Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnLine 50-1 (2017) 5953–5960

A Computationally Fast Iterative Dynamic Programming Method for Optimal Control of Loosely Coupled Dynamical Systems with Different Time Scales ⋆

Jonathan Lock * **Tomas McKelvey** **

* *Dept. of Signals and Systems, Chalmers University of Technology, SE-412 96 Gothenburg, Sweden, lock@chalmers.se*
** *Dept. of Signals and Systems, Chalmers University of Technology, SE-412 96 Gothenburg, Sweden, mckelvey@chalmers.se*

Abstract: Iterative dynamic programming is a powerful method that is often used to solve finite-dimensional nonlinear constrained global optimal control problems. However, multi-dimensional problems are often computationally complex, and in some cases an infeasible result is generated despite the existence of a feasible solution. A new iterative multi-pass method is presented that reduces the execution time of multi-dimensional, loosely-coupled, dynamic programming problems, where some state variables exhibit dynamic behavior with time scales significantly smaller than the others. One potential application is the optimal control of a hybrid electrical vehicle, where the computational burden can be reduced by a factor on the order of 100 – 10000. Furthermore, new regularization terms are introduced that typically improve the likelihood of generating a feasible optimal trajectory. Though the regularization terms may generate suboptimal solutions in the interim, with successive iterations the generated solution typically asymptotically approaches the true optimal solution.
Note: Full source code is freely available online with an implementation of the solver, some usage examples, and the test cases used to generate the results shown in this paper.

Keywords: Dynamic programming, Optimal control, Global optimization, Nonlinear control, Bang-bang control, Efficiency enhancement

1. INTRODUCTION

Non-causal global nonlinear constrained optimal control is a notoriously difficult problem which, in general, does not have a known analytical solution. Hence, it is often necessary to use numerical methods. One method that is often used for finite-dimensional problems is dynamic programming (DP). For example, DP is often used for designing hybrid vehicle controllers, where DP is typically used to benchmark the quality of simpler, suboptimal, causal controllers (Liu and Peng (2008); Pérez et al. (2006); Sciarretta and Guzzella (2007)). DP is guaranteed to generate the global optimum for problems that can be represented in a graph. However, DP is computationally complex for multidimensional problems, where the required number of computations scales exponentially with the number of dimensions.

This paper presents a new DP method (and an implementation of it in Matlab) for multidimensional problems that can be described as a loosely coupled set of ordinary differential/difference equations with different time scales. For problems of this type the presented method significantly reduces the time required to generate a solution, and furthermore increases the likelihood of generating a feasible solution. One example of an application that this method works well for is that of hybrid vehicle control, where performance gains on the order of the quotient of the system's time scales are realizable. Typically, this gives a performance improvement on the order of $10^2 - 10^4$.

In this paper DP is used to solve a discrete-valued, discrete-time approximation of a continuous-value problem in discrete- or continuous-time. The DP method used in this paper starts with a backward-calculation phase where, for a sample k, each element from a set of system inputs $\mathcal{U}_k$ is exhaustively applied to each element of a set of system states $\mathcal{X}_k$. The *best* control $u_{opt}[k]$ and corresponding cost $c_{opt}[k]$ are stored for every system state, where the *best* control and cost minimizes the total cost from the current sample to the final sample. This process is repeated for all samples starting from the next-to-last sample and working backwards to the first sample. The optimal control and state trajectories are generated in a forward-calculation phase, where for a given initial state the best stored control signal $u_{opt}[k]$ is successively applied to the system state $x[k]$ for all samples. Interpolation is used when the system state $x[k]$ does not exactly match one of the states evaluated during the back-calculation phase. This directly gives the optimal control and state trajectories $u_{opt}[k]$ and $x_{opt}[k]$. A formal definition of DP for optimal control is beyond the

⋆ This work has been performed within the Combustion Engine Research Center at Chalmers (CERC) with financial support from the Swedish Energy Agency.

10.1016/j.ifacol.2017.08.1498

scope of this paper, curious readers are referred to any of Bellman (1956); Bertsekas (2005); Sundström and Guzzella (2009).

1.1 Problem definition

For many engineering applications, typical optimal control problems are continuous-time, continuous-variable ($CTCV$) problems. Here, we consider the case where the control input function $u(t)$ from time t_k to time t_{k+1} is linearly parameterized with an l-dimensional control variable $u[k]$. If the function is the step function this representation is known as zero-order-hold sampling (Åström and Wittenmark, 1997, p. 32). This optimal control problem can then be represented as a discrete-time, continuous-variable ($DTCV$) problem, defined as

$$\begin{aligned} J^*_{0,N_s} &= \min_{\mathcal{U}} L_{0,N_s}(\mathcal{U}) \\ &\text{s.t.} \\ L_{0,N_s} &= \sum_{k=0}^{N_s} c(x[k], u[k], k) \\ x[k+1] &= f(x[k], u[k], k),\ k = [0, N_s - 1] \\ b_{in}(x[k], u[k], k) &\leq 0,\ k = [0, N_s] \\ x[k] &\in \mathbb{R}^m \\ u[k] &\in \mathbb{R}^l, \end{aligned} \tag{1}$$

where $x[k]$ is an m-dimensional vector of real-valued state variables and $u[k]$ is an l-dimensional vector of control inputs. The total cost function L_{0,N_s} is minimized with respect to $u[k]$, given the system dynamics $f(\dots)$ and a set of inequality constraints $b_{in}(\dots)$. Here, k is an index that orders the state and control variable trajectories, and for the cases considered here is directly proportional to time.

DP cannot directly be applied to solve (1). Instead, the problem is further approximated by quantizing the state and control variables which yields a discrete-time, discrete-variable ($DTDV$) form, i.e. $x[k]$ and $u[k]$ must each be members of a set with a finite number of elements, denoted $\mathcal{X}_k$ and $\mathcal{U}_k$ respectively. Each element of $\mathcal{X}_k$ can be viewed as a vertex in a directed graph (shown in Figure 1) corresponding to sample k, where the existence of an edge between an element $x[k] \in \mathcal{X}_k$ and an element $x[k+1] \in \mathcal{X}_{k+1}$ implies that there exists a feasible control $u[k] \in \mathcal{U}_k$ so that the constraints in (1) are fulfilled for $x[k]$, $x[k+1]$, and $u[k]$.

In (1), the system dynamics model is given in implicit form — $x[k+1]$ is generated with the function $f(\dots)$ given a state $x[k]$ and control $u[k]$ for sample k. This particular representation is chosen as it is typically difficult to generate a model in explicit form (i.e. of type $u[k] = g(x[k], x[k+1], k)$) in many applications. As a result of this representation, there is no guarantee in the back-calculation phase that applying a member of $\mathcal{U}_k$ to a member of $\mathcal{X}_k$ will generate a value $x[k+1] \in \mathcal{X}_{k+1}$. Similarly, during the forward-calculation phase, if $x[k] \notin \mathcal{X}_k$ then there does not exist an associated stored optimal control signal $u_{opt}[k]$ to apply. A method that resolves this issue is to define the existence of on-demand pseudo-vertices $\bar{\mathcal{X}}_k$, where any $x[k] \notin \mathcal{X}_k$ is defined to be an element of $\bar{\mathcal{X}}_k$, and whose numerical values are derived based on the

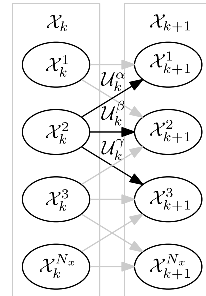

Fig. 1. Directed graph representation of a DTDV problem. For the state configuration $\mathcal{X}_k^2$, only the α'th, β'th and γ'th elements from $\mathcal{U}_k$ are feasible and bring the state to $\mathcal{X}_{k+1}^1$, $\mathcal{X}_{k+1}^2$, and $\mathcal{X}_{k+1}^3$ respectively at the next sample.

nearby elements in $\mathcal{X}_k$ using some suitable interpolation method. Similarly, a pseudo-optimal control $\bar{u}_{opt}[k]$ can be generated based on the nearby stored optimal controls $\mathcal{U}_k$. If the elements in $\mathcal{X}_k$ are carefully chosen this becomes a computationally inexpensive gridded interpolation, e.g. nD linear interpolation (Bellman and Dreyfus (2015); Elbert et al. (2013)).

1.2 Iterative Dynamic Programming

Iterative dynamic programming (IDP), as defined by Luus (1990), can be used to solve real-valued optimization problems, i.e. problems where the state and control variables take values from the set of real numbers. IDP reduces the state and control quantization to an arbitrarily small amount by first searching over a relatively coarse but large set of system inputs and states using DP, and then successively generating a denser and narrower search range centered about the previous result. This successive reduction in search range is then repeated, eventually allowing for an arbitrarily small variable quantization. This method has, for example, been used in the field of hybrid vehicles, primarily as a solver for limited-horizon nonlinear MPC control, see Wahl and Gauterin (2013).

IDP can also handle problems where the optimal control trajectory lies along a boundary of the feasible set — typically with successive iterations the generated trajectory asymptotically approaches the optimal one. This is an important advantage of IDP as compared to DP defined by e.g. Sundström et al. (2010), which will generate trajectories that avoid the edges of the feasible set, potentially resulting in a suboptimal solution.

Recently, Elbert et al. (2013) implemented a *non-iterative* DP method that correctly handles problems that lie along a boundary of infeasibility. However, this method does not have the additional benefit of reducing variable quantization.

1.3 Current issues

IDP is a powerful method for solving many types of global optimization problems. However, previously it has been unsuitable for certain sub-classes of problems due to issues with poor feasibility guarantees and large search spaces. This paper presents a few extensions that can help resolve these issues.