MDP’s via dynamic programming, a brief overview.

Det er meget teoretisk, tror det er mere egnet til opslag, end til at læse alene.

# Dynamic economic quantative methods and application

## Chapter 1: overview

This chapter is mostly about what they will go through throughout the book, and why they do so

## Chapter 2: theory of dynamic programming

### Indirect utility

This part is just about how they are generally optimization problems, which can be used for firms and consumers to maximize their utility

### Cake eating example

If there was a cake, some person could choose the optimal consumption saving, providing the highest amount of utility for the user. The cake is assumed not to neither grow nor decay.

We start by analyzing the cake eating problem through a lens of a finite problem.

The lifetime utility of the cake is written as

Where is the discount factor, and

#### Direct attack

Based on formula (1) we might look at the problem through wanting to maximize the wealth and the consumption

### Some extensions to the cake eating problem

#### Infinite horizon

This chapter starts out talking about the infinite horizon of the basic structure. It states, that it is dependent on the very standard formula meaning that we wish to maximize the consumption and cake size throughout all periods. Further, we wish to gain as much utility as possible.

We are also given the usual equation which means that the wealth(cake) in next period is given as the wealth in this period minus what is being consumed.

This is then recreated to make the formula

which is the dynamic programming problem, stating that the value of the infinite horizon problem is dependent on the utility of consumption plus the discounted value of the wealth minus the consumption.

In the formentioned equation, the state variable is the size of the cake W, the control variable is c, which is chosen. Those two combined creates the transition equation W’=W-c

We can then move the equation to depend on the next period to make it easier algebraically.

To find the first order condition we would first differentiate the important one written above

##### Example

We are now given an example, where

Given the results from the t-period problem, the general form of the equation is

This then goes on to become

We then find the first order condition, which is

This is then inserted into the equation above

We then collect all the terms that include the ln(W)

Herfra kan man se at hvilket når man isolerer for B giver

#### Taste shocks

En af de store fordele ved dynamisk programmering, er at man relativt nemt kan tilføje usikkerheder til sine udregninger. Til at udregne ”taste shocks” bruger man til at påsætte værdien af disse taste schock.

Man bliver nødt til at udregne hvor meget disse ”taste shocks” kommer til at betyde i fremtiden, samt hvor meget det betyder i nuværende periode, for at finde det optimale forbrug. Man sætter her i første omgang til at være i diskrete værdier altså eller . Værdien af disse baseres derefter også på sandsynligheden for at det bliver den ene eller den anden af de to. Det opgøres derefter i en matrice, som kaldes **Transition matrix** som jeg antager ser således ud

Man løser herefter førsteordens bettingelserne.

#### Discrete choice

Nu antager vi at kagen skal spises i en periode. Vi tilføjer dertil muligheden for at kagen kan vokse (mindske) med raten . Disse typer valg bruges blandt andet til at finde ud af hvornår man bør tage et job, hvornår man bør slå sig ned…

### General formulation

#### Non-stochastic case

Forestil dig et infinite horizon optimerinsproblem med en payoff function for period t givet som

Agentens payoff over perioden skrives som

For at maximere payoff skal man enten finde førsteordensbettingelserne eller man kan løse

Dette skulle derefter løses for at finde den optimale ”policy function”

#### Stochastic dynamic programming

I denne udregning tilføjer vi stochastiske shocks ligesom vi gjorde tilbage i økonometri når vi lavede en ar model.

### Conclusion

Simply states that the chapter is mostly about how the different theories work, and how they are a cornerstone of how dynamic programming works. This is to say also that there will be worked further on all the theories, which will be more empirically oriented.

## Chapter 3: Numerical analysis

### 3.1 Overview

This tells how this chapter will go over the numerical analysis aspect of dynamic programming, which will be used a lot during the remainder of this book. We will especially go over **Value function iteration(VFI)** Which we have also already used in class.

### 3.2 Stochastic cake eating problem

The stochastic cake eating problem is written as the function

With

In this equation there are two state variables which is the size of the cake, and y, which is the endowment of cake in the following periods.

We start by analysing in the situation, where the function for y is iid, meaning independent and identically distributed, meaning that each y doesn’t tell anything about the next periods y.

The consumer would in this case only care about the amount which can be eating given as the function

We can then rewrite the original problem to only be based on the x making it

When programming a value iteration function one should go through 4 steps

1. Choosing a functional form of the utility function
2. Discretizing the state and control variables
3. Building a computer code to perform value iteration functions
4. Evaluating the value and the policy function.

#### 3.2.1 Value function iteration

##### Functional form and parameterization

Here we are given the function

We assume for now that the growth of the cake is equal to the discounting factor of the discounting value.

##### State and control space

When we wish to make these calculations we cannot do it on a continuous time, as that we demand infinite computation. We can however closely estimate is using some vector of different values, with more values equaling more precise answer but also more computational time.

##### Value function iteration and policy function

Here it informs about how the values are iterated over a number of times until the difference between this period and the next is below a certain threshold. This is something you pick yourself, again with a trade-off between computational time and precision.

This is also based on an initial guess for the correct value for the consumption function, where a better guess will decrease the computation time.

#### 3.2.2 Policy function iteration

Sometimes the value functions can be a bit slow to converge, as it will only converge with the value there are therefore created faster methods, herunde the policy function iteration, which is about

#### 3.2.3 Projection methods

##### Solving for the policy rule

##### Collocation methods

##### Finite element methods

### 3.3 Stochastic discrete cake eating problem

#### 3.3.1 Value function iterations

### 3.4 Extension and conclusion

#### 3.4.1 Larger state spaces

# Savings and liquidity constraints

## Introduction

The paper seems to be people who have some liquidity constraint, as they do not make more than enough money to go by, why they are less able to invest an gain traction throughout their lifetime. This is also partly because it shown te be very hard to borrow for increased consumption, since theis would in theory worsen the families economic situation.

He seems to further investigate how different kinds of incomes and borrowing opportunities result in different levels of investments, and also how they at times result in less than optimal savings strategies.

## Savings and liquidity constraints with basic income

### The basic model

This chapter seemed to provide further basis that there was a problem if the consumer weren’t able to save above a certain threshold, they wouldn’t be able top have sustainable savings, and would then be regressing to their average income.

### Stationary serially stationary income

In this model, there is assumed an autoreggresive model, meaning that an increase in income in one period, would mean that there is a higher income in the next and vice verca.

## Savings and liquidity constraints with nonstationary income

Here he studies the savings and liquidity with nonstationary income, meaning that the income doesn’t have to go back to some mean.

### independent and identically distributed growth

Nothing I really understood.

### autocorrelated growth and the cycle

### individual behavior, noisy income and individual behavior

# Questions and answers

What is the Markov decision process?

* The Markov decision process is a maximization problem, where we wish to optimize the utility based on how we decide to react to the state space. This can then be done either in a finite horizon perspective or an infinite horizon.

What are the differences between the infinite and the finite horizon dynamic programming problems?

* The finite horizon is solved using dynamic programming, but where you start with the ending period, and then go back to see how we should do in the first period.
* With the infinite horizon however, there wouldn’t be the same constraints, as you would have infinite time in all periods. This is solved by guessing I think, though I am not entirely sure.

What does the v mean in the bellman equation

* It seems to imply the value with which we are attempting to maximize.
* V is the value function – not entirely sure what that exactly means
* Maybe it is the asset price?
* Value or the sum of all future values after accounting for the discounting.

# Theory tools

## DP intro

**Markov decision process –** is a framework for modelling sequential decision making under uncertainty. These are often solved using dynamic programming, which has been done by a lot of different people.

Generally the Markov decision process is written up as four parts

1. The decision points
2. The state variables
3. The payoff function – These are the payoffs under the circumstances that something or something else happens.
4. The motion rules – These are the chances that different outcomes will happen

If this Markov decision process is finite, it will be solved using the backwards induction process.

If it however is infinite, we can’t simply go back from the end, we therefore look to Value Function Iterations(VFI) instead.

## Cake eating

The cake eating problem is an example of the Markov decision process, with the size of the current cake being the state, the decision is how much cake to eat, and the payoff is the utility you get from eating cake in each period. The transition is deterministic, meaning that the motion rules, are that the amount of cake you have in next period is the cake you have now minus the cake you have eaten.

## Deaton 1 dimensional integration

## Multidimensional integration

## Portfolio integration

## Portfolio optimization

## Interpolation

# Structural estimation of **dynamic discrete choice models**

## The nested fixed point algorithm

This was an algorithm made by Harold Alois Zuercher, to implement maximum likelihood models on discrete choice models, hereunder the rust model, which is about the replacement of bus engines

The components of a dynamic model consists of

1. Decision variables
2. State variables
3. Payoffs – utility functions with time seperable payoffs
4. Motion rules - beliefs about how the future will be dependent on the current state and decision. Eg. If someone decides to invest in a company, they believe that there is a 50% chance that they will earn a lot

The solution is then either given by

1. The optimal value function
2. The policy function which gives the best decision based on the state variable.

## The bus engine replacement model

**Binary choice set**

meaning you fix the bus meaning you don’t fix the bus

**Different state variables**

:mileage at time t

: decision state specific variable

**Utility function**

In terms of the rust problem, it can be said that if the engine is replaced, the costs are the replacement cost and tha costs of running the engine, which is lower if it has just been replaced. If you don’t replace the engine, there is no need for the replament costs, meaning that the costs are the costs of usual maintenance, which is increasing with mileage plus both have an error term

**Motion rules**

Which means that if the engine is replaced, the mileage resets

**Parameters to be estimated**

## Structural estimation

Once given the data and the likelihood function, one solves for the log likelihood function, which also means that the function can be split into two.

## Data

Harold Zurcher has data for 162 busses, with monthly information on each bus, and data on maintenance decisions. These are then plotted into the function to calculate the likelihood that someone is changing the engine

Et billede, der indeholder tekst, linje/række, diagram, Parallel

Automatisk genereret beskrivelse

The large pins are because they don’t have much data, meaning that once they get to 390 miles, they have almost never gotten that far without replacing the engine.

## Equilibrium bus mileage and demand for engines

## Mathematical programming with equilibrium constraints(MPEC)

The difference between this one and NFXP, equilibrium is found with constraints. With the unconstrained model, one has to solve the bellman equation a lot of time. The constraint model is faster, since it holds more parameters constant, thereby constraining the model. This constrained model then maximizes the likelihood function for both Structural parameters and the expected value.

Some people have given to much credit to the mpec method on the cost of other methods. However using the NFXP method is as fast, when used correctly.

To use the NFXP correctly, one has to go through the 6 steps of NFXP

1. Read the nfxp manual and print out the pocket guide.
   1. The NFXP can be looked at as solving the following constrained maximization problem
2. Recenter the logit and logsum formulas
   1. Logit formulas must be reentered

The logsum must be reentered to

1. Use the fixed point poly algorithm
   1. To find the fixed point of the contraction mappint one has to
      1. Succesive approximation (SA) by contraction iteration
2. Provide analytical gradiants of the bellman operator
3. Provide analytical gradients of the likelihood
4. Use BHHH (outer products of gradients in a hessian approximation)

## 3 Conditional choice probabilities(CCP) and nested pseudo likelihood

What if we could solve dynamic problems without calculating the nested fixed point problem?

### Hotz and miller

Their plan is to estimate using observable data on p and x. Then using their inversion theorem, they plan to map onto the value function.

### CCP estimators

Step 1 – Estimate a reduced form conditional choice probability using the d and x data, and labe it

Step 2 – Use the hotz miller inversion map to estimate value function differences, and thereby measure the things that goes into the value criterion.

### The power of hotx-miller inversion

1. Develop two step estimators to evaluate structural parameters without solving the model
2. Provide empirically tractable estimations of conditional value functions
3. Analyze identification in dynamic discrete choice models
4. Introduce new ways to incorporate unoibserved heterogeneity using the EM algorithm
5. Exploit time dependence when estimating non stationary problems.

It is however inefficient and biased in small samples. – Today, one would use the neural nets, to make ai do the work for you.

## Sequential estimation

### Nested pseudo likelihood(NPL) estimation

They use the hotz miller method, and the sequentially update it to obtain a sequence of estimators.

When NPL is initialized with consistent nonparametric estimates of the CCP’s sequence, it can be used to solve the extre cases of Hotz miller with for k=1 and Rust’s nested fixed point MLE estimators using k

### The general problem

Bellmann equation

A few changes to the general formulation, but still following a somewhat standard protocol.

### A&M maintains rusts assumptions

There is again some conditional independence.

Addaptive separability, as

### Bellman equation and choice probabilities

We define the smoothed value function where sigma is the parameter which represents those parametres, which would be theta\_1 in rust notation

Under assumptions CI AS and finite domain of x we can summarize the solution be the smoothed bellman operator.

Write up the smoothed vvellman operator and the conditional choice probability.¨

There needs to be integrated as many times as there are dimensions.

### From conditional choice probabilities to value functions

We would like to express the equation V(X) from last slide. Mapped from choice probabilities to choice probabilities.

## 4 Equilibrium trade in automobiles - The doubly nested fixed point algorithm(DNFXP)

### The model is Zurcher on steroids

**Consumers –** mass units, infinitely lived and discrete types

The ownership decisions are: Keep, trade, purge, scrap or sell.

Driving decisions are how much to drive

j-types of cars and a-types of ages

**Scrappage:** Forced or by choice – Stochastic when due to accident, end of life(when it comes to the last age, Endogenous when getting rid of the car voluntarily.

**Idiosyncratic heterogeneity.**

### The DNFXP algorithm

Step 1 – Do a max likelihood optimization over

Step 2 – inner equilibrium solver is about finding the price so

Step 3 – excess demand means that each trial value of P must

* Solve a single agent DP/fixed point given p
* Compute transition matrices and Q
* Find stationary holdings distributions
* Evaluate excess demand

### Utility of the car

The formula is written as

The is a decreasing function as the car ages, which reflects the increased amounts of check ups, and maintenance costs.

is the utility of money.

### How to compute stationary flow equilibrium quickly

**Gradient solver**

**Analytical derivatives**

**Precise starting values**

We therefore use newtons method.

We use first the value function iteration, which is linear, then we use the newton method, which is quadratic.

Is there a difference between the newton method and the newton kotes? – Gradient based solver

## Time iterations and endogenous gridpoint method.

### Euler Equation

**Bellman equation**

Unless the constraints are binding, one should receive the same results using other methods, which means that the other measures can be used to see if you have done everything correctly

**Example consumption savings problem(deaton).**

Using the first order conditions of the value function and the envelope theorem, which is the same, one will get.

Perfect consumption smoothing is then established when

### Time iterations

Not much to say, it is a relatively basic thing

### Endogenous grid point methog(EGM)

This one is the most accurate method for both finite and infinite horizont discrete choice problem, and very useful for the consumption savings problem.

This is the same as the deaton model, but instead of writing w-c we write A

## Continuous and discrete-continuous decision problems

### Discrete and continuous choice

There are generally a lot of both continuous and discrete choices which you have to go through in life. These are often modelled differently based on the problem, with:

# Kodnings noter

## Hvordan løser man en kode med backwards induction

**Setup**

Du starter med at lave dit setup, det vil sige at give dine parametre værdi

**Initialize**

Derefter initialiserer du din model, det gøres ved at opsætte de steder man indsætter sine værdier. Det vil sige at man lave to matricer, en med de forskellige consumption værdier, hvilket vil sige at man indsætter fra 0 til 5 i sidste periode, hvis det er det maksimale antal kage man kan indtage.

**Solve**

Når du skal solve, skal du grundet backwards induction gå gennem tidsperioderne fra slutperioden, og tilbage. Du laver derfor et for-loop hvorfra man kører fra næstsidste periode, da vi allerede har sidste periode. For hver periode, skal man nu opsætte for samtlige størrelser af kagen, heri opsat fra 0 til 5. Man opsætter nu forbrug(c) som kan være alle reale tal fra 0 til kagestørrelsen. Man opsætter derefter kagestørrelsen i næste periode, hvilket er kagestørrelsen minus forbrug

## Hvordan løser man en kode med value function iterations(VFI)

**Setup**

Her starter man med at sætte de forskellige parametre op, hvilket inkluderer beta hvilket er parametren for diskonteringen, w hvilket er ”kagestørrelsen”, grid\_w hvilket er de forskellige mulige karastørrelse op til kagestørrelsen, og sidst Cstar, hvilket gerne skal blive til det optimale forbrug, men starter ud med at være et W+1 dimensionelt grid

**Parametre for VFI**

Max\_iter – er det antal gange python må gå gennen funktionen,

Delta – er den forskel der må være mellem v\_next og v\_now, altså hvor meget værdi man får i denne periode, og hvor meget man får i næste. Hvis man sænker forskellen tager det længere tid før de bliver ens.

**While-loop**

Her gør man det klart at man fortsætter med at køre over værdierne/formlerne så længe it<max\_iter hvilket betyder så længe man ikke har kørt over formlerne flere gange end man har villet. Dette gøres for at det ikke skal tage for lang tid og in case det ikke konvergerer. Dertil tilføjer man at man ellers først skal stoppe når forskellen mellem value i denne periode og næste periode er under en hvis forskel(Delta). Man kører derefter igennem værdierne man læste for i backwards induction, men hvor de gemmes i de tidligere opsatte grids

**Tilføjelser**

Denne opgave bliver også løst i exercises, hvor man indeler de forskellige parametre i **par** og de forskellige løsninger i **sol**

### Hvad betyder sol, par og sim

Disse er variable, der bliver lave til at opbevare henholdsvis løsninger, parametre og simuleringer.

@ bruges til matrix multiplikation.

# Notes for the term paper