Oil & Gas Science and Technology – Rev. IFP, Vol. 65 (2010), No. 1, pp. 91-102

DOI: 10.2516/ogst/2009020

IFP International Conference
Rencontres Scientifiques de l'IFP

Advances in Hybrid Powertrains
Évolution des motorisations hybrides

On Implementation of Dynamic Programming for Optimal Control Problems with Final State Constraints

O. Sundström[1,2*], D. Ambühl[1] and L. Guzzella[1]

1 Department of Mechanical and Process Engineering ETH Zurich, 8092 Zurich - Switzerland
2 Empa. Swiss Federal Laboratories for Materials Testing and Research, 8600 Dübendorf - Switzerland
e-mail: sundstroem@imrt.mavt.ethz.ch - daniel.ambuehl@alumni.ethz.ch - guzzella@imrt.mavt.ethz.ch

* Corresponding author

Résumé — À propos de l'implémentation de la programmation dynamique pour des problèmes de contrôle optimal avec des contraintes sur l'état final — Cette publication présente certains problèmes concernant l'implémentation de la programmation dynamique pour le contrôle optimal d'un modèle dynamique scalaire, comme par exemple la gestion énergétique d'un véhicule hybride électrique. Une étude sur la résolution de l'espace d'état discrétisé souligne le besoin d'une implémentation minutieuse. Une nouvelle méthode qui permet de traiter des problèmes numériques d'une façon adéquate est présentée. Cette méthode permet particulièrement de résoudre des problèmes numériques engendrés par de forts gradients dans la fonction coût optimale. Ces gradients se situent surtout aux bornes de l'ensemble d'états atteignables. La méthode proposée améliore non seulement la précision de l'optimum global, mais permet aussi de réduire la résolution de l'espace d'état en conservant la précision. Le nombre de calculs nécessaire pour évaluer l'optimum global est ainsi considérablement réduit. Cela permet des applications ultérieures de la programmation dynamique pour des véhicules hybrides électriques comme par exemple des études paramétriques extensives.

Abstract — ***On Implementation of Dynamic Programming for Optimal Control Problems with Final State Constraints*** — *In this paper we present issues related to the implementation of dynamic programming for optimal control of a one-dimensional dynamic model, such as the hybrid electric vehicle energy management problem. A study on the resolution of the discretized state space emphasizes the need for careful implementation. A new method is presented to treat numerical issues appropriately. In particular, the method deals with numerical problems that arise due to high gradients in the optimal cost-to-go function. These gradients mainly occur on the border of the feasible state region. The proposed method not only enhances the accuracy of the final global optimum but also allows for a reduction of the state-space resolution with maintained accuracy. The latter substantially reduces the computational effort to calculate the global optimum. This allows for further applications of dynamic programming for hybrid electric vehicles such as extensive parameter studies.*

INTRODUCTION

Developing and optimizing hybrid electric powertrains include several tasks, such as energy management strategy design and component dimensioning. Both when designing the energy management strategy and when dimensioning the powertrain components it is a clear advantage if the global optimal fuel consumption is known for a given configuration and drive cycle. Knowing the global optimal fuel consumption allows a proposed control system to be evaluated with respect to the global optimum. This evaluation must be carried out for the same final state-of-charge using the proposed controller and the global optimal control. As a result, the optimization problem includes final state constraints. Furthermore, hybrid vehicles inherently include constraints on the battery state-of-charge since a too high or too low state-of-charge can damage the battery. What is more, the control signal which decides the power split between electric and thermal driving also includes constraints due to power limits of the two power sources. The optimal control problem for deciding the power split in a hybrid electric vehicle therefore includes final state constraints, state constraints, and control signal constraints.

For the energy management strategy design, and also for powertrain dimensioning, the hybrid vehicle model can be simplified, with maintained accuracy, and reduced to having the battery state-of-charge as the only dynamic state variable [1].

To calculate the optimal control signal trajectory for a hybrid vehicle model with a single state variable, including state and input constraints, most studies [2, 3] use the dynamic programming algorithm [4, 5]. The theory behind dynamic programming as a tool for calculating the optimal control is relatively simple. However, numerical problems arise when implementing the algorithm. In general, if these problems are not treated, numerical errors can have a major impact on the final result. This paper deals with numerical problems that arise due to high gradients in the optimal cost-to-go function which occur on the boundary of the feasible state region. One method for handling state constraints when using dynamic programming is to apply a penalty scheme for infeasible states [6, 7].

This paper presents a new method for including final state constraints. The proposed method is compared to using a penalty scheme for infeasible states when using dynamic programming for two different dynamic systems. First, the issues are emphasized for the well-known optimal control problem of a fishery based on a Lotka-Volterra system [8]. The proposed method is compared to the penalty scheme and to the analytic optimal solution of the Lotka-Volterra fishery problem. Secondly, numerical issues are investigated that are associated with the dynamic programming algorithm for solving the optimal control problem of a parallel hybrid electric vehicle on a given drive cycle. The numerical errors in the hybrid vehicle case are then evaluated for the proposed method and are compared to the penalty scheme.

The proposed method improves the dynamic programming only if the optimal state trajectory is close to the bounds of the feasible region at some points. However, this is typically the case for constrained optimal control problems such as the hybrid electric vehicle problem.

1 OPTIMAL CONTROL PROBLEM

In this paper a special class of optimal control problems is studied, namely problems with fixed final time and a partially constrained final state. Furthermore, the considered problems are assumed to include state constraints and input constraints. What is more, the dynamic systems in this study include only a single state variable, and the disturbances are assumed to be perfectly known. In summary, this problem can be written as an optimal control problem

$$\min_{u(t)} J(u(t)) \tag{1}$$

s.t.

$$\dot{x}(t) = F(x(t), u(t), t) \tag{2}$$

$$x(0) = x_0 \tag{3}$$

$$x(t_f) \in [x_{f,min},\ x_{f,max}] \tag{4}$$

$$x(t) \in \mathcal{X}(t) \tag{5}$$

$$u(t) \in \mathcal{U}(t) \tag{6}$$

where

$$J(u(t)) = G(x(t_f)) + \int_0^{t_f} H(x(t), u(t), t)dt \tag{7}$$

is the cost functional.

2 DYNAMIC PROGRAMMING

This section gives a brief overview of the deterministic dynamic programming algorithm [4], which throughout this study is referred to as dynamic programming (DP). Since dynamic programming is a numerical algorithm used here to solve a continuous control problem, the continuous-time model (2) must be discretized. Let the discrete-time model be given by

$$x_{k+1} = F_k(x_k, u_k), \qquad k = 0, 1, \ldots, N-1 \tag{8}$$

with the state variable $x_k \in \mathcal{X}_k$ and the control signal $u_k \in \mathcal{U}_k$. Furthermore, assume that the disturbance is perfectly known in advance and at every time instance k.

2.1 Basic Algorithm

Let $\pi = \{\mu_0, \mu_1, \ldots, \mu_{N-1}\}$ be a control policy. Further let the discretized cost of Equation (7) using π with the initial state $x(0) = x_0$ be

$$J_\pi(x_0) = g_N(x_N) + \phi_N(x_N) \ldots + \sum_{k=0}^{N-1} h_k(x_k, \mu_k(x_k)) + \phi_k(x_k) \tag{9}$$

where $g_N(x_N) + \phi_N(x_N)$ is the final cost. The first term $g_N(x_N)$ corresponds to the final cost in Equation (7). The second term is the additional penalty function $\phi_N(x_N)$ forcing a partially constrained final state (4). The function $h_k(x_k, \mu_k(x_k))$ is the cost of applying $\mu_k(x_k)$ at x_k, according to $H(x(t), u(t), t)$ in Equation (7). The state constraints (5) are enforced by the penalty function $\phi_k(x_k)$ for $k = 0, 1, \ldots, N-1$.

The optimal control policy π^o is the policy that minimizes J_π

$$J^o(x_0) = \min_{\pi \in \Pi} J_\pi(x_0) \tag{10}$$

where Π is the set of all admissible policies.

Based on the principle of optimality [4], dynamic programming is the algorithm which evaluates the optimal cost-to-go[(1)] function $\mathcal{J}_k(x^i)$ at every node in the discretized state-time space[(2)] by proceeding backward in time:

1. End cost calculation step

$$\mathcal{J}_N(x^i) = g_N(x^i) + \phi_N(x^i) \tag{11}$$

2. Intermediate calculation step for $k = N-1$ to 0

$$\mathcal{J}_k(x^i) = \min_{u_k \in \mathcal{U}_k} \{h_k(x^i, u_k) + \phi_k(x^i) \ldots + \mathcal{J}_{k+1}(F_k(x^i, u_k))\} \tag{12}$$

The optimal control is given by the argument that minimizes the right-hand side of Equation (12) for each x^i at time index k of the discretized state-time space.

The cost-to-go function $\mathcal{J}_{k+1}(x)$ used in Equation (12) is evaluated only on discretized points in the state space. Furthermore, the output of the model function $F_k(x^i, u_k)$ is a continuous variable in the state space which can be between the nodes of the state grid. Consequently, the last term in Equation (12), namely $\mathcal{J}_{k+1}(F_k(x^i, u_k))$ must be evaluated appropriately. There exist several methods of finding the appropriate cost-to-go function $\mathcal{J}_{k+1}(F_k(x^i, u_k))$ such as using a nearest-neighbor approximation or using an interpolation scheme. Throughout this study, linear interpolation of the cost-to-go function $\mathcal{J}_{k+1}$ is used to account for the problem of the discretized state space.

The output of the algorithm (11-12) is an optimal control signal map. This map is used to find the optimal control signal during a forward simulation of the model (8), starting from a given initial state x_0, to generate the optimal state trajectory. In the map the control signal is only given for the discrete points in the state-space grid. The control signal is therefore interpolated when the actual state does not coincide with the points in the state grid.

2.2 Numerical Issues on the Boundary Line

When implementing the algorithm numerical errors must be considered and minimized. One issue to consider is the definition of the cost function for infeasible states and inputs. Infeasible states and inputs are of course infinitely expensive and should therefore have infinite cost $\phi_k(x^i \notin \mathcal{X}_k) \to \infty$ for $k = 1, \ldots, N$ since the defined objectives (such as final state constraints and model limitations) cannot be achieved. When using infinite cost for such states, some substantial numerical errors occur due to the discretization of time and state space.

Define the set of reachable states Ω_k^i over one time-step by using all admissible inputs and starting at a given state x^i at time k

$$\Omega_k^i = \{x | x = F_k(x^i, u) \ \forall\ u \in \mathcal{U}_k\} \tag{13}$$

Consider the grid point/time step domain in Figure 1 (bottom) and the fact that the DP algorithm is calculating the cost-to-go for the state x^i at time $k+1$. If an infinite cost was used for infeasible states together with a linear interpolation, the feasible part of Ω_{k+1}^i would use an interpolation between an infinite cost-to-go $\mathcal{J}_{k+2}(x^i)$ and a finite cost-to-go $\mathcal{J}_{k+2}(x^{i+1})$. As a result, the cost-to-go for x^i at time $k+1$ becomes infinite, *i.e.*, $\mathcal{J}_{k+1}(x^i) \to \infty$, although the grid point $\{k+1,\ i\}$ lies perfectly within the feasible domain.

Now consider the algorithm at time k and the step of calculating the cost-to-go for the state x^i. For the same reason as for the time $k+1$, the cost-to-go $\mathcal{J}_k(x^i)$ will be infinite since $J_{k+1}(x^i)$ was calculated before to be infinite. When these effects continue and the algorithm proceeds backwards in time, the calculated infeasible region will grow into the actual feasible region.

A first step to tackle this problem is to use a big, but finite value for the cost instead of infinity $\phi_k(x^i \notin \mathcal{X}_k) = \mathcal{J}_\infty$ for $k = 1, \ldots, N$. This big, finite value $\mathcal{J}_\infty$ must be bigger than the maximum value of the cost-to-go function $\mathcal{J}_k(x^i)$. Using a finite cost value for infeasible domains improves the solution, but the effect shown above for infinity cannot be completely eliminated close to the boundary line. Throughout this paper, the method of using a finite cost value $\mathcal{J}_\infty$ for infeasible domains together with the algorithm in Section 2.1 is referred to as *basic DP*.

(1) The terms *cost-to-go* and *optimal cost-to-go* are used equivalently throughout this paper referring to *optimal cost-to-go*. It is important to note that the term optimal is used in the sense of optimality achievable despite the numeric errors.

(2) The following notation is used: x_k^i denotes the state variable x in the discretized state-time space at the node with time-index k and state-index i, while x_k denotes a (state-) continuous state variable at time k.

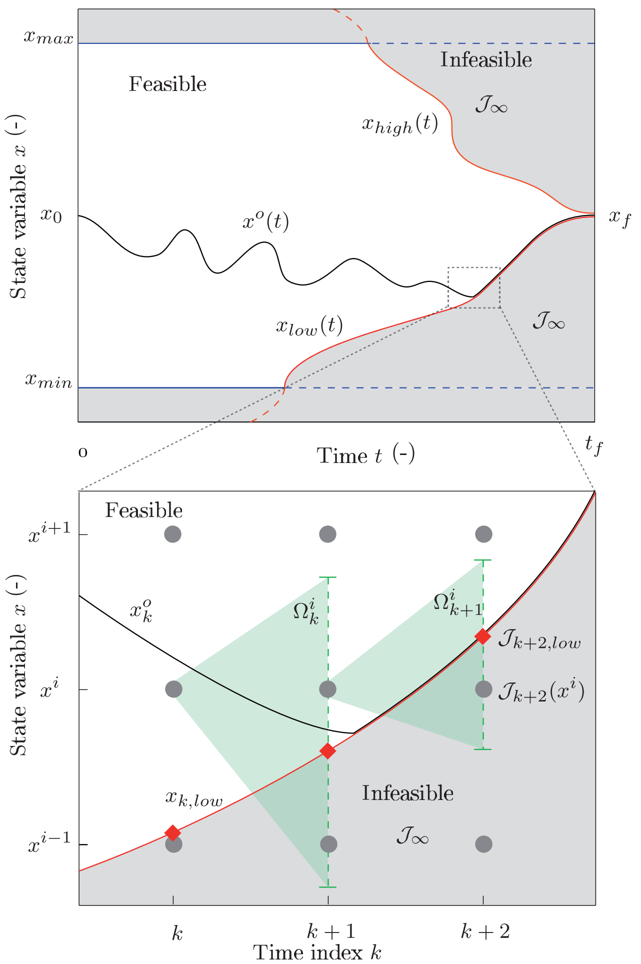

Figure 1

Schematic overview of an optimal control problem solved using the dynamic programming algorithm. The figure shows the state variable boundaries for the dynamic programming algorithm for the entire problem domain (top) and in the grid point/time step domain (bottom).

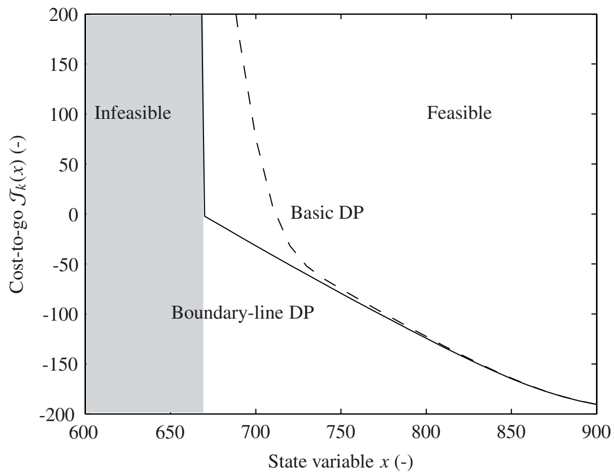

Figure 2

Section of the cost-to-go function $\mathcal{J}_k(x)$ at time index k such that $t = 180$ h for the fishing problem. State-space discretization is $\Delta x = 10$, penalty for infeasible states is set to $\mathcal{J}_\infty = 1200$.

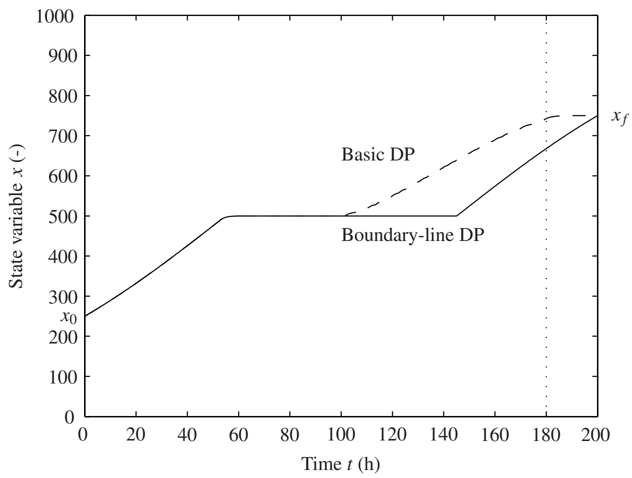

Figure 3

State trajectories from DP for the fishing problem. The solid line shows the result based on the boundary-line method. The dashed line is the state trajectory resulting from the basic DP. The dotted vertical line at $t = 180$ h indicates the time where Figure 2 is evaluated. State-space discretization is $\Delta x = 10$, penalty for infeasible states is set to $\mathcal{J}_\infty = 1200$.

Due to the interpolation between feasible and infeasible states, the infinite gradient at the boundary line is being blurred. This is shown in Figure 2 for the fishing problem (introduced later), where the dashed line is the cost-to-go computed by DP with a finite cost for infeasible states, *i.e.*, the basic DP method. The solid line corresponds to the cost-to-go from DP improved by the new method introduced in this paper. As a result of the blurred cost-to-go function, the optimal state trajectory cannot approach the boundary line since the computed cost-to-go near the boundary line is too high. Figure 3 shows the corresponding state trajectory (dashed) being deviated by this effect. The solid line is the state trajectory from DP improved by the new method.

3 BOUNDARY-LINE METHOD

The method presented in this paper tackles the problem of a blurred gradient at the boundary line due to interpolation of the cost-to-go between a feasible and an infeasible state-grid point. Therefore, the boundary line between feasible and infeasible regions must be found. This is shown in the first part of this section. The second part shows a simple, yet powerful method to improve the DP by accounting for this boundary line. This improved DP is referred to as *boundary-line DP*.

Throughout this section, Equation (8) is reformulated as

$$x_{k+1} = f_k(x_k, u_k) + x_k, \qquad k = 0, 1, \ldots, N-1 \tag{14}$$

where

$$f_k(x_k, u_k) = F_k(x_k, u_k) - x_k \tag{15}$$

3.1 Computation of the Boundary Line

There exist infeasible regions in the state-time space of an optimization problem with fixed final time and a partially constrained final state if the state dynamics are bounded. Since the dynamic system is assumed to be one-dimensional, there exist only two infeasible regions, namely an upper and a lower region. This is depicted in Figure 1. In this section, the lower boundary line between the feasible and the infeasible region is derived. The upper boundary line is found analogously.

The partially constrained final state is given by Equation (4). The lower boundary line is defined as the lowest state $x_{k,low}$ at each time instance k that allows achieving the minimal final state $x_{f,min}$. Note that the lower boundary line is only discretized in time, *i.e.*, it is continuous in the state variable. The lower boundary line can be evaluated by sequentially going backward in time from $k = N-1$ to $k = 0$ and solving the following optimization problem at each time instance k

$$\min_{x_{k,low},u_k} x_{k,low} \tag{16}$$

$$\text{s.t.}$$

$$f_k(x_{k,low}, u_k) + x_{k,low} = x_{k+1,low} \tag{17}$$

$$u_k \in \mathcal{U}_k \tag{18}$$

$$x_{k,low} \in \mathcal{X}_k \tag{19}$$

The problem is initialized with $x_{N,low} = x_{f,min}$. At each time-step, u_k and $x_{k,low}$ are the only unknowns, while $x_{k+1,low}$ is a parameter at time k. By solving Equation (17) for $x_{k,low}$ and inserting it in Equation (16) the following, more direct problem is obtained

$$\max_{x_{k,low},u_k} f_k(x_{k,low}, u_k) \tag{20}$$

$$\text{s.t.}$$

$$f_k(x_{k,low}, u_k) + x_{k,low} = x_{k+1,low} \tag{21}$$

$$u_k \in \mathcal{U}_k \tag{22}$$

$$x_{k,low} \in \mathcal{X}_k \tag{23}$$

If the state is assumed to be unconstrained, *i.e.*, (23) is omitted, the following formulation is equivalent

$$x_{k,low} = x_{k+1,low} - \max_{u_k \in \mathcal{U}_k} f_k(x_{k,low}, u_k) \tag{24}$$

Equation (24) is a so-called fixed point problem ($x = f(x)$), where $x_{k,low}$ is the unknown.

The lower boundary line is finally found by the following algorithm:

1. Initialize with the lower bound of the partially constrained final state $x_{k,low} = x_{f,min}$.
2. Proceed backward in time for $k = N-1, \ldots, 0$:
   (a) solve the fixed point problem (24) without state constraints as shown below in (25-27);
   (b) check wether the solution found respects the state constraints;
   (c) if the constraints are not respected, solve the general problem (20-23);
   (d) store the solution $x_{k,low}$ with the respective minimizer $u_{k,low}$ and the cost-to-go $\mathcal{J}_{k,low}$.

The fixed point problem (24) of time step k without state constraints can be solved with the following algorithm [(3)]:

1. Initialization:

$$x_{k,low}^{j=0} = x_{k+1,low} \tag{25}$$

2. Iteration over j until a specified tolerance is achieved:

$$x_{k,low}^{j+1} = x_{k+1,low} - \max_{u_k \in \mathcal{U}_k} \{f_k(x_{k,low}^j, u_k)\} \tag{26}$$

This algorithm converges if

$$\left| \frac{\partial}{\partial x_{k,low}^j} \max_{u_k \in \mathcal{U}_k} \{f_k(x_{k,low}^j, u_k)\} \right| < 1 \tag{27}$$

Note that the algorithm mentioned above (25-27) finds the limit value $x_{k,low}$ in the first iteration if the update function f_k is independent of the state variable x_k.

3.2 Interpolation Near the Boundary Line

It is assumed that the state boundary lines $x_{k,low}$ (and $x_{k,high}$) shown in Figure 1 with their corresponding cost-to-go $\mathcal{J}_{k,low}$ (and $\mathcal{J}_{k,high}$) along the boundary line have been calculated prior to the DP algorithm. Therefore, when the set Ω_k^i contains the boundary it is possible to interpolate between the exact boundary and a feasible state grid point, as illustrated in Figure 4 with the solid and the dashed lines. The dotted line illustrates the interpolation by the basic algorithm at the boundary between feasible and infeasible regions.

Consider the DP algorithm to evaluate the cost-to-go for the state-grid point x^i at time $k+1$ (see *Fig. 1*, bottom). Starting from state x^i, the state achieved at the end of this time-step

$$x_{k+2} = f_{k+1}(x^i, u_{k+1}) + x^i \qquad \in \Omega_{k+1}^i \tag{28}$$

can reach the feasible as well as the infeasible region. The corresponding cost-to-go $\mathcal{J}_{k+2}(x_{k+2})$ is evaluated by interpolation between $\mathcal{J}_{k+2}(x^{i+1})$ and $\mathcal{J}_{k+2,low}$ if the state x_{k+2} is

(3) The top right index of x is the iteration index, here. It is not the index of the state-grid as used in the rest of the paper.

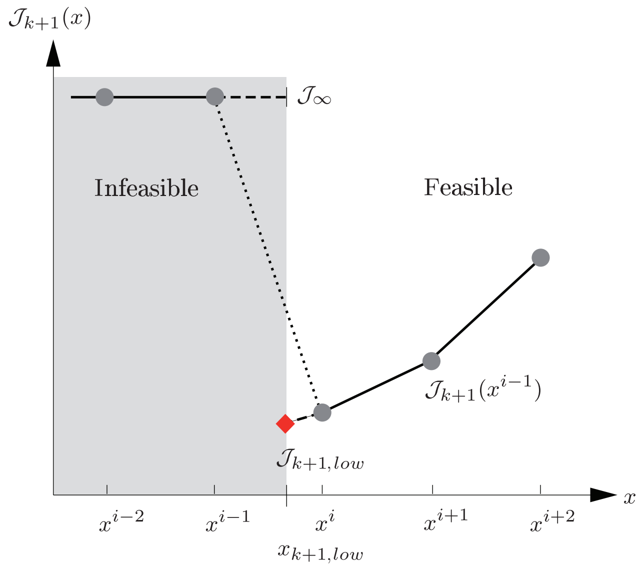

Figure 4

Interpolation of $\mathcal{J}_{k+1}(x)$ near the boundary line. The dashed lines illustrates the (linearly) interpolated values including the boundary line. The dotted line illustrates the interpolation used by the basic algorithm.

above or on the boundary line $x_{k+2,low}$. Otherwise, the cost-to-go is set to infinity or to the big, finite value $\mathcal{J}_\infty$. This procedure allows maintaining the same accuracy close to the boundary line as achieved within the feasible domain.

The application of the optimal control signal map in the forward simulation which is mentioned in Section 2.1 is improved by the boundary line analogously. Since the control signal on the boundary line was evaluated before, interpolation of the control signal is carried out between the grid points of the feasible domain or between the feasible domain and the boundary line.

4 EXAMPLE 1: SIMPLE DYNAMIC MODEL

This section studies a well-known optimal control problem, namely the optimal fishing in a Lotka-Volterra fish population. The fishing problem is chosen because it has an analytic solution.

4.1 Continuous-Time Problem

The continuous-time dynamic Lotka-Volterra system is

$$\dot{x}(t) = \frac{2}{100} \cdot \left(x(t) - \frac{x^2(t)}{1000} \right) - u(t) \tag{29}$$

where the state variable $x(t)$ is the amount of fish in a lake, the control signal $u(t)$ is the fishing rate. The control signal $u(t)$ is limited to $u(t) \in [0,\ 10]$. For the considered system the state $x(t)$ is limited to $x(t) \in [0,\ 1000]$ since

$$\lim_{\substack{t\to\infty \\ u(t)=0}} x(t) = 1000 \tag{30}$$

The objective is to maximize the amount of fish caught, which is equivalent to minimizing

$$J = \int_0^{t_f} -u(t)dt \tag{31}$$

within a fixed time t_f while the minimal amount of fish in the population at the final time must be $x_{f,min} = 750$. This can be stated as the optimal control problem

$$\min_{u(t)} \int_{t=0}^{t_f} -u(t)dt \tag{32}$$

s.t.

$$\dot{x}(t) = \frac{2}{100} \cdot \left(x(t) - \frac{x^2(t)}{1000} \right) - u(t) \tag{33}$$

$$x(0) = 250 \tag{34}$$

$$x(t_f) \geq 750 \tag{35}$$

$$x(t) \in [0,\ 1000] \tag{36}$$

$$u(t) \in [0,\ 10] \tag{37}$$

$$t_f = 200 \tag{38}$$

The solution to this optimal control problem is straightforward to determine and consists of three parts: First, there is no fishing to let the population grow, then there is fishing such that the population is kept constant, then there is no fishing again to let the population grow to the final condition. The optimal control expressed in time is

$$u^o(t) = \begin{cases} 0 \text{ if } & t \in [0,\ t_a] \\ 5 \text{ if } & t \in (t_a,\ t_f - t_b) \\ 0 \text{ if } & t \in [t_f - t_b,\ t_f] \end{cases} \tag{39}$$

where

$$t_a = t_b = 100 \cdot \text{artanh}\left(\frac{1}{2}\right) \tag{40}$$

The final maximum amount of fish caught is

$$\begin{aligned} J^o_{analytic} &= -5 \cdot (t_f - t_a - t_b) \\ &= -1000 \cdot \left(1 - \text{artanh}\left(\frac{1}{2}\right)\right) \\ &\approx -450.694 \end{aligned} \tag{41}$$

4.2 Discrete-Time Problem

In order to evaluate the optimal solution by means of dynamic programming, the continuous-time state dynamics (29) must be discretized. Using an Euler forward approximation with a time step $t_s = 0.2$ h, the discrete-time model is

$$x_{k+1} = f(x_k, u_k) + x_k, \qquad k = 0, 1, \ldots, N-1 \tag{42}$$

where

$$f(x_k, u_k) = t_s \cdot \left(\frac{2}{100} \cdot \left(x_k - \frac{x_k^2}{1000} \right) - u_k \right) \quad (43)$$

The state x_k is the amount of fish in a lake, while the control signal u_k is the constant fishing rate during one time step. The discrete-time optimal control problem is

$$\min_{u_k \in [0,\ 10]} \sum_{k=0}^{N-1} -u_k \cdot t_s \quad (44)$$

s.t.

$$x_{k+1} = f(x_k, u_k) + x_k \quad (45)$$

$$x_0 = 250 \quad (46)$$

$$x_N \geq 750 \quad (= x_{f,min}) \quad (47)$$

$$x_k \in [0,\ 1000] \quad (48)$$

$$N = \frac{200}{t_s} + 1 \quad (49)$$

As mentioned in Section 2.2, use of a big, but finite value $\mathcal{J}_\infty$ to penalize infeasible states improves the numerics. This value should be chosen as small as possible, but larger than any value of the (feasible) cost-to-go that could occur. Since this simple example allows for analytic solutions, the maximum cost-to-go of the continuous-time problem is evaluated in order to choose a suitable value for $\mathcal{J}_\infty$. The minimum of the cost-to-go $\mathcal{J}_t(x)$ is obviously at $t = 0$ and $x = 1000$ and yields

$$\mathcal{J}_{t=0}(x = 1000) = 500 \operatorname{artanh}\left(\frac{1}{2}\right) - 1000 - 125\pi \approx -1118 \quad (50)$$

For the fishing problem, at $t = 0$, the minimum cost-to-go (50) is approximately 1118 less than the cost-to-go at $t = t_f = 200$

$$\mathcal{J}_{t=200}(x \in \mathcal{X}_{t=200}) = 0 \quad (51)$$

The infeasible states at t_f must therefore be penalized by a value larger than 1118 to ensure that the final state constraint is met.

Consequently, the penalty $\mathcal{J}_\infty$ used for the cost-to-go of infeasible states is set to a value greater than $\mathcal{J}_\infty > 1118$. For the example here a value of

$$\mathcal{J}_\infty = 1200 \quad (52)$$

is chosen and is used in the DP algorithm.

The output of the dynamic programming algorithm is an optimal control signal map, specifying the optimal control signal at each time step k and each state $x_k \in \mathcal{X}_k$. The optimal control signal map for the Lotka-Volterra system is shown in Figure 5. It shows that in the beginning of the problem the optimal control is "not fishing" ($u = 0$) if the fish population is small ($x < 500$), "moderate fishing" ($u = 5$) if the population is $x = 500$ and "full fishing" ($u = 10$) if the population is large ($x > 500$). Toward the end of the problem, one must stop fishing as late as possible, such that the population reaches the specified minimum final size of $x_{f,min} = 750$. The resulting optimal state trajectory, *i.e.*, the fish population for an initial state of $x_0 = 250$ is shown as the black solid line. The solution of the dynamic programming clearly reflects the optimal control found for the continuous problem (39).

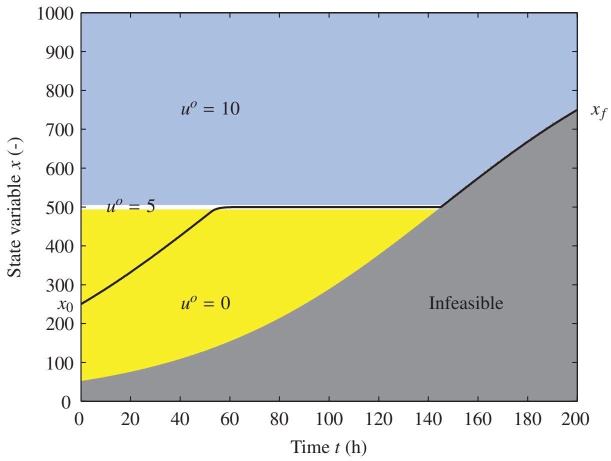

Figure 5

The optimal control signal map, determined using dynamic programming, for the discrete-time Lotka-Volterra system. The optimal state trajectory for $x_0 = 250$ when using the map is shown as the solid black line.

4.3 Resolution Study

As mentioned earlier, the accuracy of the solution obtained with dynamic programming can degrade due to numeric issues. The state space must be discretized for the DP algorithm. The resolution of the state-space discretization is a critical quantity. On one hand, the computational effort increases with a higher resolution. On the other hand, the accuracy of the solution improves with increasing resolution.

Therefore, a study is carried out here to quantify the accuracy of the solution obtained by DP for the simple example of the fishing problem. The fishing problem has been chosen because an analytic solution exists that can be used as a benchmark. The resolution study is carried out for the basic DP, but also for the new method presented in this paper, *i.e.*, the boundary-line DP.

The quality of the solution is expressed as the relative difference between optimal cost obtained by DP and the

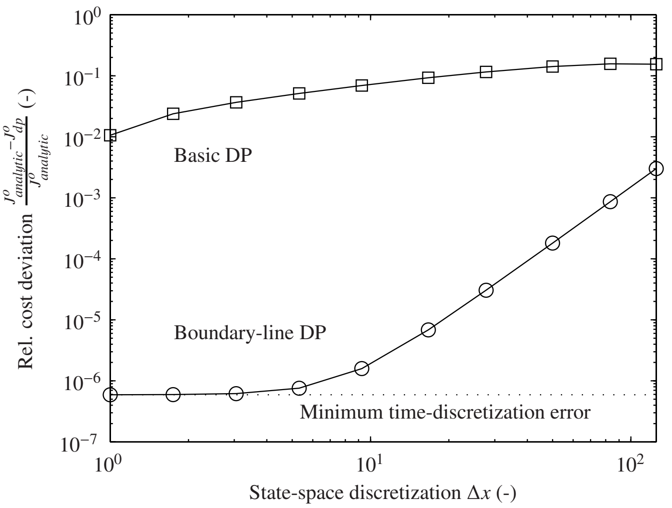

Figure 6

The relative deviation of the cost computed by dynamic programming compared to the optimal analytic solution for the fishing problem.

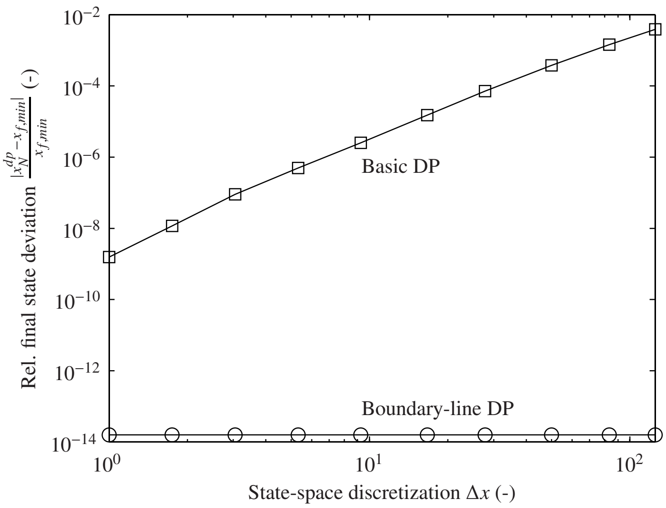

Figure 7

The relative deviation of the actual final state and the optimal final state for the fishing problem.

analytic optimal solution, $\frac{J^o_{analytic}-J^o_{dp}}{J^o_{analytic}}$. Figure 6 shows this deviation of the optimal solution evaluated with DP (basic and boundary-line) from the analytic optimal solution.

Since the analytic solution is evaluated for the original continuous-time problem, the discrete-time solution can never achieve the analytic optimal solution. This discretization error is indicated in Figure 6 with the dotted line marked as "minimum time-discretization error". It emphasizes that the solution using the boundary-line DP converges well toward the discrete-time optimum. Furthermore, this figure illustrates the importance of the boundary line: the numeric solution with the boundary-line DP is closer to the analytic solution by a factor of 50 to 68 000 than with the basic DP for the same resolution. It is interesting to note that the cost (44) resulting from boundary-line DP is inferior to the cost resulting from basic DP over the entire range of resolution that was investigated. The solution using the boundary-line DP at the lowest resolution ($\Delta x = 125$) is closer to the analytic solution than the solution of the basic DP at the highest resolution ($\Delta x = 1$).

The relative deviation of the final state achieved by the DP (basic and boundary-line) from the optimal final state is shown in Figure 7 for different resolutions. The optimal final state is the lowest admissible final state for this example, *i.e.*, $x^o(t_f) = x_{f,min}$. The figure shows clearly that the final state deviation of the basic DP decreases with decreasing Δx, *i.e.*, with increasing resolution. Using the boundary-line DP, the final state deviation is negligible over the entire range of resolutions investigated here.

4.4 Computational Effort

The computational effort of an optimization method is often a crucial factor that determines whether a method is being applied in practice for a given problem or not. Therefore, not only the accuracy of a solution as shown in Section 4.3 is relevant, but also the corresponding computational cost.

The number of model-function evaluations for the basic DP with an equally spaced grid is given by

$$N^{DPbasic}_{feval} = N_x \cdot N_u \cdot N \tag{53}$$

This is only true for a single-dimensional state space and a scalar control signal. The variable N_x represents the number of grid points for the state space, N_u for the control signal, and N for the time discretization.

When using the boundary-line DP, the infeasible domain is well known. Consequently, the computation for the grid points in this infeasible domain (see *Fig. 1*) can be omitted [2]. The number of infeasible grid points at a time step k is denoted as $N^{infeas}_{k,x}$. Hence, the number of function evaluations that can be saved are

$$N^{infeas}_{feval} = N_u \cdot \sum_{k=0}^{N-1} N^{infeas}_{k,x} \tag{54}$$

The cost for evaluating the boundary line cannot be neglected. The number of function evaluations needed to compute the line is denoted by N^{line}_{feval}.

Consequently, the number of function evaluations required for solving the DP with the boundary-line method is given by

$$N^{DPline}_{feval} = N^{DPbasic}_{feval} - N^{infeas}_{feval} + N^{line}_{feval} \tag{55}$$

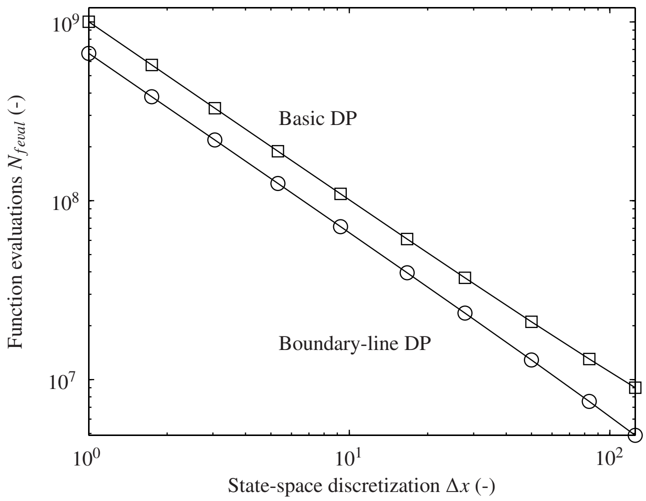

Figure 8

Number of function evaluations needed to find the solution of the fishing problem.

Figure 8 shows the number of function evaluations for the fishing problem over the discretization step Δx. It shows that more computations can be saved due to the infeasible domain than are required to evaluate the boundary line. The boundary-line DP requires fewer function evaluations by a factor of 1.5 to 1.85 than the basic DP over the entire range of discretization steps investigated here. Furthermore, it should be recalled that the accuracy of the solution is considerably higher, even though the computational burden is lower.

More interesting is a comparison of solutions of similar accuracy. Therefore, the solution using the basic DP at the lowest discretization step of $\Delta x = 1$ is compared to the solution of the boundary-line DP at its highest discretization step of $\Delta x = 125$. The values obtained are shown in Table 1. These results reveal that the boundary-line DP is computationally more efficient by a factor of $\frac{1\,002\,001\,000}{4\,877\,143} \approx 205$ than the basic DP, even though the accuracy of the solution is still better (449.340 > 445.936). This result motivates to apply the method to more complex systems.

TABLE 1

Comparison at similar accuracy for the fishing problem

		Basic DP	Boundary-line DP
State-space discretization (-)	Δx	1	125
Amount of fish caught (-)	$-J$	445.936	449.340
Function evaluations (-)	N_{feval}	1 002 001 000	4 877 143

5 EXAMPLE 2: HYBRID ELECTRIC VEHICLE

This section studies the control problem of deciding the power split between the two power converters in a full parallel electric hybrid vehicle throughout a given drive cycle. A hybrid vehicle includes state-of-charge constraints and normally also a final state-of-charge constraint to ensure a charge-sustaining solution. As a result, the dynamic programming solution is sensitive to the method used to account for final state constraints. The basic DP is compared to the proposed boundary-line DP for the optimal control problem of the hybrid vehicle. The focus of this section is not on modeling, but rather on handling the state constraints appropriately when implementing dynamic programming. Readers interested in detailed equations of the model are referred to [9].

5.1 Discrete-Time Problem

The full parallel hybrid electric vehicle model is a quasi-static, discrete-time model. The modeling follows the theories described in [1, 10]. Essentially, the model contains the battery state-of-charge as the only state variable which is sufficient for energy management considerations. To summarize the model, the combustion engine is modeled using an affine Willans approximation, the electric motor is modeled using an electric-power map (derived from detailed simulations), and the battery is modeled as a voltage source together with a resistance in series. The vehicle model includes air drag, rolling friction, and inertial forces. The gearbox is modeled using a constant efficiency of 95%. The hybrid vehicle considered in this study has a 20% hybridization as defined in [9]. The model equations can be summarized and described as

$$x_{k+1} = f(x_k, u_k, v_k, a_k, i_k) + x_k \tag{56}$$

where x_k is the battery state-of-charge, u_k is the torque split factor, v_k is the vehicle speed, a_k is the vehicle acceleration, and i_k is the gear number. The model assumes isothermal conditions, no extra fuel consumption during the startup of the combustion engine, and no energy losses during gear shifting. A constant auxiliary electric power demand of 350 W is used in the model.

Since the drive cycle is assumed to be known in advance, the particular driving speed v_k, acceleration a_k, and gear number i_k at instance k can be included in the model function to form the time-variant model

$$x_{k+1} = f_k(x_k, u_k) + x_k, \qquad k = 0, 1, \ldots, N-1 \tag{57}$$

The optimization problem of minimizing the total fuel mass consumed

$$J = \sum_{k=0}^{N-1} \Delta m_f(u_k, k) \tag{58}$$

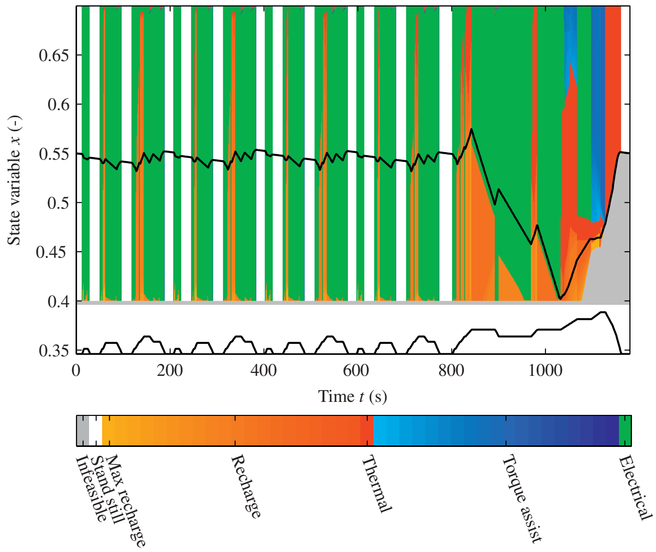

Figure 9

Output of the dynamic programming algorithm. The optimal input map for a full HEV driving the NEDC and the state-of-charge trajectory (black) when using the optimal control signal map.

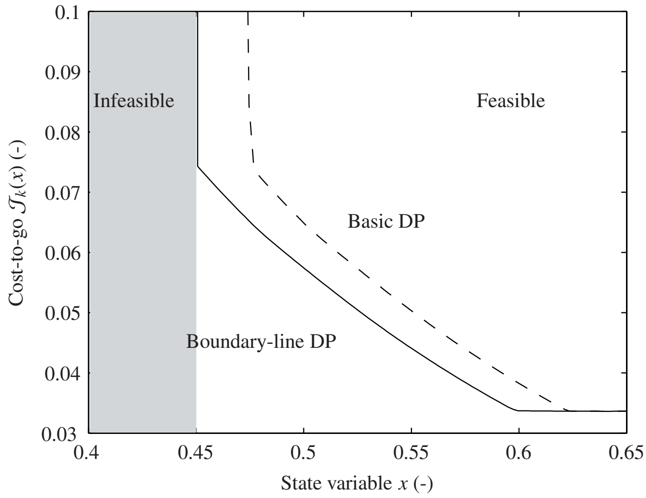

Figure 10

Section of the cost-to-go function $\mathcal{J}_k(x)$ at time index k such that $t = 1100$ s for the HEV-problem. State-space discretization is $\Delta x = 2.15 \times 10^{-3}$, penalty for infeasible states is set to $\mathcal{J}_\infty = 10^4$.

for the hybrid vehicle over a given drive cycle, here the New European Driving Cycle (NEDC), can be stated as the discrete-time optimal control problem

$$\min_{u_k \in \mathcal{U}_k} \sum_{k=0}^{N-1} \Delta m_f(u_k, k) \tag{59}$$

s.t.

$$x_{k+1} = f_k(x_k, u_k) + x_k \tag{60}$$

$$x_0 = 0.55 \tag{61}$$

$$x_N \geq 0.55 \quad (= x_{f,min}) \tag{62}$$

$$x_k \in [0.4,\ 0.7] \tag{63}$$

$$N = \frac{1180}{t_s} + 1 \tag{64}$$

where Δm_f is the fuel mass consumption at each time step. The time step in this example is $t_s = 1$ s. Throughout this section the optimal control problem (59-64) is studied and solved using dynamic programming for the NEDC. In Figure 9 the optimal torque split is shown at each state-of-charge over the duration of the drive cycle for the hybrid vehicle. Similarly to the fishing problem, the optimal state-of-charge trajectory is close to the boundary of the feasible state region at the end of the problem.

As mentioned in Section 2.2 for the fishing problem, the cost-to-go function is blurred when a finite value for $\mathcal{J}_\infty$ is used together with an interpolation. For the hybrid vehicle problem, the blurring effect is similar to that of the fishing problem when a finite cost for infeasible states is used. Figure 10 shows the cost-to-go at $t = 1100$ s when the basic DP and the boundary-line DP are used. Since the blurring effect is dependent on the state-space resolution, the next section investigates this dependency for the basic DP and on the boundary-line DP.

5.2 Resolution Study

Dynamic programming is often used to calculate the optimal fuel consumption for hybrid vehicles. The optimal fuel consumption is then used, for example, to compare different causal energy management strategies or component dimensions. However, such comparisons rely on the accuracy of the dynamic programming solution. In general, a higher resolution of the state space improves the accuracy of the solution. This section studies different state-space resolutions for the hybrid vehicle problem using the basic DP and the boundary-line DP.

Figure 11 shows the fuel consumption calculated with the basic DP and the boundary-line DP. The fuel consumption is proportional to the total cost J, which is the fuel mass consumption. In contrast to the simple model in Section 4, no analytic solution to the hybrid vehicle problem is known. Consequently, the absolute value of the fuel consumption is shown instead of the relative deviation from the optimum. Figure 11 shows that the fuel consumption is considerably lower when using the boundary-line DP compared to the basic DP, especially for large state-space discretizations Δx. Furthermore, the figure shows that the fuel consumption resulting from the boundary-line DP, compared to the results

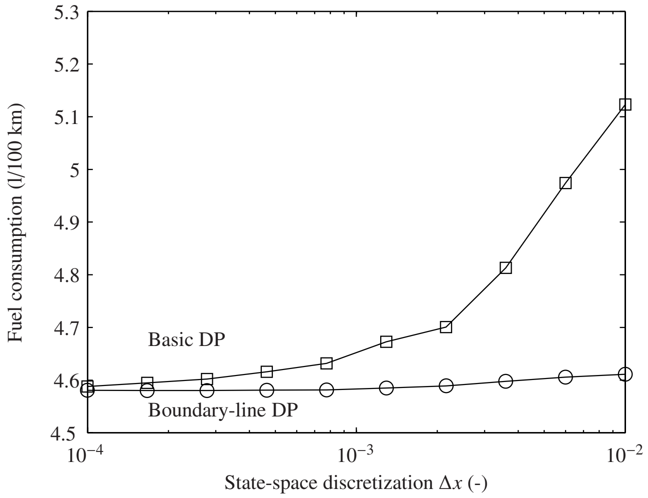

Figure 11

Fuel consumption of the HEV obtained using DP for different state-space discretizations.

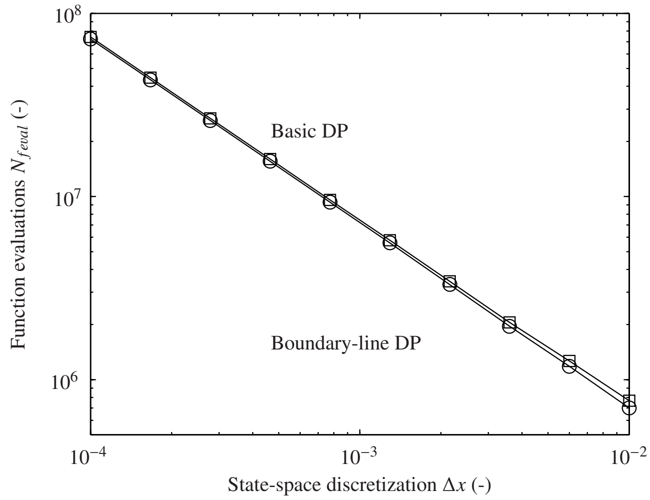

Figure 13

Number of function evaluations needed for the HEV problem.

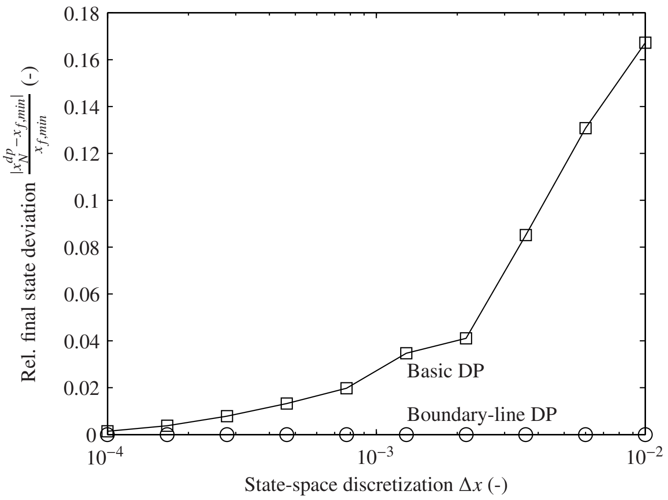

Figure 12

The deviation of the actual final state and the optimal final state for the HEV problem.

of the basic DP, is insensitive to the state-space discretization.

Figure 12 shows the relative deviation of the final state achieved by the DP algorithm from the optimal final state $x^o(t_f) = x_{f,min}$. The behavior of the fuel consumption shown in Figure 12 resembles the behavior of the relative deviation of the final state in Figure 11. This is a consequence of the fact that any final state deviation is accumulated electric energy that could have been used to save fuel during the drive cycle. For low resolutions of the state space, the blurring effect of the cost-to-go substantially affects the final state-of-charge when using the basic DP. For a state-space discretization of $\Delta x = 0.01$ the error is greater than 16%. However, when increasing the state-space resolution, the final state approaches the lowest admissible final state $x_{f,min}$, and the fuel consumption decreases.

In general, for the hybrid vehicle problem, the errors in the final state-of-charge and in the total cost originate from two main problems. First, the blurring effect unrealistically increases the cost-to-go and therefore increases the final state-of-charge. Second, the state-space discretization must include enough points to capture the nonlinearities in the cost-to-go, which both methods suffer from. However, for the hybrid vehicle problem, the error associated with the blurring effect is by far the largest error source. Using the boundary-line method eliminates this blurring effect and is therefore an appropriate method for increasing the accuracy of the final solution.

5.3 Computational Effort

The accuracy of the final solution described in the previous section must be put in relation to the computational effort of the particular discretization. This section shows the relationship between the accuracy of the solution and the computational effort, expressed as model function evaluations, similarly to those described in Section 4.4.

Figure 13 shows the number of function evaluations for the hybrid vehicle problem over the discretization step Δx. The difference between the boundary-line DP and the basic DP is due to the removal of infeasible states in the state space for the boundary-line DP. However, this effect is not as large as in the fishing problem since it depends on the problem length and on the boundary line.

TABLE 2

Comparison at similar accuracy for the HEV problem

		Basic DP	Boundary-line DP
State-space discretization (-)	Δx	10^{-4}	1.3×10^{-3}
Fuel mass consumed (kg)	J	0.3790	0.3788
Fuel consumption (l/100 km)		4.5879	4.5852
Function evaluations (-)	N_{feval}	74 364 780	5 578 935

When comparing the boundary-line DP and the basic DP at a similar accuracy the difference in computational efforts becomes clear, as shown in Table 2. For the basic DP using $\Delta x = 10^{-4}$ the total cost of the HEV is 0.3790 kg fuel mass consumed. Furthermore, the boundary-line method achieves a similar total cost of 0.3788 kg using $\Delta x = 1.3 \times 10^{-3}$. Consequently, the number of model function evaluations used for the boundary-line DP is computationally $\frac{74\,364\,780}{5\,578\,935} \approx$ 13 times more efficient than using the basic DP.

Similarly to the fishing problem, the biggest reduction in computational effort is due to the possibility of reducing the state-space discretization while maintaining the accuracy of the solution.

CONCLUSIONS

The boundary-line DP presented in this paper improves the efficiency of dynamic programming considerably for the case of a single state variable. Not only the final state constraint is fulfilled at high accuracy, but also the calculated optimal cost is very close to the true solution even for a moderate resolution of the discretized state space. The computational cost is substantially reduced since the new method allows the accuracy to be maintained at a much lower state-space resolution. Furthermore, the number of calculated grid points is reduced by evaluating the feasible domain only. The method presented here improves the dynamic programming only if the optimal state trajectory is close to the bounds of the feasible region at some points. This is typically the case for constrained optimal control problems.

The new method is well suited for optimal control of hybrid electric vehicles because the optimal trajectory coincides with the boundary line at the end of most drive cycles and because the system can be sufficiently well described with a single state variable. The resulting gain in computational time can be used for extensive parametric studies, for instance, such as optimal component dimensioning without increasing the total computational effort.

In this study, the proposed method is applied to partially constrained optimal control problems where only the lower state boundary line is determined. However, the proposed method can easily be applied to constrained optimal control problems where a lower and an upper state boundary line have to be determined.

Future work includes investigations of a possible extension of the proposed method to include multi-dimensional discrete optimal control problems.

REFERENCES

1 Guzzella L., Sciarretta A. (2007) *Vehicle propulsion systems: Introduction to modeling and optimization*, Springer, Berlin, 2nd ed.

2 Back M., Terwen S., Krebs V. (2004) Predictive powertrain control for hybrid electrical vehicles, in *IFAC Symposium on Advances in Automotive Control*, Salerno, Italy, April 2004, pp. 451-457.

3 Pu J., Yin C. (2007) Optimal control of fuel economy in parallel hybrid electric vehicles, *J. Automobile Eng.* **221**, 1097-1106.

4 Bellman R.E. (1957) *Dynamic programming*, Princeton University Press, Princeton, NJ.

5 Bertsekas D.P. (2005) *Dynamic programming and optimal control*, Athena Scientific, Belmont, MA, 3rd ed.

6 Luus R., Rosen O. (1991) Application of dynamic programming to final state constrained optimal control problems, *Ind. Eng. Chem. Res.* **30**, 7, 1525-1530.

7 Luus R. (2000) *Iterative dynamic programming*, Vol. 110 of *Monographs and surveys in pure and applied mathematics*, Chapman & Hall/CRC, Boca Raton, FL.

8 Schaefer M.B. (1954) Some aspects of the dynamics of populations important to the management of the commercial marine fisheries, *B. Math. Biol.* **53**, 253-279. Reprinted from the *B. Inter-Am. Trop. Tuna Commission* **1**, 2, 27-56.

9 Sundström O., Guzzella L., Soltic P. (2008) Optimal hybridization in two parallel hybrid electric vehicles using dynamic programming, in *17th IFAC World Congress*, Seoul, Korea, July 2008, pp. 4642-4647.

10 Guzzella L., Onder C.H. (2004) *Modeling and control of internal combustion engine systems*, Springer, Berlin.

Final manuscript received in March 2009
Published online in September 2009