Energy management for vehicle power net using stochastic dynamic programming

El Karouni, R.

Award date:
2005

Eindhoven University of Technology
Department of Electrical Engineering
Control Systems

TU/e technische universiteit eindhoven

Energy Management for vehicle power net using Stochastic Dynamic Programming

by Rachid El Karouni

Project period: November 2004 – June 2005

Report nr. 05A/12
Supervisors:
Prof.dr.ir. P.P.J. van den Bosch
Ir. J.T.B.A Kessels

Additional Commission members:
Ir. K.M Nauta

ACKNOWLEDGEMENTS

First of all, I would like to thank John Kessels for helpful comments, inspiring discussions and direction.

Furthermore, I would like to express my gratitude to all persons at the 'Control System' department for providing the nice and stimulating environment. Among all the people at the department, I owe special gratitude to Prof. P.P.J Van Den bosch for making this research possible and for his insight in stochastic engineering. Last but not least, I am extremely grateful to my family for their patience and support, especially during the late night and weekend editing the thesis.

CONTENTS

LIST OF FIGURES

ABSTRACT

Hybrid electric vehicles have two power sources, an internal combustion engine and an electric motor. These vehicles are of great interest because they contribute to decreasing fuel consumption and less air pollution and still maintain the performance of a conventional car.

The supervisory control strategy of a hybrid vehicle coordinates the operation of vehicle sub-systems to achieve performance targets such as maximizing fuel economy and reducing exhaust emissions. This high-level control is commonly referred as the power management problem.

The energy management problem can also be mapped to vehicles with a conventional drivetrain. This will be the aim of this study. Different than in a hybrid electric vehicle, now the electric machine can only be used in generator mode and not in motor mode.

In this study, the power management problem is tackled from a stochastic viewpoint. An infinite-horizon stochastic dynamic optimization problem is formulated. The power demand from the driver is modeled as a random Markov process. The optimal control strategy is then obtained by using Stochastic Dynamic Programming (SDP). The obtained control law is in the form of a stationary state feedback and can be directly implemented.

The goal is to minimize fuel consumption and keep the energy level of the Battery near to a reference level. Unlike in the conventional approach (deterministic case), the minimization is performed not for a predetermined drive cycle but in a stochastic, "average" sense over a class of trajectories from an underlying Markov chain drive cycle generator.

Abbreviations

Abbreviations used (alphabetical order):

MDP: Markov Decision Process

MLE: MLE Maximum Likelihood Estimation

NEDC: New European Driving Cycle

SDP: Stochastic Dynamic Programming

Nomenclature:

- ω_{wheel} : the wheel speed

- τ_d : the torque of the drive train

- P_d : the power needed by the drive train for vehicle propulsion

- P_l : the electric power for the electric loads

- P_s : the storage power in the battery

- E_s : the energy stored in the battery

- E_{cap} : the energy capacity of the battery

- $SOC = \frac{E_s}{E_{cap}} 100\%$: the state of charge

- P_b : the power entering or leaving the battery terminals

- $\prod$: set of control policies

- J^* : the optimal cost function

- μ : stationary policy

- π : control policy

-x: state of the system

-u: control variable or action

-w: random disturbance

- x_t : state of the system at time t

- u_t : action taken at time t

- w_t : value of the random disturbance at time t state space

-D: disturbance space

-C: control space

-U(x): set of actions that can be taken in state x

INTRODUCTION

The thesis is organized as follows: In the first chapter, the theory of stochastic dynamic programming is explained by an introduction to the value iteration, policy iteration and linear programming algorithms. Through this chapter, the Bellman equation will be introduced. In the second chapter, we will explain in details the model of the environment, also the description of the signals present in the system will be introduced. The development of the controller will be explained in chapter three. The fourth chapter is composed by a couple of results like the random drive cycles generated by the matrix of transition.

CHAPTER I

I. Introduction to "Stochastic Dynamic Programming"

In this chapter, stochastic dynamic programming is used to handle discrete-time optimal control problems with infinite time horizon. Although these infinite time horizon control problems do not exist in practice because physical processes are bounded in time, their analysis is elegant and the implementation of the optimal policies computed is simple because they are stationary. Among the different types of infinite horizon problems we consider only the discounted problems for which the "cost_per_stage" are weighted so that the importance of the cost_per_stage decays exponentially *with their time of appearance. Furthermore we will assume that the cost_per_stage are bounded;*

1.1 Problem formulation

We start by formulating an infinite horizon problem, which has the advantage (compare to the finite horizon) that the control policy generated is time-invariant and thus can be easily implemented.

Definitions:

1. $\prod$: is the set of all *admissible* policies π, that is the set of all sequences of functions $\pi = \{\mu_0, \mu_1, \mu_2, ...\}$ with $\mu_k : S \to C, \mu_k(x_k) \in U(x_k)$, *for all* $x_k \in S, k = 0,1,...,$

2. The optimal cost function J^* is defined by $J^*(x_0) = \min_{\pi \in \prod} J_\pi(x), x_0 \in S.$

3. A stationary policy is an admissible policy of the form $\pi = \{\mu, \mu, \mu, ...\}$, and its corresponding cost function is denoted by J_μ

4. A stationary policy (for simplicity, we denote it by μ) is optimal if

$J_\mu(x) = J^*(x)$ for all states x

5. *Deterministic stationary (Markov) policy.* A policy is a deterministic stationary policy if it corresponds to a policy that selects at time k the control $u_k = \mu_k(x_k)$

where $\mu_k : S \to C, \mu_k(x_k) \in U(x_k), for\ all\ x_k \in S, k = 0,1,...$ such policies choose while being in a state x, always the same action and are denoted for simplicity by μ (cf def.4)

1.1.1 From infinite horizon problem to Bellman's equation

Total cost infinite horizon

Consider the stationary discrete-time dynamic system

$$x_{k+1} = f(x_k, u_k, w_k), \quad k = 0,1,..., \tag{0.1}$$

where for all k, the state x_k is an element of a space S, the control u_k is an element of a space C, and the random disturbance w_k is an element of a space D. We assume that D is a countable set.

The control u_k is constrained to take values in a given nonempty subset U(x_k) of C, which depends on the current state x_k [$u_k \in U(x_k), for\ all\ x_k \in S$]

The random disturbances w_k, k=0,1,...,have identical statistics and are characterized by probabilities $P(\cdot \mid x_k, u_k)$ defined on D, where $P(w_k \mid x_k, u_k)$ is the probability of occurrence of w_k ,when the current state and control are x_k *and* u_k ,respectively.

Given an initial state x_0, we want to find a policy $\pi = \{\mu_0, \mu_1, \mu_2, ...\}$,where $\mu_k : S \to C, \mu_k(x_k) \in U(x_k), for\ all\ x_k \in S, k = 0,1,...,$ that minimizes the cost function

$$J_\pi(x_0) = \lim_{N \to \infty} \underset{\substack{w_k \\ k=0,1,..}}{E} \left(\sum_{k=0}^{N-1} \alpha^k g(x_k, \mu_k(x_k), w_k) \right), \tag{0.2}$$

Subject to the system equation constraint (0.1)

The cost per stage g: $S \otimes C \otimes D \to \Re$ is given, and α is the discount factor with $0 < \alpha \leq 1$ that weights the cost per stage. The cost per stage we consider are supposed to be bounded, here "⊗" is the Cartesian product.

The DP Algorithm for the Finite-Horizon

Truncated version ($N \neq \infty$)

Given any admissible policy $\pi = \{\mu_0, \mu_1, ..., \mu_{N-1}\}$ with positive integer N, any function $J: S \to \Re$,and any initial state x_0, we want to find a policy that minimizes the cost function

$$J_\pi(x_0) = \underset{\substack{w_k \\ k=0,1,..}}{E} \left(\sum_{k=0}^{N-1} \alpha^k g(x_k, \mu_k(x_k), w_k) \right), \tag{0.3}$$

If we accumulate the costs of the first N stages, and to them we add the terminal cost then we obtain

$$\underset{\substack{w_k \\ k=0,1,...}}{E} \{\alpha^N J(x_N) + \sum_{k=0}^{N-1} \alpha^k g(x_k, \mu_k(x_k), w_k)\} \tag{0.4}$$

The minimum of this cost over the policy π can be calculated by starting with $\alpha^N J(x)$ and by carrying out N iterations of the DP algorithm

The DP Algorithm

We start with the initial condition

$$J_N(x) = \alpha^N J(x). \tag{0.5}$$

and go backwards using

$$J_k(x) = \min_{u \in U(x)} E\{\alpha^{N-k} g(x,u) + J_{k+1}(f(x,u))\}, k = 0,1,...,N-1, \tag{0.6}$$

The optimal N-stage cost is the function $J_0(x)$

If we define for all x and k, the function

$$V_k(x) = \frac{J_{N-k}(x)}{\alpha^{N-k}}.$$

then the DP recursion (0.6) can be written in terms of the functions V_k as

$$V_{k+1}(x) = \min_{u \in U(x)} E\{g(x,u) + \alpha V_k(f(x,u))\}, k = 0,1,...,N-1, \tag{0.7}$$

with the initial condition $V_0(x) = J(x)$

$V_N(x)$ is the optimal N-stage cost $J_0(x)$

You can consider (0.2) as the summation of two pieces:

$$J_\pi(x_0) = \underset{\substack{w_k \\ k=0,1,\ldots,K-1}}{E} \{\sum_{k=0}^{K-1} \alpha^k g(x_k, \mu_k(x_k), w_k)\} + \lim_{N\to\infty} \underset{\substack{w_k \\ k=K,\ldots,N-1}}{E} \{\sum_{k=K}^{N-1} \alpha^k g(x_k, \mu_k(x_k), w_k)\} \qquad (0.8)$$

Because of the discount factor, the second term becomes zero for large K. So the infinite horizon approximates the infinite horizon better if K becomes larger. This is achieved by repeating (0.7) for larger N.

The problem (0.2) and (0.7) are equivalent for large N and K, and to explain that we have to define two mappings.

1.1.2Definition of two mappings

We are going to introduce these two mapping that play an important role.

For any function, $J : S \to \Re$

We define the function obtained by applying the DP mapping to J, and we denote it by

$$(TJ)(x) = \min_{u\in U(x)} \underset{w}{E}\{g(x,u,w) + \alpha J(f(x,u,w))\}, \quad x \in S \qquad (0.9)$$

Similarly, for any function $J : S \to \Re$ and any action function $\mu : S \to C$, we denote

$$(T_\mu J)(x) = \underset{w}{E}\{g(x,\mu(x),w) + \alpha J(f(x,\mu(x),w))\}, x \in S \qquad (0.10)$$

Other notation:

We will denote by T^k the composition of the mapping T with itself k times; that is for all k we write

$$(T^k J)(x) = (T(T^{k-1}J))(x), x \in S \qquad (0.11)$$

and with k=0 , we have $(T^0 J)(x) = J(x), x \in S$ (0.12)

Similarly with the mapping T_μ , that is the same definition, you have just to add the index μ .

The Q-function

We use the notation $S \times U$ to represent the set of all possible state-action pairs (x , u).

Therefore, we have $S \times U = \{(x,u) \mid x \in S \text{ and } u \in U(x)\}$.

We define the Q-function $Q : X \times U \to \Re$ as follows:

$$Q(x,u) = \underset{w}{E}[g(x,u,w) + \alpha J(f(x,u,w))], \qquad (0.13)$$

Thus we can redefine the mapping function (0.9) and (0.10) as follows:

$$(TJ)(x) = \min_{u \in U(x)} \underset{w}{E}\{g(x,u,w) + \alpha J(f(x,u,w))\}, \\ = \min_{u \in U(x)} Q(x,u) \tag{0.14}$$

$$(T_\mu J)(x) = \underset{w}{E}\{g(x,\mu(x),w) + \alpha J(f(x,\mu(x),w))\}, \\ = Q(x,\mu(x)) \tag{0.15}$$

The Q-function knowledge allows to directly compute an optimal policy(ies)[1] through the following expression:

$$\mu^*(x) = \underset{u \in U(x)}{\arg\min} \underset{w}{E}\{g(x,u,w) + \alpha J^*(f(x,u,w))\} = \underset{u \in U(x)}{\arg\min} Q^*(x,u) \tag{0.16}$$

Notice that in practice, the determination of an optimal stationary policy will be realized most of the time from the Q-function.

The case of infinite horizon ($N \to \infty$)

Thanks to the propositions below, we are able to finish the explanation of the equivalence of the 2 problems.

Proposition 1.1

For any bounded function $J : S \to \Re$ *the optimal cost function satisfies.*

$$J^*(x) = \lim_{N \to \infty} (T^N J)(x), \text{ for all } x \in S \tag{0.17}$$

One can prove (see [4]) that by infinitely repeating mapping T,one will obtain the optimal cost function J^*

Proposition 1.2(Bellman's Equation)

The optimal cost function J^* *satisfies*

$$J^*(x) = \min_{u \in U(x)} \underset{w}{E}\{g(x,u,w) + \alpha J^*(f(x,u,w))\}, \tag{0.18}$$

Or, equivalently,

$$J^* = TJ^* \tag{0.19}$$

[1] The optimal policy μ^* is not unique

Also here, the prove is given in Berstekas ([4]). J^* is the unique solution of this equation because it is the unique fixed point of the mapping defined earlier (see eq. (0.9)), one can use this result to find the optimal stationary policy μ.

We have just shown that solving the Bellman equations comes down to solving the infinite horizon problem; in other words, problem *(0.18)*is equivalent to problem (0.2)

1.2 Stochastic Dynamic Programming Approach

Solving a Stochastic Dynamic Programming problem can be done in various ways. The best-known methods are POLICY-ITERATION and VALUE-ITERATION. The first one manipulates policies directly whereas the second one is looking for the optimal value function

Before introducing those algorithms, we will explain how the environment and the agent interact.

The Agent-Environment Interface

In a very general way, as we can see in **Figure 1**, we have an environment and an agent. This interface is applied for calculating the controller, the off-line calculation. The on-line situation is used in real-world driving. The controller is called AGENT. The thing it interacts with, comprising everything outside the agent, is called the ENVIRONMENT[2]. The agent and environment interact with each other in a sequence of discrete time steps, k=0, 1, 2... At each time step k, the agent receives some representation of the environment's state, $x_k \in S$, where S is the set of possible states, and on that basis selects an action u_k, where the action is constrained to take values in a given nonempty subset U(x_k) of C, which depends on the current state x_k .One time step later, the agent selects u_k not only as a function of x_k but also as a function of the cost at time $k, g_k \in \Re$

[2] Also called plant or controlled system

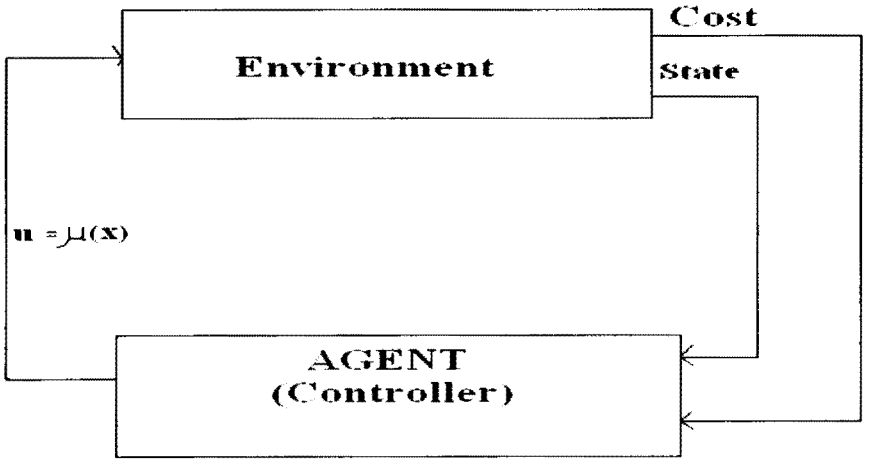

Figure 1 Interaction between environment and agent

The aim of the SDP is to find the optimal control policy such that the cost function is minimized. The Environment (controlled system) will be developed in chapter II and the controller in the Chapter III.

1.2.1 How to compute the solution?

3 methods will be explained:

-Policy iteration

-Value iteration

-Linear programming

Before the explanation of these algorithms, a notation is given that is more convenient for Markov chains

Let the state space S consist of n states denoted by 1,2,…,n:

$S = \{1, 2, ..., n\}$

and denoted by $p_{ij}(u)$ the transition probabilities

$$p_{ij}(u) = P(x_{k+1} = j \mid x_k = i, u_k = u), \quad i, j \in S, u \in U(i) \tag{0.20}$$

These transition probabilities may be given a priory

We have also the cost per stage

$$g(i,u) = \sum_{j=1}^{n} p_{ij}(u)\, \tilde{g}(i,u,j). \tag{0.21}$$

where $\tilde{g}(i,u,j)$ is the cost of using u at state i and moving to state j.

$E\{—\} = \sum_{j=1}^{n} p_{ij}(u)\{—\}$. is our definition of the expectation

The mappings T and T_μ (cf (0.9)) and the Q-function can be written as

$$(TJ)(i) = \min_{u \in U(x)} \{g_1(x,u) + \alpha \sum_{j=1}^{n} p_{ij}(u) J(j)\}, \tag{0.22}$$

$$Q(x,u) = \{g\ (x,u) + \alpha \sum_{j=1}^{n} p_{ij}(u) J(j)\}, \tag{0.23}$$

where the expected cost obtained by choosing u in state x is given by

$$g\ (x,u) = \underset{w}{E}\{g(x,u,w)\} \tag{0.24}$$

1.2.1.1 Policy iteration

The *policy iteration* algorithm manipulates the policy directly, rather than finding it indirectly via the optimal value function.

Policy iteration consists of two simultaneous, interacting processes, one making the value function consistent with the current policy (policy evaluation) and the other making the policy greedy with respect to the current value function (policy improvement)

NB: A policy is greedy with respect to any finite value function J if it prescribes to each state a control action that minimizes the sum of the immediate costs g(i,u) plus the sum of the discounted expected value of the next state as determined by the value function J.

In broad outline:

Given an initial stationary policy μ_0

$$\mu_0 \to J_{\mu_0} \Rightarrow \mu_1 \to J_{\mu_1} \Rightarrow \ldots \mu^* \to J^* \Rightarrow \mu^*$$

we compute the cost function (policy evaluation) and then we improve the policy until the policy remains unchanged

notation "$\to$" is the policy evaluation

"$\Rightarrow$" is the policy improvement ("greedification")

Policy Evaluation

How to compute the cost functions J_μ for any stationary policy?

Thanks to the proposition 2.3 (given below), we know that J_μ is the unique solution of the equation

$$J_\mu = T_\mu J_\mu \tag{0.25}$$

$$\text{then } J_\mu \overset{i}{=} T_\mu J_\mu \overset{ii}{=} g_\mu + \alpha P_\mu J_\mu \tag{0.26}$$

i)due to proposition 1.2

ii) due to *(0.19)*

The equation (0.26) should be viewed as a system of n linear equations with n unknowns (n is the number of states) if the environment's dynamics are completely known (g_μ *and* P_μ)

NB:

$$g_\mu = (g(1,\mu(1)), g(2,\mu(2))..., g(n,\mu(n)))^T \tag{0.27}$$

$$P\mu = \begin{pmatrix} p_{11}(\mu(1)) & \cdots & p_{1n}(\mu(1)) \\ \vdots & \ddots & \vdots \\ p_{n1}(\mu(n)) & \cdots & p_{nn}(\mu(n)) \end{pmatrix}, \tag{0.28}$$

The equation (0.26) can be also written as

$$J_\mu = (I - \alpha P_\mu)^{-1} g_\mu, \tag{0.29}$$

where I denotes the $n \times n$ identity matrix.

NB: The invertibility of the matrix $(I - \alpha P_\mu)$ is assured because the system of equations representing $J_\mu = T_\mu J_\mu$ has a unique solution for any vector g_μ.

The main weakness of this algorithm is that it requires the solving of the equations (0.29). The dimension of this system is equal to the number of states and thus if the state space is large, this algorithm does not stay attractive anymore.

Proposition 1.2: For every stationary policy μ, the associated cost function satisfies

$$J_\mu(x) = \underset{w}{E}\{g(x,\mu(x),w) + \alpha J_\mu(f(x,\mu(x),w))\}, \tag{0.30}$$

or, equivalently,

$$J_\mu = T_\mu J_\mu. \tag{0.31}$$

Furthermore, J_μ is the unique solution of this equation within the class of bounded functions.

Policy Improvement
It is based on the following proposition

Proposition 1.3

Let μ *and* $\bar{\mu}$ be stationary policies such that $T_{\bar{\mu}} J_\mu = TJ_\mu$,

Or equivalently, for $i=1,...,n$,

$$g(i,\bar{\mu}(i)) + \alpha \sum_{j=1}^{n} p_{ij}(\bar{\mu}(i)) J_\mu(j) = \min_{u \in U(i)} \{g(i,u) + \alpha \sum_{j=1}^{n} p_{ij}(u) J_\mu(j)\}. \tag{0.32}$$

or equivalently with the Q-function, for $i=1,...,n$,

$$Q(i,\bar{\mu}(i)) = \min_{u \in U(i)} Q(i,u) \tag{0.33}$$

Then we have

$$J_{\bar{\mu}}(i) \leq J_\mu(i), \quad i = 1,...,n. \tag{0.34}$$

Furthermore, if μ is not optimal, strict inequality holds in the above equation for at least one state i.

Policy Iteration Algorithm

i) Initialization…

Select an initial random stationary policy μ^0

ii) Policy Evaluation

Given the stationary policy μ^k (k-th iteration), compute the corresponding cost function J_{μ^k} from the linear system of equations

$(I - \alpha P_{\mu^k}) J_{\mu^k} = g_{\mu^k}$.

iii) Policy improvement

Obtain a new stationary policy μ^{k+1} satisfying

$T_{\mu^{k+1}}J_{\mu^k} = TJ_{\mu^k}$ (We improve the policy for each state x until the policy remains unchanged)

If $J_{\mu^k} = TJ_{\mu^k}$

then stop

else return to ii) Policy evaluation and repeat the process

One drawback to policy iteration is that each of its iterations involves policy evaluation, which may itself be a protracted iterative computation requiring multiple sweeps through the state set. If policy evaluation is done iteratively, then convergence exactly to J_μ occurs only in the limit.

1.2.1.2 Value iteration

The value iteration method is also called successive approximation.

We start with any n-dimensional vector J and successively compute $TJ, T^2J, \ldots$

Where T is defined in (0.9).

And we know that this sequence of vectors converges to $J^*(i)$

We have (see *(0.17)*)for all i

$$\lim_{k\to\infty}(T^kJ)(i) = J^*(i) \qquad (0.35)$$

The value iteration algorithm

Initialize random $J(i)$ *for all* $i \in S$

Repeat indefinitely

errorMax=0

Repeat once for all $i \in S$

$$J_{k+1}(i) \leftarrow \min_{u\in U(i)}\{g(i,u)+\alpha\sum_{j=1}^{n} p_{ij}(u)J_k(j)\}.$$

$$errorMax = \max(errorMax, |J_{k+1}(i) - J_k(i)|)$$

$J_k \leftarrow J_{k+1}$

If errorMax < ε then stop the algorithm and use J as an approximation of J^*

The use of the value iteration algorithm requires, at the end of the estimation of the cost function, an additional step to compute the optimal stationary policy.

1.2.1.3 Linear Programming

Since $\lim_{N\to\infty} T^N J = J^*$ for all J(cf *(0.17)*)

We have

$J \geq TJ \quad \Rightarrow \quad J \geq J^* = TJ^*.$

This constraint can be written as a finite system of linear inequalities

$$J(i) \geq g(i,u) + \alpha \sum_{j=1}^{n} p_{ij}(u) J(j), i = 1,2,...,n \quad u \in U(i),$$

The problem is equivalent to find $J^*(1),...,J^*(n)$ $(\lambda_1 = J^*(1),...,\lambda_n = J^*(n))$

minimize $\sum_{i \in \tilde{S}} \lambda_i$

subject to $\lambda_i \geq g(i,u) + \alpha \sum_{j=1}^{n} p_{ij}(u) \lambda_j, i = 1,2,...,n \quad u \in U(i),$

Where $\tilde{S}$ is any nonempty subset of the state space S={1,…,n}.This is a linear program with n variables and as many n*q constraints, where q is the maximum number of elements of the sets U(i).

NB: For very large n and q, the linear programming approach can be practical only with the use of special large-scale linear programming methods.

CHAPTER II

In this chapter, an introduction to hybrid vehicles is presented, followed by a complete problematic description –from a stochastic viewpoint- that includes all relevant signals, vehicle and battery models.

2. Vehicle modelling

2.1 Introduction to hybrid vehicle

A hybrid vehicle is a vehicle that uses a mixture of technologies such as internal combustion engines (ICEs), electric motors, batteries, hydrogen, and fuel cells. Today's hybrid cars are driven by electric motors powered by both batteries and an ICE.
Hybrids do not necessarily have to be plugged in, yet still deliver superior mileage and performance. They are environmentally-friendly alternatives to traditional internal combustion engine vehicles.
Prior to its modern usage to mean hybrid propulsion, the word hybrid was used in the United States to mean a vehicle of mixed national origin; generally, a European car fitted with American mechanical components. This meaning has currently fallen out of use.
Some assert that current hybrid cars are not true hybrids because they are not capable of using either engine or motor independent of each other. Others feel that none of the current "Hybrids" are such since none are capable of using alternate fuels such as electricity from an outlet, ie: "Hybrid Fueled". However, hybrids come in a variety of different configurations.

A hybrid vehicle uses two different power sources. They are an electric motor and almost always an internal-combustion engine. In the hybrid design, an electric motor or several electric motors power the car, and a combustion engine keeps the batteries charged and assists when more power is needed (e.g., for sudden acceleration). This contrasts with all-electric cars which use batteries charged by an external source. Benefits of the hybrid design include the following:

- The vehicle is usually lighter and roomier than a purely electric vehicle of comparable size and power because fewer batteries are needed.
- The internal-combustion engine in a hybrid vehicle is much smaller, lighter, than the one in a conventional vehicle, because the electric motor can provide a boost of power for acceleration.
- Braking in a hybrid vehicle is controlled by the electric motor which recaptures part of the kinetic energy of the car to partially recharge the batteries. This is called regenerative braking and one of the reasons for the high efficiency of hybrid cars. In a conventional vehicle, with internal-combustion engine, braking is done by mechanical brakes, and the kinetic energy of the car is wasted as heat.
- Most hybrid powertrains are characterized by a split power path called a powersplit. One side of this split is electrical and the other side of this power path is mechanical.

We should emphasize that we will study vehicle with a conventional drivetrain. This will be the aim of this research. Different than in a hybrid electric vehicle, now the electric machine can only be used in generator mode and not in motor mode.

2.2 Description of signals present in the system

2.2.1 State Space

The proposed State space x is a four-dimensional State space:

$$x : \{SOC, \omega_{wheel}, P_d, P_l\} \quad with\ x \in S \tag{0.36}$$

where - *SOC is a deterministic variable*
- ω_{wheel}, P_d ' are *stochastic variables*
- P_l is a *constant.*

2.2.2 Action Space

We choose to call the control variable "u" representing the Power stored in the Battery P_s and is an element of a space C.

$$u : P_s \quad with \quad u \in C \tag{0.37}$$

Reminder:

- $SOC = {E_s}/{E_{cap}}$ *100% with E_s, the battery energy level and E_{cap}, the energy capacity of the battery.

- ω_{wheel}, the wheel speed

- P_d the power of the drive train for the vehicle propulsion

- P_l electric power for the electric loads

NB:The variable ω_{wheel}, P_d being observable but not controllable, we could have been attempt to not put these variables in the state space

In a general way, the problem can be to formalize by (by negligent the stochastic aspects in the notations)

1.y(t+1)=h(y(t)) (where y is the vector of observable variable but not controllable, here ω_{wheel}, P_d)

2.x(t+1)=f(x(t),y(t),u(t)) (where u is the control Ps, and x is Es) (The function f(x(t),y(t),u(t)) can be independent of y(t))

3.g(t)=g(x(t),y(t),u(t)) (The cost function)

Under these conditions, it is enough that either the function of cost g(...) or the function f(...) be dependant of the variable y(t) to justify for putting as well x as y in the state.

Then, due to the fact that in the majority of the literature about this theory, the policy is only a function of the state, we prefer to follow that direction, thus include the non controllable variable (ω_{wheel}, P_d) into the state.

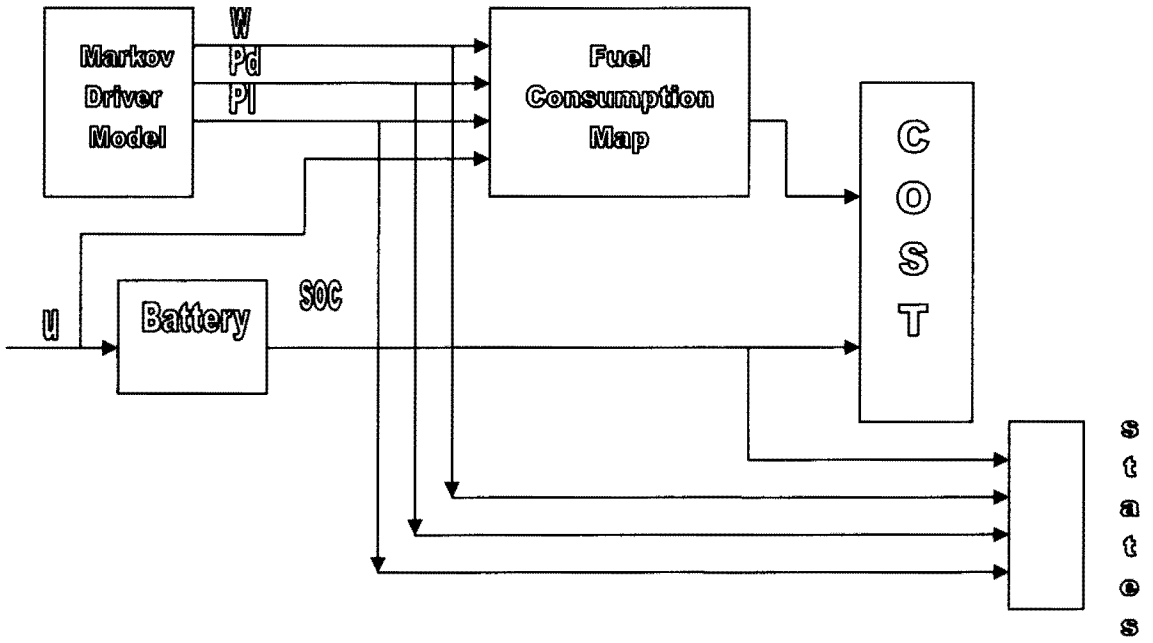

Figure 2 Detailed view of the Environment[3]

2.3 Development of the environment

In this chapter, we are developing and elaborating on the whole stochastic environment that includes a vehicle model and a description of the proposed Markov Driver Model.

2.3.1 VEHICLE MODEL (Introduction of physics variables)

We consider a vehicle with a conventional drive train and a manual transmission. This vehicle is comparable to a parallel hybrid vehicle although the power of the generator must be non-negative.

[3] Cf figure 1 page 11

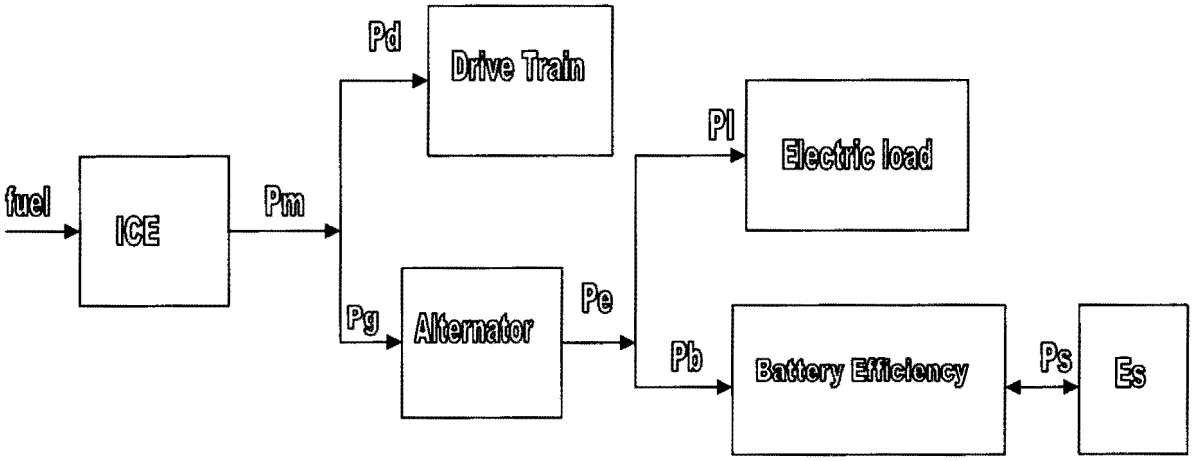

Figure 3 Vehicle model

The power flow in the vehicle starts with fuel that goes into the internal combustion engine (ICE). The mechanical power (P_m) that comes out of the engine (ICE) splits up into two directions: (cf equation (0.41) below)

i) One part goes to the mechanical drive train for vehicle propulsion (P_d) whereas the other part goes to

ii) The alternator (P_g).

Next, the alternator provides electric power for the electric loads but also takes care of charging the battery. Contrary to the other components, the power flow through the battery can be positive as well as negative (see **Figure 3**, the arrows show the positive direction of the power). In the end, the power becomes available for vehicle propulsion and for electric loads connected to the power net.

In this model of the car, there are six blocks that we have to model.

The goal of energy management is to control the alternator power such that the fuel consumption is reduced, while the drivability remains unaffected.

It implies that the vehicle speed and thus the drive train torque and engine speed remain unaffected and therefore it is possible to use them as given information.
We assume this assumption because the driver should not experience different vehicle behaviour when the controller is applied. Consequence of that assumption, we can make a static model of the Power demand Pd.

P_l is electric power for the electric loads. It can be modeled by a constant function. The model of the battery will be explained later. P_b represents the power entering or leaving the battery terminals, and Ps represents the power actually stored in the battery. P_{loss} represents the battery losses that depend on the storage power, the energy level in the battery E_s, and the temperature T (see equations (0.39)),in our case, we will neglect the effect of these two last parameters (Es and T).

The losses[4] in the battery are positive for both charging and discharging. This can be obtained by making the losses quadratic with the storage power:

$$P_b = P_s + P_{loss}(P_s, E_s, T) \tag{0.38}$$

$$P_{loss} = \beta P_s^2 \tag{0.39}$$

The remaining components of interest are the engine, the alternator, and the battery.

The internal combustion engine (ICE) can be represented by a nonlinear static map ($f(P_m, w)$) which describes the relation between fuel consumption, engine speed, and engine power:

$$fuelrate = f(P_m, w) \tag{0.40}$$

The mechanical Power P_m is the sum of the power to the drive train P_d and the alternator power P_g

$$P_m = P_d + P_g \tag{0.41}$$

[4] The losses in the battery are implemented in the program

The alternator model consists of a static map $ge(P_e, \omega)$ connecting the electric power P_e to the mechanical power P_g

$$P_g = ge(P_e, \omega) \tag{0.42}$$

$$P_e = P_l + P_b \tag{0.43}$$

NB: The models of the alternator and of the internal combustion engine are available by means of a look-up-table.

2.3.2 Markov driver Model

The first block represents the power asked by the driver travelling on the drive cycles. It models torque and engine speed for the drive cycles. The theory of Maximum Likelihood Estimation (MLE) was applied and is explained below.

2.3.2.1 Stochastic Modeling of Driver Power Demand

Two representative driving cycles are chosen to construct the observation samples of the stochastic system. The New European Driving Cycle (NEDC) and the CYC_ARB02 cycle.

The NEDC cycle, as we can see in **Figure 4 NEDC cycle** below, is a combined cycle consisting of four "ECE 15" cycles followed by a EUDC cycle. The "ECE 15" driving cycle represents urban driving. It is characterized by low vehicle speed (max.50 km/h), low engine load. The EUDC cycle describes a suburban route. At the end of the cycle the vehicle accelerates to highway-speed. Speed is higher than the "ECE 15".

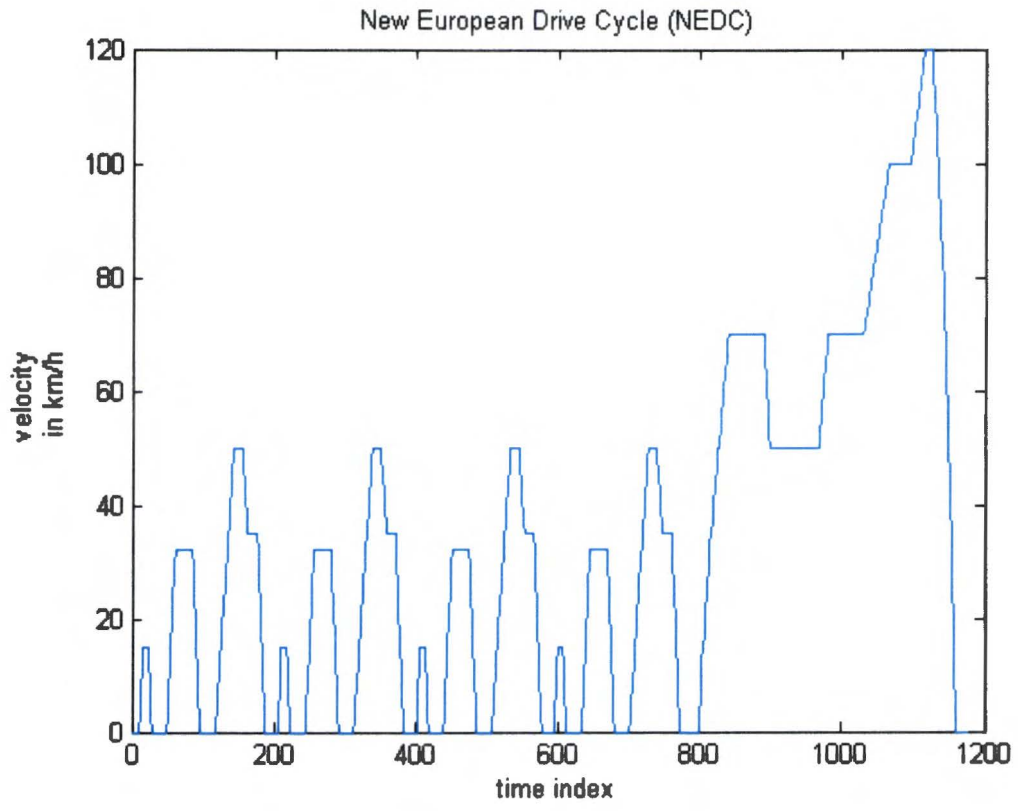

Figure 4 NEDC cycle

The characteristic of the CYC_ARB02 are: 1639 s long thus a drive cycle of 27 minutes, average speed=43.5 mph (70.0 km/h) and the maximum speed=80.3 mph (129.2km/h) (see **Figure 5 CYC_ARB02 cycle**)

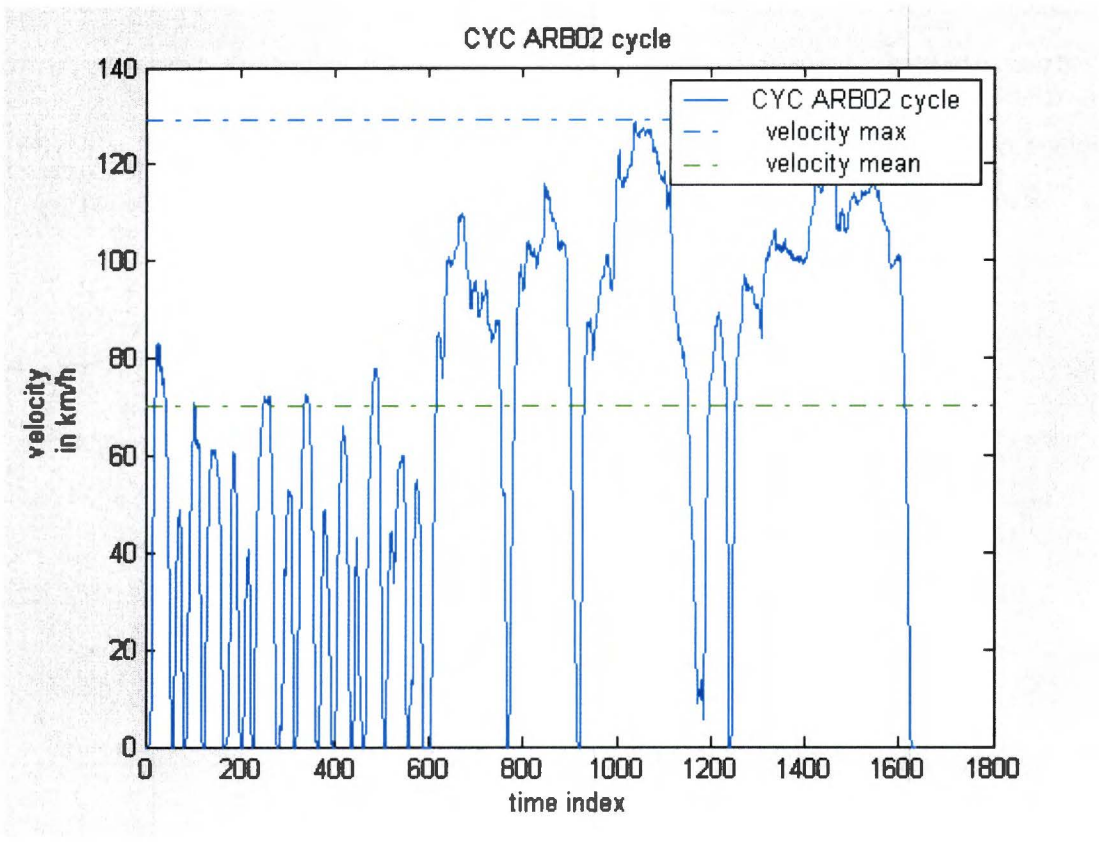

Figure 5 CYC_ARB02 cycle

The Markov model is built by estimating the transition probabilities from the sample cycle data.

The combination of both cycles is simply NEDC followed directly by CYC_ARB02.

NB: We can influence the behavior of the controller by selecting a different combination. Indeed, a different combination will give us a different matrix of transition which influences directly the controller.

2.3.2.2 From deterministic to stochastic model

For a given vehicle speed $v_{veh}(t)$ taking from the combined drive cycle and a selected gear ratio $g_r(t)$, the corresponding deterministic engine speed $\omega(t)$ and torque for the drive train $\tau_d(t)$ are calculated using the following formulas:

$$\omega(t) = \frac{f_r}{\omega_r} g_r(t) v(t)$$

$$\tau_d(t) = \frac{w_r}{f_r} \frac{1}{g_r(t)} F_d(t) \tag{0.44}$$

$$F_d(t) = m\dot{v}(t) + \frac{1}{2}\rho C_d A_d v(t)^2 + mgC_r$$

Afterwards, we assign a grid for speed and torque at each time instant and then the transition probability is estimated from the sample cycle data (ω_{wheel}, τ_d) by counting the number of transition (cf below eq.(0.47)). And we get the stochastic model of the driver power demand and wheel speed.

Quantity	Symbol	Value	Unit
Mass	M	1400	kg
Frontal area	A_d	2.0	m^2
Air drag coefficient	C_d	0.3	-
Rolling resistance	C_r	0.0015	-
Air density	ρ	1.2	kg/m^3
Gravity	G	9.8	m/s^2
Wheel radius	w_r	0.3	m
Final drive ratio	f_r	4.0	-
Gear ratio	g_r	3.4 – 2.1 – 1.4 – 1.0 – 0.77	-

Parameters of the vehicle model

Both variables ω_{wheel} and P_d will be discretized into say 20 values. This gives 20*20=400 values. The state transition matrix will then have 400x400 possible transitions, since the variables are strongly correlated, we expect this matrix to be sparse (lots of transitions are physically impossible).

We define $s = \{\omega_{wheel}, P_d\}$, we called it the stochastic variables because we will see later that the whole state space is composed of those stochastic variables plus a deterministic variable for the energy level of the battery

NB: due to the fact that $P_d(t) = \omega_{wheel}(t) * \tau_d(t)$, we can model τ_d, ω_{wheel} and obtain afterward the transition model of P_d and ω_{wheel}

The wheel speed (directly related to the vehicle speed) and the the torque needed for propulsion are discretized into a finite number of values (20 samples).

We have thus 20*20=400 states for the Markov Chains, where every state corresponds to a different combination of wheel speed and torque and thus the transition matrix has a dimension of 400*400.

The matrix of transition is denoted by P. Where p_{ij} is an element of the matrix (i-th row, j-th column) which determines the probability to go from the state i at present time t to the next state j at time t+1 and of course $\sum_{j=1}^{400} p_{ij} = 1$

In other words

$$\Pr\{s_{t+1} = couple(j) \mid s_t = couple(i)\} \quad \text{i,j=1,2,...400} \tag{0.45}$$

where couple is a vector (400 elements)of the different combination of wheel speed and torque ($couple(i) = \{\omega_{wheel}^{16}, \tau_d^{18}\}$ for example)

P_d is denoted by

$$P_{d,k} = \omega_{wheel,k} * \tau_{d,k} \quad = couple_k^1 * couple_k^2 \tag{0.46}$$

(The first element times the second element of couple where the probability distribution of ω_{wheel} *and* τ_d will be estimated from experimental data (drive cycles)

We use the maximum likelihood estimator, which counts the observation data as

$$\hat{p}_{ij} = \frac{m_{ij}}{m_i^{total}} \text{ if } m_i^{total} \neq 0 \text{ and } \hat{p}_{ij} = 0 \text{ if } m_i^{total} = 0 \tag{0.47}$$

Where m_{ij} is the number of times that the transition from couple (i) to couple (j) occurs and $m_i^{total} = \sum_{j=1}^{400} m_{ij}$ is the total number of times that there is a transition from couple (i).

2.3.3 Fuel consumption map

The second block is a system with inputs (ω, P_d, P_l, P_s) and the output is the consumption of fuel (instantaneous).The establishment of this mapping was explained early in the chapter "vehicle model".This function will be used in the cost function.

2.3.4 The cost function

The third block "cost" implements the function of cost-per-stage:

$$g(\omega_{wheel,k}, P_{d,k}, P_{s,k}, P_{l,k}, SOC_k) = f(\omega_{wheel,k}, P_{d,k}, P_{s,k}, P_{l,k}) + \gamma(\Delta SOC_k)^2 \quad (0.48)$$

where

- $\Delta SOC = SOC_{ref} - SOC_{ref}$, where $SOC_{ref} = 0.625$ (0.49)
- $P_{l,k} = P_l = cst \quad for\ all\ k$ (0.50)
- γ is a positive constant that we have to tune.
- $f(\omega_{wheel,k}, P_{d,k}, P_{s,k}, P_{l,k})$ is the fuelmap defined earlier (see equation(0.40))

There is a Trade-off between the fuel consumption and the deviation of the state of charge. Indeed, as we can see in equation (0.48), the cost function is composed of two terms which are the fuel consumption and the deviation of the state of charge. Depending on the demand of the car's manufacturers, this wants, of course, minimum fuel consumption but also a long life span of the battery, we will apply a gamma such that the optimization on the two parameters is acceptable. If we take a gamma very big, then the controller will give us a policy such that the deviation of the state of charge remains minimum. In the other case, a gamma very small gives more relative importance to the reduction of the fuel consumption.

2.3.5 Battery model

The fourth block represents the battery, we use to model it a simple integrator (it is called first difference in the discrete time case)

$$E_{s,k+1} = E_{s,k} + P_{s,k} * \Delta t \tag{0.51}$$

And express with The State Of Charge (SOC) variable.

$$SOC_{k+1} = SOC_k + \frac{P_{s,k} * \Delta t}{E_{cap} * V * 3600} \text{ x } 100\%$$

where the unity of the variables are

$[SOC] = \%, \quad [P_s] = Watt, \quad [E_{cap}] = Ampere * hour, \quad [\Delta t] = \text{sec}\, onde, \quad [V] = Volt$

CHAPTER III

This chapter gives a detailed description of the controller design and how it has been engineered so an optimal policy can be determined.

3 Controller design: development of the AGENT

The Dynamic Programming equation

One remarkable property of the optimal cost function J^* is that it satisfies the Equation:

$$J^*(x) = \min_{u \in U(x)} \underset{w}{E}\{g(x,u,w) + \alpha J^*(f(x,u,w))\} \tag{0.52}$$

known as the **DP equation** or the **Bellman equation** and that it is the *unique* solution of this equation.

The knowledge of J^* can be used to compute an optimal stationary policy. Indeed, it can be shown that among the stationary policies all and only those which satisfy the following expression are optimal :

$$\mu^*(x) = \arg\min_{u \in U(x)} \underset{w}{E}\{g(x,u,w) + \alpha J^*(f(x,u,w))\} \tag{0.53}$$

We remark that the system dynamics (cf equation (0.1)) and the "cost-per-stage" function (see equ. (0.48)) have to be known to deduce μ^* from J^*.

To implement the agent, we have to solve the problem of optimization like formulated early (cf Chapter I).

We have shown that solving the Bellman equations comes down to solving the infinite horizon problem:

$$J(x_i) = \min_{u \in U(x_i)} [g(x_i,u) + \alpha \sum_{j=1}^{n} m_{ij}(u) J(x_j)], \quad i = 1,2,...,n \tag{0.54}$$

Let the state space S consist of n states denoted by 1,2,…,n:

$S = \{1,2,...,n\}$, $m_{ij}(u)$ denotes the transition probabilities

$$m_{ij}(u) = P(x_{k+1} = j \mid x_k = i, u_k = u), \quad i,j \in S, u \in U(i) \tag{0.55}$$

with $x = \{\omega, P_d, SOC\}$ and $u = P_s$ is the control variable.

For more convenience, we will introduce a notation: $x = \{\omega, P_d, SOC\}$ is composed of stochastic and deterministic variables, we denote by s the stochastic variable

$s = \{\omega, P_d\}$, then $x = \{s, SOC\}$

We rewrite the BELLMAN EQUATION

$$J(\omega, P_d, SOC) = J(s, SOC) = \min_{u \in U(x)} \{g(s, SOC, u) + \alpha \sum_{x'} M(x, x') J(x')\} \quad (0.56)$$

where x' is the next state and g is the function defined in (0.48).

$g(s, SOC, u) = f(s, SOC) + f_2(SOC)$ with f is the fuelrate defined early (cf (0.40)) and

$f_2 = \gamma (SOC_{ref} - SOC)^2$

The function $f_2(SOC)$ is easy to implement and the computation of the fuelrate function needs the model of the alternator and the engine (cf [5]).

Then we have to implement this equation.

The matrix M of 400 000*400 000 is too big to implement thus we have to find a trick to avoid this problem. Solution of the problem, we can take advantage of the fact that the variable SOC is deterministic, then the dimension of the matrix is reduced with a dimension equals to 400*400. We can rewrite the BELLMAN EQUATION like that:

$$J(s, SOC) = \min_{u \in U(x)} \{f(s, u) + f_2(SOC) + \alpha \sum_{s'} P(s, s') J(s', SOC + \frac{u}{E_{cap}} \Delta t)\} \quad (0.57)$$

Rewriting of the mappings and the Q-function using a MDP structure

By using the MDP structure to rewrite the mapping T defined earlier[6]

For any function, $J : S \to \Re$

We define the function obtained by applying the DP mapping to J, and we denote it by

$$(TJ)(i) = \min_{u \in U(i)} [g(i, u) + \alpha \sum_{j=1}^{n} m_{ij}(u) J(j)], \quad i = 1, 2, ..., n \quad (0.58)$$

Similarly, for any function $J : S \to \Re$ and any action function $\mu : S \to C$, we denote

---

[5] Cf chapter 1.2.1. VEHICLE MODEL (Introduction of physics variables)

[6] cf chapter 1.1.2Definition of two mappings

$$(T_\mu J)(i) = g(i,\mu(i)) + \alpha \sum_{j=1}^{n} m_{ij}(\mu(i))J(j), \quad i = 1,2,...n \tag{0.59}$$

We will denote by T^k the composition of the mapping T with itself k times; that is for all k we write

$$(T^k J)(x) = (T(T^{k-1}J))(x), x \in S \tag{0.60}$$

and with k=0 , we have $(T^0 J)(x) = J(x), x \in S$ (0.61)

Similarly with the mapping T_μ , that is the same definition, you have just to add the index μ .

$$Q(i,u) = g(i,u) + \alpha \sum_{j=1}^{n} m_{ij}(u)J(j), \quad i = 1,2,...,n \tag{0.62}$$

3.1 Discretization of the variables

The state vector $x = \{SOC, \omega_{wheel}, P_{dem}\}$ (0.63)

forms a three-dimensional state space S .

NB: We assume that the variable P_l is a constant, thus we do not consider this in the state vector (cf(0.36))

We discretised the torque needed for propulsion as

$$\tau_d \in \{\tau_d^1,...,\tau_d^{N_{td}}\} = \{-448.15,-414.38,...,193.51\}\ in\ Nm \tag{0.64}$$

and the wheel speed (directly related to the vehicle speed) as

$$\begin{aligned} \omega_{wheel} \in \{\omega_{wheel}^1,...,\omega_{wheel}^{N_{\omega_{wheel}}}\} &= \{0,\ 29.11,...,553.10\}\ (rad/\sec) \\ &= \{0,\ 277.98,...,5281.8\}\ (rpm) \end{aligned} \tag{0.65}$$

and the State of Charge as

$$SOC \in \{SOC^1,...,SOC^{Nsoc}\} = \{0.6,...,0.65\}(\%) \tag{0.66}$$

The storage power in the battery, which is the control variable, is denoted by P_s and is also discretised into

$$P_s = \{P_{s\_\min},...,P_{s\_\max}\} \tag{0.67}$$

The computation of P_{s_min} *and* P_{s_max} is done as follows:

$$P_s \overset{i}{=} P_b \overset{ii}{=} P_e - P_l$$

i) If we assume that the losses in the battery are very small (cf eq. (0.39)),this assumption is respected because we will study the deviation of the state of charge in the region 60% and 65% which is the region where the losses are minimal

ii) thanks to eq. (0.43)

Thus $P_s = [P_{s\_min}, ..., P_{s\_max}]$

Where $P_{s\_min} = \min_{P_e}(P_e - P_l) = 0 - P_l = -P_l$

$P_{s\_max} = P_{e\_max} - P_l$ with P_{e_max} is a function of ω_{wheel} (cf **Figure 7 Alternator power limitation**)

Finally, the grid for the action variable P_s is composed of 16 samples and is equals to:

$P_s(\omega_{wheel}, SOC) = \{-P_l, ..., P_{e\_max} - P_l\}$ but this is only applied if the upper and lower bounds of SOC (60 % and 65 %) are not reach with this grid.

Of course depending on the SOC level, the action is restricted.

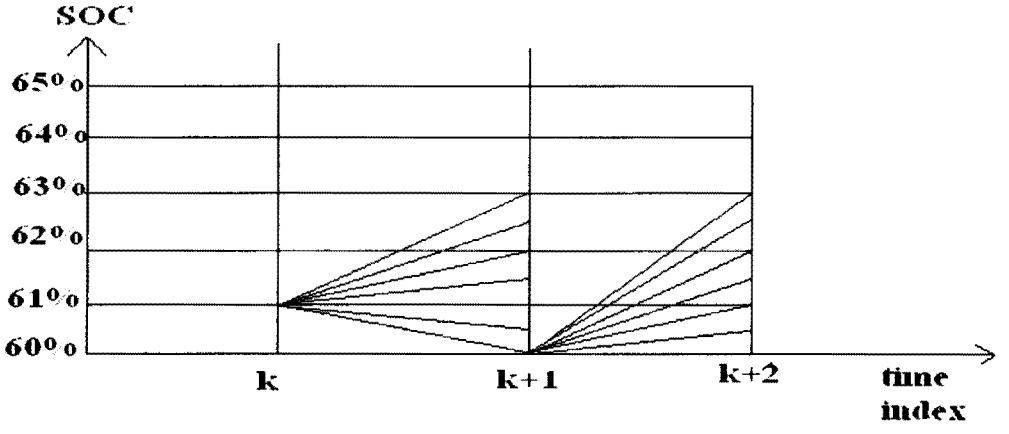

Figure 6 Restriction of the action variable's range

As we can see in the Figure 6, at time index k, with a SOC=61%, we can use negative value for Ps, corresponding to discharge of the battery. At time index k+1, with a SOC=60% (The lower bound), the action are limited to positive values. And the same limitation occurs when the SOC is near to 65% (the upper bound)[7]

3.1.1 Limitation of the alternator setpoint

Depending on the wheel speed, the maximum power that the alternator can deliver is limited (cf **Figure 7 Alternator power limitation**)

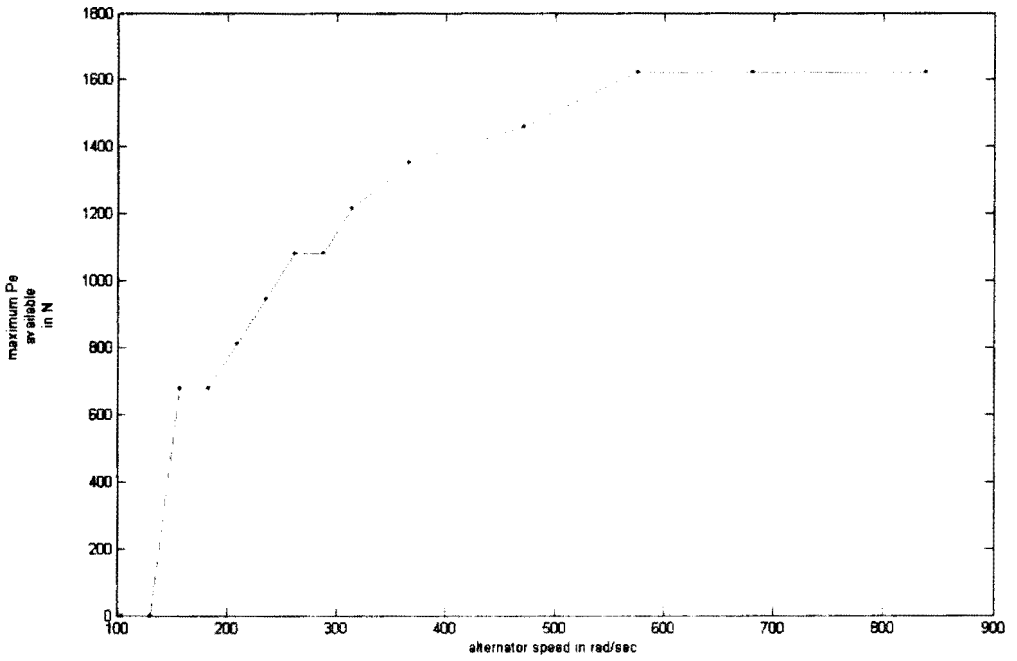

Figure 7 Alternator power limitation

The alternator speed is related to the engine speed as follows:

$$\omega_{eng} = \frac{\omega_{alt}}{alt\_ratio} \text{ with alt\_ratio=2.52}$$

[7] This restriction is implemented in the controller

3.2 The dimension of the state space

3.2.1 The set of control variables

Knowing that the maximum power offered by the system is 1600Watts, we have chosen to take 16 possible discrete values as input (so-called actions) with a STEP SIZE of 100 Watts.

With $P_s(\omega_{wheel}, SOC) = \{-P_l, ..., P_{e\_max} - P_l\}$

Considering $P_l = 500\,Watts$ and by looking at « Alternator power limitation » results, the possible discrete actions are

$P_s = [-500, -400, -300, ..., 1000, 1100]$

3.2.2 The accuracy of the SOC variable's grid

The characteristic of the battery are

$E_{cap} = 60Ah$

$V = 12\,Volts$

$P_{s,step} = 100\,Watts$

To charge an empty battery of 60Ah and with a voltage of 12V, we need an energy of 12 * 3600*60=2 592 000 Joules

Then, due to the fact that we study only the region between 60 % and 70 % (first choice...), only 259 299 Joules are needed.

The step size of the action variable is equal to 100 Watts[8], then 1000 samples is enough to be able to follows the variation of the battery's level.

In summary, the dimension of our first state space is equal to 400 000.

3.2.3 Definition of the subset of "observed" states

Let S_m, a subset of the sub-state space $s = \{\omega_{wheel}, P_{dem}\}$ (the state of stochastic variable) containing all the states being observed in the drive cycle, namely all the states whose transition probabilities are different to zero. Due to the fact that the drive cycles available only contain a limited number of data, it is expected that some physical transition -that may occur on the road- could not have been observed in the training data of the drive

[8] reminder: 1 Joule= 1 Watt / Second

cycle. Therefore, the probabilities of some unseen transitions will be zero. The subset S_m is smaller than the state space. For the sake of being more accurate, the 's' cardinal is equal to 400 while the sub-state S_m is equal to 157[9].

3.2.4 Reduction of the state space dimensions

The reduction of the number of states in the state-space has been possible thanks to a set of 2 parameters fine-tuning.

- ✓ The first parameter is the sampling time Δt defined in the battery model. It was put to 1 second in the first place, we have decided to double it which leads to a reduction of the number samples by half for the State of Charge Variable (SOC). Another consequence is the increase of missing data in the transition matrix (243 empty lines instead of 235). Indeed, increasing the sampling time leads to a reduction of the number of observed transitions because the length of the drive cycle is also reduced by half.
- ✓ The second parameter is the level of battery constraints. Rather than to allow the variation of STATE OF CHARGE in the range of [60, 70 %], we have limited the upper limit to 65% so that the range becomes [60, 65%].

This fine-tuning will allow to reduce of the number of states from 165*1000=165000 to 157*250=39250 [with $\#S_m = 157 \quad and \quad \#SOC = 250$]

3.3 The matrix of transition of the whole state space

The next state $x_{k+1} = f(x_k, u_k, w_k), \quad k = 0,1,...,$ is given by (0.45) and(0.51).

To be more precise, we should write $s_{k+1} = f(s_k, w_k), \quad k = 0,1,...$ with $s = \{\omega_{wheel}, P_d\}$ and

$E_{s,k+1} = E_{s,k} + u_k \Delta t$ with $SOC_k = {E_s}/{E_{cap}} * 100\%$

We can spell out by drawing the Matrix of transition (space of 3 dimensions)

[9] 157 samples after the reduction explain in chapter 3.2.4

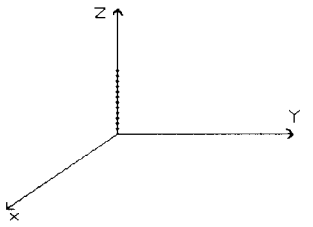

Figure 8 The matrix of transition

$$z = \{SOC_0, SOC_1, ..., SOC_{N_{SOC}}\} \tag{3.17}$$

$$x = \{couple^1(t), couple^2(t), ..., couple^{N_{\omega_{wheel}} * N_{\tau_d}}(t)\} \tag{3.18}$$

$$y = \{couple^1(t+1), couple^2(t+1), ..., couple^{N_{\omega_{wheel}} * N_{\tau_d}}(t+1)\} \tag{3.19}$$

Where the elements (x,y,z) is the probability to go from the state x to the next state y, with a new state of charge of z (the previous state of charge is not useful)

NB: the probability to go from a state $couple(i) = \{\omega_{wheel}^m, \tau_d^n\}$ to a state $couple(i) = \{\omega_{wheel}^o, \tau_d^p\}$ is independent of the State of Charge, thus, the matrix of 3 dimensions is the superposition of the same matrix of transition (the plane x-y is thus the same plane for all z).

3.4 Policy iteration

This algorithm is impossible to implement in our case, because of the number of state (reminder: the cardinal of the state space is equal to 100 000). Unlike the value iteration algorithm , we have not found a trick such that we do not have to use the matrix of transition of 100 000*100 000

For the policy iteration, during the policy evaluation step, we have to solve the equation below[10]:

$$J_\mu = (I - \alpha P_\mu)^{-1} g_\mu, \tag{0.68}$$

As already explained in chapter 1, the main weakness of this algorithm is that it requires to solve equation (1.29). The dimension of this system is equal to the number of states and thus if the state space is large, as in our case, this algorithm does not stay attractive anymore.

3.5 Value iteration

The value iteration algorithm:

A: *Initialize random* $J(i)$ *for all* $i \in S$

Repeat indefinitely

$errorMax=0$

B: *Repeat once for all* $x_i \in S_m$ (39250 *elements in* S_m)

C: $$J_{k+1}(i) \leftarrow \min_{u \in U(i)} \{g(i,u) + \alpha \sum_{j=1}^{n} p_{ij}(u) J_k(j)\}.$$

$$errorMax = \max(errorMax, |J_{k+1}(i) - J_k(i)|)$$

$J_k \leftarrow J_{k+1}$

D: *If* $errorMax < \varepsilon$ *then stop the algorithm and use* J *as an approximation of* J^*

Explanation:

A: The first step is to initialize a random $J(i)$ *for all* $i \in S_m$ and in our case $J(i) = 0$ *for all* $i \in S_m$.

B: Thus, we restrict the update rule to the state observed during the experimental data processing. The main reason of this choice is that, we do not have to find a policy for those unseen state because we have limited our study to the OFF_LINE case.

C: We do not have to apply this mapping where a matrix of transition of 39250*39250 should be used, but we apply the trick explained earlier in the chapter III. The advantage is taken from the fact that a variable is deterministic, then we can use a matrix of transition of 400*400 instead of 100 000*100 000:

$$J(s,SOC) = \min_{u \in U(x)} \{f(s,u) + f_2(SOC) + \alpha \sum_{s'} P(s,s') J(s', SOC + \frac{u}{E_{cap}} \Delta t)\} \qquad (0.69)$$

[10] cf page 14 for the definition of the variable

D:Stopping criterion.We stop the algorithm

when $|J_{k+1}(x_i) - J_k(x_i)| < \varepsilon \quad for \quad all \quad x_i \in S_m$, i.e. when the delta is less than ε between 2 consecutive approximations.

In other words, the parameter ε triggers the stopping of the algorithm. A too large value of ε can lead to important errors in the computation of J^* while a too small value can cause the algorithm to iterate too many times.

The precision of the approximation of J^* so computed by the value iteration algorithm is well controlled independently of the initial condition J_0 of the iterative algorithm. It can be proved that the approximation of J^* computed by using the value iteration algorithm lies necessarily in the interval:

$$\left[J^*(i) - \frac{\alpha * \varepsilon}{1-\alpha}, J^*(i) + \frac{\alpha * \varepsilon}{1-\alpha}\right] = \left[J^*(i) - \frac{0{,}007}{3}, J^*(i) + \frac{0{,}007}{3}\right] for\ all\ i \in S \qquad (3.20)$$

Where α =0.7 is the discount factor and ε =1e-3

CHAPTER IV

4 RESULTS

Two representative driving cycles are chosen to construct the observation samples of the stochastic system. The Markov model is built by estimating the transition probabilities from the sample cycle data as described in Section 1.1. The matrix of transition is sparse[11]; this is due to two reasons. The first one is that we do not have enough data because the two representatives driving cycles can not take into account all the physical transition. The second reason is that a lot of transitions are physically impossible.

4.1 Constructing random drive cycles

The principle: we start at the state $\omega = 0, \tau = 0$ and we generate a vector of random numbers ranging between 0 and 1.Knowing that the sum of the elements of a line is always equal to one. Given a random number v, ranging between 0 and 1, we make the sum of the elements of the line (corresponding to the present state) until the number obtained is equal or superior to v .

The corresponding column is the index of the next state and so on with the new state.

The matrix of transition is SPARSE because a lot of transitions are physically impossible. Then a lot of line are empty (243/400%=60% of empty lines).

There are 243 isolated states.

In other words, if the process starts within a recurrent class[12] (here, 243 different recurrent class), it stays within that class. A problem to avoid when we generate the random drive cycle.

---

[11] 60% of empty lines in the matrix of transition

[12] Cf APPENDIX B: On finite-state Markov Chains

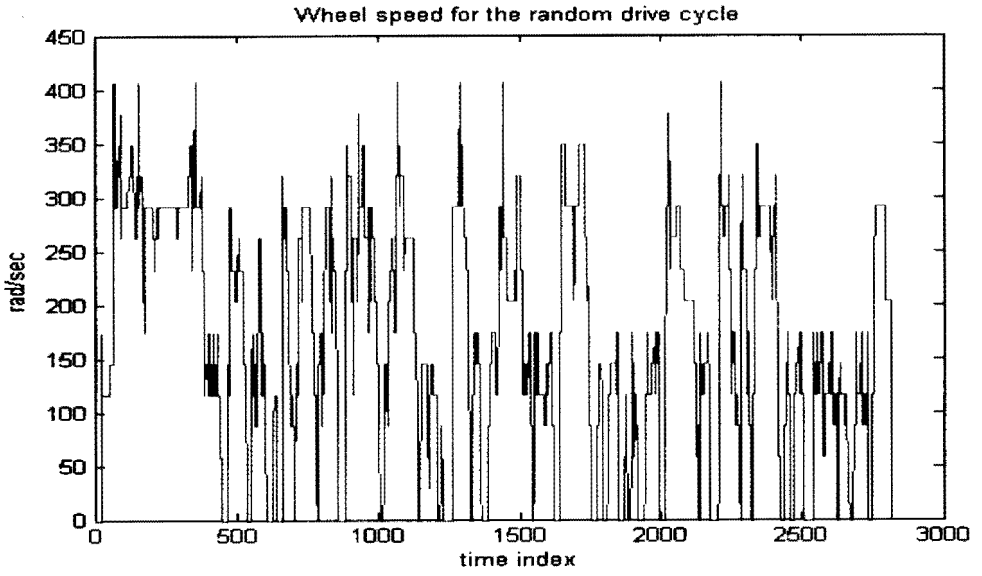

Figure 9 wheel speed of the random drive cycle (RDC)

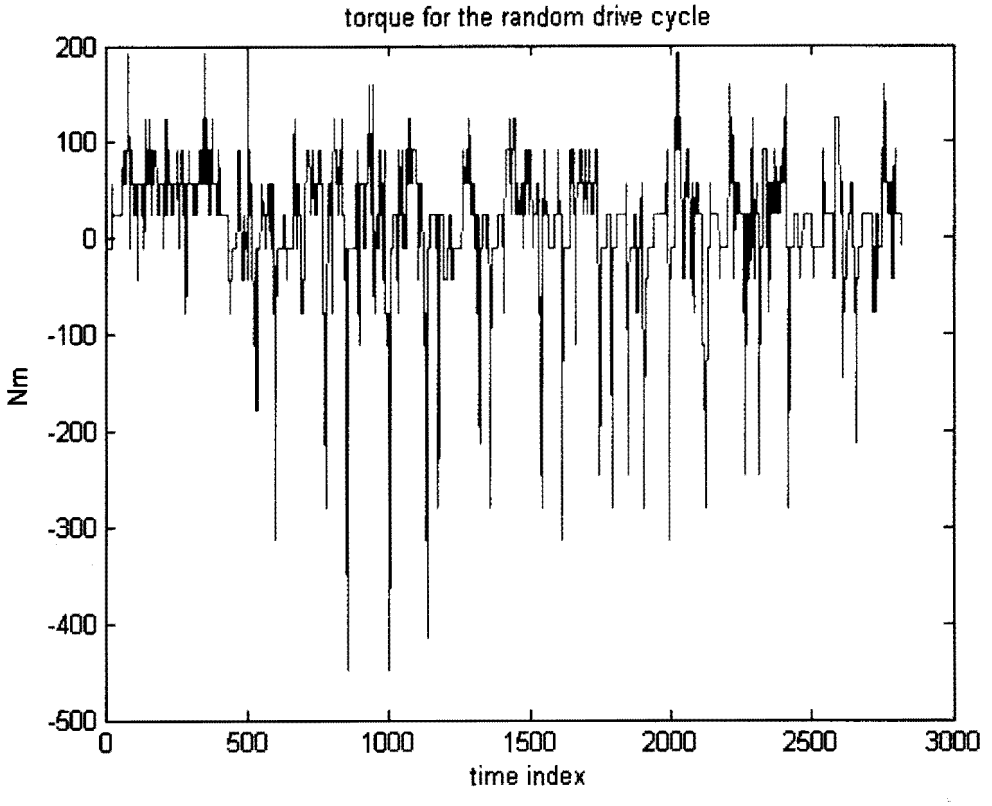

Figure 10 torque of the RDC

As we can see in Figure 7, we have a limitation for positive torque, indeed, the vehicle can maximum deliver 200Nm.But for negative torque; we can see that the magnitude of the "drive train" torque for negative value can be very important. This is due to the fact that when the maximum negative torque is smaller than the desired deceleration torque, we can use the brakes.

4.2 Trade-off between fuel consumption and the life span of the battery

The algorithm was applied with two different values of the tuning parameter γ. Before starting to comment the result, we should display a useful result about the variation of the Fuel consumption with respect to the control signal Ps.

As we can see in Figure 11, the fuel consumption increases with Ps. This property allows us to predict the optimal policy.

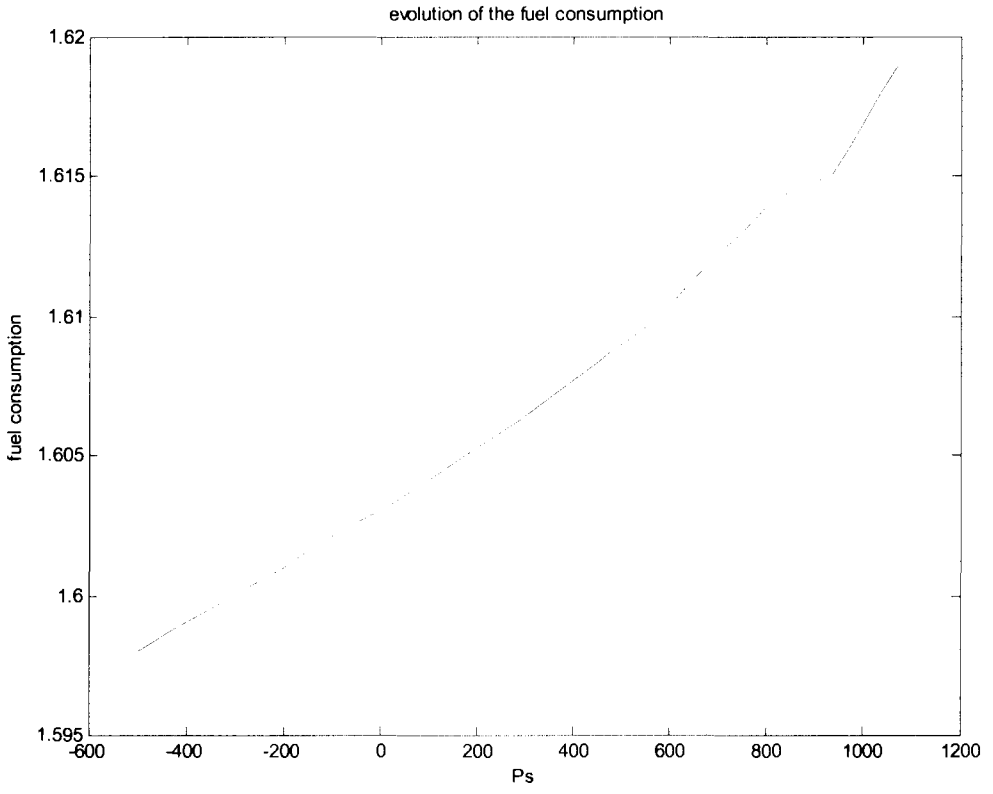

Figure 11 Fuelconsumption with respect to the control signal

Reminder, the cost function g:

$$g(h_k, u_k, SOC_k) = f(h_k, P_{s,k}) + \gamma(\Delta SOC_k)^2$$
$$where\ h_k = \{\omega_{wheel,k}, P_{d,k}, P_{l,k}\} \quad and \quad u_k = P_{s,k} \tag{4.1}$$

The parameter γ allows us to give control what we want. Indeed, a γ very small will give more importance to the fuel consumption.

First CASE: A γ very small, thus we try to minimize the fuel consumption and we do not mind about the deviation of the state of charge of the battery.

The optimal policy will always take power from the battery, corresponding to a negative control signal Ps, until the lower bounds is reached.

Second CASE: a big γ, we try to keep the deviation of the SOC minimum. Thus we care about the battery and try to maximize his life span. The optimal policy, in this case, should give us the same result as we can see in Figure 12 (gamma=inf)

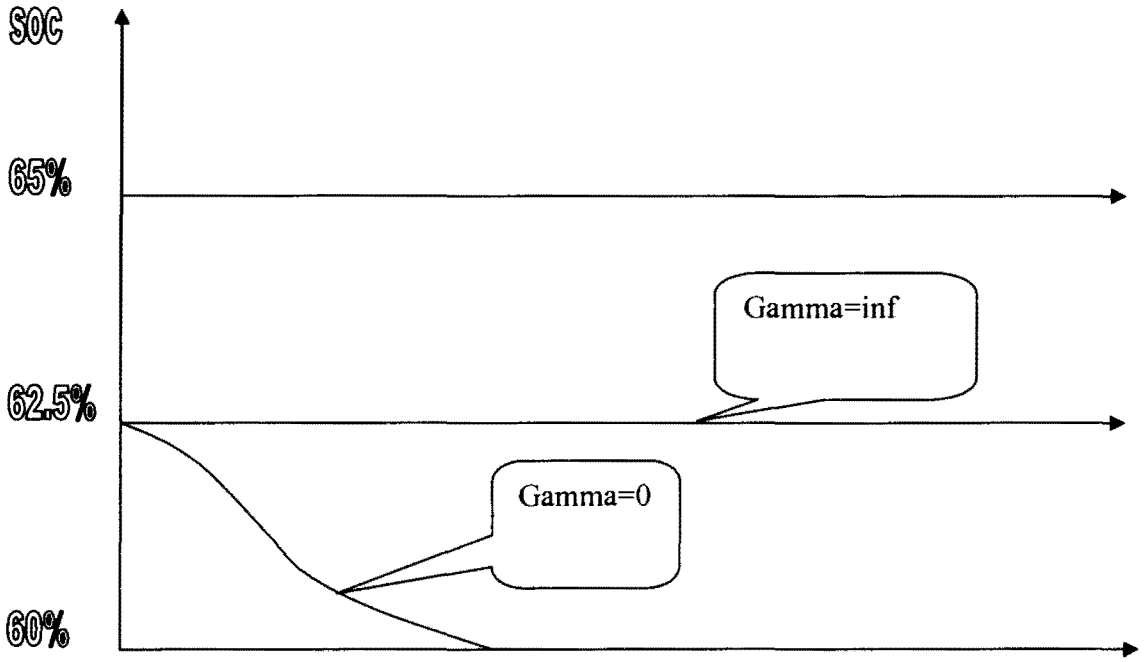

Figure 12 Evolution of the SOC by applying the optimal policy

NB : We talk about the life span of the battery because we assume that the fact of charging and discharging the battery influence his life span.

Third CASE: Gamma such that the two terms in the cost function have the same weight. Thus in this case, the policy will try to keep the SOC near to the level reference but also try to minimize the fuel consumption, thus decrease the level of the battery. The evolution of the SOC should be situated between the two curves of the two first cases.

CONCLUSION

One advantage of stochastic dynamic programming compared to deterministic dynamic programming is that the minimization is performed not for a predetermined drive cycle but in a stochastic, "average" sense over a class of trajectories from an underlying Markov chain drive cycle generator.

The dimension of the state space was a problem in this thesis. Stochastic dynamic programming is computational very demanding and due to this large state space, no solution has been found[13].

We can influence the behavior of the controller by selecting a typical combination of driving cycles and use them to learn the Markov transition matrix.

The fact that the cost function covers two terms (fuel and SOC) gives freedom to influence the behavior of the controller. Depending on the value of the tuning factor gamma, the fuel consumption was minimized or the deviation of the state of charge or a trade-off between those two quantities.

Topics for future research: Reconsider the current Matlab implementation for solving the Stochastic dynamic programming problem, such that a feasible solution is calculated. Furthermore, the application of smoothing techniques will be necessary to fill in the empty lines of the Markov transition matrix. This way, the controller is able to provide a control action in all situations that occur in real-world driving situations.

[13] A policy was found but due to the lack of time, not developed.

APPENDIX A: On finite-state Markov Chains

B.1 STATIONARY MARKOV CHAINS

A square $n \times n$ matrix $[p_{ij}]$ is said to be a *stochastic* matrix if all its elements are nonnegative, that is , $p_{ij} \geq 0, i, j = 1,...,n,$ and the sum of the elements of each of its rows is equal to 1, that is, $\sum_{j=1}^{n} p_{ij} = 1$ for all $i = 1,...,n.$

Suppose we are given a stochastic $n \times n$ matrix P together with a finite set of state $S = \{1,...,n\}$. The pair (S,P) will be referred to as a *stationary finite-state Markov Chain.*

A transition is made from state x_0 to a new state $x_1 \in S$ in accordance with a probability distribution specified by P as follows. The probability that the new state will be *j* is equal to p_{ij} whenever the initial state is i, *that* is ,

$$P(x_1 = j \mid x_0 = i) = p_{ij}, \qquad i, j = 1,...,n. \tag{B.1}$$

Similarly, subsequent transitions produce states $x_2, x_3,...$ in accordance with

$$P(x_{k+1} = j \mid x_k = i) = p_{ij}, \qquad i, j = 1,...,n. \tag{B.2}$$

The probability that after the *k*th transition the state x_k will be *j,* given that the initial state x_0 is *i,* is denoted by

$$p_{ij}^k = P(x_k = j \mid x_0 = i), \qquad i, j = 1,...,n. \tag{B.3}$$

A straightforward calculation shows that thses probabilities are equal to the elements of the matrix P^k (P raised to the *k*th power), in the sense that p_{ij}^k is the elements in the *i*th row and the *j*th column of P^k :

$$P^k = [p_{ij}^k]$$

B.2 CLASSIFICATION OF STATES

Given a stationary finite-state Markov chain (S,P), we say that two state *i* and *j* *communicate* if there exist two positive integers k_1 and k_2 such that $p_{ij}^{k_1} > 0$ and $p_{ij}^{k_2} > 0$. In words, state *i* and *j* communicate if one can be reached from the other with positive probability.

Let $\tilde{S} \subset S$ be a subset of states such that:

1. All states in $\tilde{S}$ communicate.

2.If $i \in \tilde{S}$ and $j \notin \tilde{S}$, then $p_{ij}^{k} = 0$ for all k.

Then we say that $\tilde{S}$ forms a *recurrent class* of states.

If S forms by itself a recurrent class (i.e., all states communicates with each other), then we say that the Markov chain is *irreductible*. It is possible that there exist several recurrent classes, it is what we have in our problem, we have 240 states of recurrent class, where each subset of states is a singleton, on other words, and we have 240 subset S with one element for each state. Reminder, we have 240 isolated states due to the fact we don t have enough data from our drive cycle

References

[1] Chan-Chiao Lin,Huei Peng , and J.W.Grizzle ''A stochastic Control Strategy for Hybrid Electric Vehicles'' *Proc. Of the 2004 American Control Conference*, Boston Massachusetts June 30 –July 2,2004

[2] http://www-2.cs.cmu.edu/~awm/tutorials/mdp09.pdf

[3]J.T.B.A Kessels,W.P.M.H Heemels,P.P.J van den Bosch,Michiel Koot,Bram de Jager "Energy Management Strategies for Vehicle Power Nets "

[4]D.P. Bertsekas *dynamic Programming and Optimal control* . Athena Scientific, Belmont, MA, 1995

[5]Richard S. Sutton and Andrew G. Barto "Reinforcement learning : An introduction"