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.

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 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)).

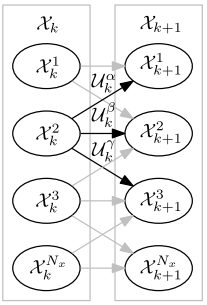

Fig. 1. Directed graph representation of a DTDV problem. For the state configuration $\mathcal{X}_k^2$, only the α'th, β'th and γ'th elements from $\mathcal{U}_k$ are feasible and bring the state to $\mathcal{X}_{k+1}^1$, $\mathcal{X}_{k+1}^2$, and $\mathcal{X}_{k+1}^3$ respectively at the next sample.

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.

Poor feasibility guarantees In general, there is no guarantee that a feasible control trajectory will be generated during the forward-calculation phase. It is typically assumed that interpolation of the state and control signal between elements in $\mathcal{X}_k$ and $\mathcal{U}_k$ gives a feasible and close-to optimal control sequence. However, it is possible that selecting this interpolated control signal will lead to a state leaving the feasible set. For problems where the optimal state trajectory lies along the boundary of the feasible set this issue is particularly problematic.

Large search space For problems where the state variable dynamics have very different time scales, the search space quickly becomes unreasonably large for the standard DP algorithm. For example, consider a system where an electric vehicle's velocity and battery state-of-charge (SOC) are modeled as state variables and the vehicle's acceleration is available as an input. Intuitively we expect the velocity to exhibit dynamics on the order of seconds, while the SOC shows dynamics on the order of minutes or hours. Solving this problem directly with IDP requires;

- A time step, 1 second, which is sufficiently small to resolve the fast dynamics of the vehicle velocity.
- That the set of control inputs is dense enough. For a 1% discretization error 100 control signals must be tested for each state configuration.
- A grid density for the vehicle velocity that is dense enough to ensure *reachability*; defined as the ability to *reach* at least one feasible neighboring state — the "nearest" state — at the next sample, given the entire range of state configurations and control signals applied to the system at each sample. In this problem, assuming a velocity grid covering the range 0 – $30\,\mathrm{m/s}$ and a maximum acceleration of $3\,\mathrm{m/s^2}$, 11 equidistant points are required in the velocity grid with the chosen sample rate. As it is crucial that *reachability* is ensured it is generally good practice to inflate this value slightly to take numerical precision into account. Assume a grid density of 15 points is sufficient.
- A grid density for the vehicle SOC that is dense enough to ensure *reachability*. Assuming a battery capacity of $30\,\mathrm{kWh} = 108\,\mathrm{MJ}$ and a maximum power demand of $90\,\mathrm{kW}$, this implies that each grid point must be separated by at most $90\,\mathrm{kW}/1\,\mathrm{s} = 90\,\mathrm{kJ}$. This in turns implies a minimum of $108\,\mathrm{MJ}/90\,\mathrm{kJ} = 1200$ points for the SOC grid. For some added headroom, assume a grid density of 1400 points.

Solving this problem with IDP results in that for every sample k there will be $100 \cdot 15 \cdot 1400 = 2.1 \cdot 10^6$ evaluations of f, b_{in}, and c — the equations that define the system dynamics, constraints, and cost defined in (1). For a time horizon of 1 hour, which captures the dynamics of the SOC, this would imply that the total optimization problem involves $2.1 \cdot 10^6 \cdot 60 \cdot 60 \approx 7.6 \cdot 10^9$ evaluations of the system model for each IDP iteration.

Note that this problem is "only" a 2+1–dimensional problem (two state variables, one control variable), and higher-dimensional problems grow exponentially in complexity. For systems with radically different time scales this becomes computationally exhausting as small time-steps are needed along with a prohibitively large grid for the slowly varying state variable(s).

2. MULTI-PASS ITERATIVE DYNAMIC PROGRAMMING WITH REGULARIZATION (IDP-MP)

The following modifications to the standard IDP scheme, collectively referred to as IDP-MP, mitigate the feasibility and complexity issues described in Section 1.3.

2.1 Improving feasibility with trajectory regularization

Let $\mathcal{F}_k$ denote the feasible set of $x\,[k]$, i.e. the set of $x\,[k]$ where there exists a control trajectory $u_{opt}\,(x\,[k])$ that satisfies the problem's constraints defined in (1). Let $\mathcal{G}_k$ denote the infeasible set of $x\,[k]$, i.e. the set of $x\,[k]$ where there does not exist a state trajectory satisfying (1). Define $J^*_{k,N_s}\,(x\,[k])$ as the minimum total cumulative cost from sample k to sample N_s when following the optimal control trajectory from $x\,[k] \in \mathcal{F}_k$. For convenience, define $J^*_{k,N_s}\,(x\,[k]) = \infty\, \forall x\,[k] \in \mathcal{G}_k$. Finally, define $\mathcal{O}_k$ as the set of values that $x\,[k]$ is allowed to cover in (1), i.e. the values of $x\,[k]$ that satisfies $b_{in}\,(\dots, k)$. Define a recursive regularized cost function $\hat{J}^*_{k,N_s}$ and constraint function $\hat{b}_{in}\,(\dots)$ that replaces the terms J^*_{k,N_s} and $b_{in}\,(\dots)$ in (1) during the back-calculation phase in the DP algorithm as

$$
\begin{aligned}
\hat{J}^*_{k,N_s}\,(x\,[k]) &= \min_{u[k]\in\mathcal{U}_k} \left[c\,(x\,[k]\,, u\,[k]\,, k) \right. \\
&\quad \left. + \hat{J}^*_{k+1,N_s}\,(x\,[k+1]) + \mu\right] \\
\hat{J}^*_{N_s,N_s} &= \min_{u[N_s]\in\mathcal{U}_{N_s}} c\,(x\,[N_s]\,, u\,[N_s]\,, N_s) + \mu \\
&\text{s.t.} \\
\hat{b}_{in}\,(u\,[k]\,, k) &\leq 0,\; k = [0, N_s] \\
x\,[k+1] &= f\,(x\,[k]\,, u\,[k]\,, k) \\
\mu &= \begin{cases} 0 & d_{min} > d_{thrs} \wedge x\,[k] \in \mathcal{O}_k \\ \beta & \text{otherwise} \end{cases} \\
d_{min} &= \min_{I_G \in \mathcal{I}_G} \|I_x - I_G\|\,,
\end{aligned}
\tag{2}
$$

where β is a sufficiently large additive penalization factor, I_x is the grid coordinate of a given element $x\,[k]$, $\mathcal{I}_G$ contains the grid coordinates for all elements in $\mathcal{G}_k$, and $\hat{b}_{in}\,(\dots)$ does not depend on $x\,[k]$ but otherwise has the same constrains as $b_{in}\,(\dots)$ in (1). In essence, (2) recursively generates the optimal trajectory by removing any hard constraints on x and instead adding a penalty β for every sample $k \in [0, N_s]$ where the optimal trajectory $J^*_{k,N_s}\,(x\,[k])$ either exceeds $\mathcal{O}_k$ (i.e. violates the original constraints $b_{in}\,(\dots)$) or the distance between $x\,[k]$ and the nearest infeasible state is less than d_{min}. The $d_{min} > d_{thrs}$ term is akin to the use of a barrier function (Bertsekas, 1999, p. 370), though in this application there is no need for it to be continuous or differentiable.

Figure 2 illustrates example values of d_{min} for a two-state problem with a search space of six orthogonal and uniformly distributed values for each state variable, using the 2-norm for defining the distance d_{min}, where the minimum distance between any two grid coordinates is defined as 1. In this example, $\mathcal{I}_G = \{[1,1]\,, [2,2]\,, \dots\}$ (indicated by the red diamonds) and the feasible state grid coordinates are $\{[2,1]\,, [3,1]\,, \dots\}$ (indicated by differently colored circles). For $d_{thrs} = 1.5$ the dashed blue line indicates the boundary between the penalized region (to the left) and the unpenalized region (to the right).

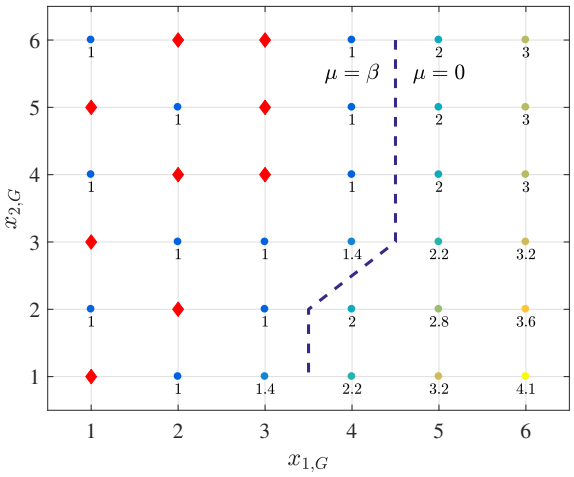

Fig. 2. Feasible (round) and infeasible (diamond) elements for an exemplified $\mathcal{X}_k$ with two variables consisting of six points each, with d_{min}'s value in (2) for each feasible element using the 2-norm. The regions corresponding to $\mu = \beta$ and $\mu = 0$ are indicated for $d_{thrs} = 1.5$.

As the hard constraints on $x[k]$ in (1) are replaced by soft constraints in (2), the problem feasibility is significantly improved for problems where the only feasible state trajectory lies near the state variable bounds (e.g. some bang-bang-control problems, (E. Bryson and Ho, 1975, p. 112) among others). Note that there is no need to add soft constraints to $u[k]$ as these values can directly be chosen, unlike $x[k]$ which evolves over k. (Note that this constraint replacement method is conceptually similar to that of (Bertsekas, 1999, p. 281).)

Determining values for the regularization parameters d_{thrs} and β is not obvious; the examples in this paper use $\beta = \mathrm{c}_{max} - \mathrm{c}_{min}$ (where $\mathrm{c}_{max} = \max c(\dots)$ and $\mathrm{c}_{min} = \min c(\dots)$, i.e. the maximum and minimum possible sample costs respectively), and 2-norm grid index distances $\sqrt{2} \leq d_{thrs} \leq 5$.

Typically, active regularization terms will generate suboptimal trajectories. However, with successive iterations, the suboptimal path typically asymptotically approaches the true optimal path as

- the suboptimal state and control trajectories introduced by the $d_{min} > d_{thrs}$ condition generates a path that is typically at most d_{thrs} grid indices away from the true optimal path, and with a decreasing grid extent (i.e. a finer grid) the absolute deviation decreases in proportion, and
- an optimal trajectory will only leave $\mathcal{O}_{k+1}$ if there only exists feasible, but penalized, trajectories from $x[k]$ (i.e. trajectories that violate the constraints in (1)). With successive IDP iterations the elements in $\mathcal{U}_k$ and $\mathcal{X}_{k+1}$ will be more closely spaced and a control $u[k]$ that brings $x[k] \to x[k+1]$ such that $x[k+1] \in \mathcal{O}_{k+1}$ will typically be generated if it exists.

Essentially, with a sufficiently fine grid, and for problems where the region of feasible trajectories is not pathologically shaped, the penalization term β will not be applied, and the initial problem's constraints, (1), will not be violated.

2.2 Improving feasibility with heuristic increase of grid search space

Despite the regularization terms used to improve the feasibility of the problem, as described in Section 2.1, it is possible that a feasible state/control trajectory is not found after reducing the grid size (for example, an unfortunately chosen grid may generate an $\mathcal{X}_k$ and/or $\mathcal{U}_k$ whose members are poorly amenable to interpolation). For the $p+1$'th IDP iteration, let $\mu_{dec} < 1$ and $\mu_{inc} > 1$ be the possible factors to scale $\mathcal{X}_k$ and $\mathcal{U}_k$ with for each sample k. For a given scaling factor, center $\mathcal{X}_k/\mathcal{U}_k$ about the result for the p'th IDP iteration, while leaving the number of grid elements unchanged. Here μ_{dec} is analogous to $1-\varepsilon$ in Luus (1990).

In this event, a simple heuristic that attempts to eventually generate feasible trajectories is to scale the grid size by μ_{dec} if the previous iteration generated a feasible result and by μ_{inc} if it did not. A good practice for this value is to choose a value that only very slightly increases the grid, i.e. $1 < \mu_{inc} \ll 1/\mu_{dec}$, and selecting μ_{inc} such that $\mu_{inc}^n \neq 1/\mu_{dec}\, \forall n \in \mathbb{Z}$, i.e. select μ_{inc} and μ_{dec} to inhibit repeating cyclic sequences of grid sizes given continuously infeasible iterations after an initially successful iteration. For example, if $\mu_{dec} = 0.8$, one appropriate value is $\mu_{inc} = 1.05$ as $\mu_{inc}^n = [1.05, 1.1025, 1.1576, 1.2155, 1.2763, \dots]$, which does not contain $1/\mu_{dec} = 1.25$.

2.3 Solving the m-1-dimensional problem to reduce the search space

For problems that display loosely coupled states (in the sense that each state's dynamics are relatively independent of the other) and whose dynamics have very different time scales, one can intuitively expect that the state trajectory for the slowly varying state variable(s) is similar to the trajectory generated by solving a simplified problem, where the dynamics of the quickly-varying state variable(s) is neglected. (Note that problems that are stiff and/or chaotic — where the trajectory of the quickly varying state variable(s) has a large effect on the slowly varying state variable(s) — do not lend themselves to this method.)

For these types of problems, the execution time for the optimal control problem can be significantly reduced by,

a solving an approximate lower-dimensional problem using IDP with a possibly longer sample period, where the dynamics of the quickly-varying states are neglected (which is computationally much faster than the full-dimensional problem with a short sample period), followed by

b solving the full-dimensional problem with a sufficiently short sample period and grid spacing density (the latter due to *reachability* requirements) using IDP, where the search space for the slowly varying variable is limited to some region in the vicinity of the solution from the low-dimensional problem *a*.

This procedure allows for significantly reducing the number of elements in $\mathcal{X}$ that correspond to the slowly-varying variable(s) in stage *b*, while maintaining optimality so long as the true optimal control solution for the slowly varying state variable(s) lies within the range of values searched in stage *b*. This method can be easily extended

to m-dimensional problems with n different sets of state variables with different time scales.

The following exemplifies how stages a and b can be applied to the vehicle control problem defined in Section 1.3.

Apply stage a; Solve a 1-state/1-input problem using an equivalent model where the SOC is modeled as a state variable and the model input is the vehicle power (corresponding to the power required to maintain a constant speed). This approximation is equivalent to neglecting the dynamics related to changes in vehicle velocity, i.e. neglecting the vehicle's kinetic energy and allowing the vehicle velocity to be discontinuous. For a longer sample period of 2 seconds (which shifts computational burden from stage a to stage b) *reachability* implies that a grid density of $108\,\mathrm{MJ}/\left(90\,\mathrm{kW}\cdot 2\,\mathrm{s}\right) = 600$ points is required. Assuming 700 points are used this gives a total of $700 \cdot 60 \cdot 60/2 = 1.2 \cdot 10^6$ model evaluations per IDP iteration. Doubling the sample period for this stage results in reducing the number of computations in this stage by a factor of four.

Apply stage b; Solve the full 2-state/1-input problem (i.e. model SOC and velocity as states with acceleration as an input) with a reduced SOC search range. For a 1 second sample rate *reachability* yields a SOC grid separation of 90 kJ, though now a smaller SOC range can be searched compared to the original problem in Section 1.3. A conservative lower bound for the SOC variable extent is to require the permissible variation in battery energy ΔE to be able to accelerate/decelerate the vehicle from zero to maximum speed or vice versa. For equal sample rates in stages a and b and a vehicle mass of 1000 kg, this results in a battery search range of $\pm\Delta E \geq \frac{1}{2}\cdot 10^3\left(30\right)^2 = \pm 450\,\mathrm{kJ}$ centered about the SOC trajectory determined in stage a. In this example, the sample period in stage a was twice the sample period of this stage, suggesting that a search range on the order of $\pm 450 \cdot 2 = \pm 900\,\mathrm{kJ}$ is sufficient, giving a minimum of $2 \cdot 900\,\mathrm{kJ}/90\,\mathrm{kJ} = 20$ grid points (i.e. doubling the sample period in stage a doubles the computational burden in this stage). For some added headroom, assume 24 grid points are used. This gives a total of $100{\cdot}15{\cdot}24 = 36{\cdot}10^3$ state/control combinations to test at each sample, which causes the final two-dimensional problem to consist of $36 \cdot 10^3 \cdot 60 \cdot 60 = 130 \cdot 10^6$ calls to the system model.

In this example, IDP-MP requires $1.2 \cdot 10^6$ and $130 \cdot 10^6$ model evaluations in stage a and b respectively, in contrast to the $7.56 \cdot 10^9$ model evaluations per iteration for the standard IDP method. The total number of model evaluations is reduced by a factor of 58, with a similar reduction in execution time.

In fact, for this particular problem, shifting computational burden from stage a to stage b worsens the net performance, as $\pm\Delta E$ is "only" two orders of magnitude smaller than the total battery capacity. For a 1-second sample rate in both stages, stage a and b would require $4.8 \cdot 10^6$ and $65 \cdot 10^6$ model evaluations respectively, reducing the total number of model evaluations by a factor of 108 compared to the standard IDP method. Naturally, the reduction in computational time is a result of the chosen numerical values in this example, and it will be shown in Section 3.2 that some problems may exhibit a much greater performance improvement.

Note that, in general, determining the minimum search range for stage b and optimally balancing the computational burden between stages a and b is beyond the scope of this paper.

3. RESULTS

This section highlights the benefits of the method presented in Section 2 by solving two optimal-control problems using a Matlab implementation of the presented IDP-MP algorithm.

3.1 Double-integrator

Assume the goal is to optimally control a sampled constrained double integrator with different time scales given by

$$\begin{aligned}
x_2\left[k+1\right] &= x_2\left[k\right] + \alpha T_s u\left[k\right] \\
x_1\left[k+1\right] &= x_1\left[k\right] + T_s x_2\left[k\right] + \alpha u\left[k\right]\frac{T_s^2}{2} \\
c\left[k\right] &= \left(\left|x_1\left[k\right]\right| + \left|\frac{1}{10}x_2\left[k\right]\right|\right) T_s \\
&\text{s.t.} \\
T_s &= 0.125 \\
N_s &= 3/T_s \\
\alpha &= 4 \\
u\left[k\right] &\in \left[-1,1\right] \\
x_2\left[k\right] &\in \left[-1,1\right] \\
\begin{bmatrix} x_1\left[0\right] \\ -x_2\left[0\right] \\ x_1\left[N_s\right] \\ x_2\left[N_s\right] \end{bmatrix} &\geq \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix},
\end{aligned} \tag{3}$$

where α is the difference in time scales between x_2 and x_1, T_s is a constant sampling period, and the magnitudes of x_1 and x_2 are penalized. The penalization terms are selected so that the resulting optimal trajectory is identical to that of a problem using the cost function $c\left[k\right] = \left|x_1\left[k\right]\right| T_s$, while ensuring numerical stability in regions where this simpler cost function displays singular control. Several attributes make this problem a suitable example that illustrates the benefits of the presented method: the optimal control solution is known analytically, the state trajectory of the optimal solution lies on the boundary of the feasible set (as x_2 will be ± 1 for some time), and with $\alpha = 4$ this problem is relatively loosely coupled with respect to x_1 and x_2.

Using the IDP-MP method defined in Section 2.3, some reduction in the search space for x_1 is possible if an approximation of (3) can be generated where the quickly varying state variable is replaced by a control signal. One example of a system that does this is

$$\begin{matrix} x\left[k+1\right] = x\left[k\right] + T_s u\left[k\right] \\ c\left(x\right) = \left|x\left[k\right]\right| T_s \end{matrix} \quad \text{s.t.} \quad \begin{matrix} u\left[k\right] \in \left[-1,1\right] \\ \begin{bmatrix} x\left[0\right] \\ x\left[N_s\right] \end{bmatrix} \geq \begin{bmatrix} 1 \\ 1 \end{bmatrix} \end{matrix}, \tag{4}$$

where x_2 has been directly replaced with an input u. This system is a good approximation of the dynamics of the slowly changing state variable in (3), i.e. a constrained integrator. Furthermore, (4) is easy to solve using an ordinary IDP algorithm; define $\hat{x}\left[k\right]$, $k = \left[0, N_s\right]$ as the solution given by IDP.

Now, the two-dimensional problem in (3) can be solved and the search space can be reduced for x_1 by letting $\mathcal{X}_k^1 = \left\{\hat{x}\left[k\right] + \Delta x_1 \frac{n}{N_{g1}}\right\}_{n=-N_{g1}}^{N_{g1}}$. Here, $\mathcal{X}_k^1$ is a set of $2N_{g1}+1$ linearly distributed elements corresponding to x_1 that will be tested at sample k, Δx_1 is the extent of the values to test, and $\hat{x}\left[k\right]$ is the generated solution of the 1-state optimal control problem in (4). The particular choice of linearly distributed elements in $\mathcal{X}_k^1$ is convenient, but ultimately arbitrary; the range-reducing method works well for other choices. $\mathcal{X}_k$ can be generated in a number of ways, one practical method is to set $\mathcal{X}_k = \left\{(a,b)\,|a \in \mathcal{X}_k^1 \text{ and } b \in \mathcal{X}_k^2\right\}$, where $\mathcal{X}_k^2 = \left\{\frac{n}{N_{g2}}\right\}_{n=-N_{g2}}^{N_{g2}}$. Setting $\Delta x_1 = \left(2 \cdot \text{range}\left(x_1\right)\right)/\alpha = 1.5$ results in reducing the number of model evaluations by a factor of two, and the reduction is primarily limited by the quotient of the time scales, α.

The reduced search space decreases the computational burden of the problem, however, this also reduces the feasible set. If reduced too aggressively, the optimal solution will not be completely contained in the search space and solution optimality may be lost. The feasibility of the problem can be improved by introducing the regularization terms defined in Section 2.1. For $\beta = \text{c}_{\max} - \text{c}_{\min} = \max c\left(\dots\right) - \min c\left(\dots\right) = 1.1 - 0 = 1.1$, $d_{min} = \sqrt{2}$, using the 2-norm for distance, and an increased search space of $x_2 \in [-1.25, 1.25]$ the feasibility issues are mitigated in this example. It is beyond the scope of this paper to determine the minimum range that can be searched that is guaranteed to contain the optimal solution.

Solving this problem using IDP-MP gives the state trajectory shown in Figure 3 after 50 iterations, taking approximately 10 minutes on a typical desktop computer (primarily CPU-bound, single-core execution on an AMD FX-6300). The feasible set and optimal trajectories for sample $k = 10$ for the first two-dimensional iteration is shown in Figure 4. As can be seen, the results in both figures match the well-known optimal bang-bang control and state space trajectory (E. Bryson and Ho, 1975, p. 112).

Some noteworthy attributes are:

- the optimal control trajectory for states near the infeasible region tend to exhibit a larger magnitude (i.e. the system is more quickly brought to a state at least d_{min} grid indices away from the infeasible bound if such a control exists),
- for states where $x_1 = -0.7$ the optimal control exceeds the soft constraints $x_2 \in [-1, 1]$, implying that these states would belong to the infeasible set had the allowable range not been extended to $x_2 \in [-1.25, 1.25]$,
- the state $[x_1, x_2] = [-0.6, 1.1]$ has an optimal transition to $[x_1, x_2] \approx [-0.5, 1]$, i.e. the optimal control brings the state out of soft constraint violation (i.e. keeps $x_2 \in [-1, 1]$) if there exists a trajectory that does this.

For more details on the numerical values used and the implementation of the IDP-MP solver, see `https://github.com/lerneaenhydra/dpm`.

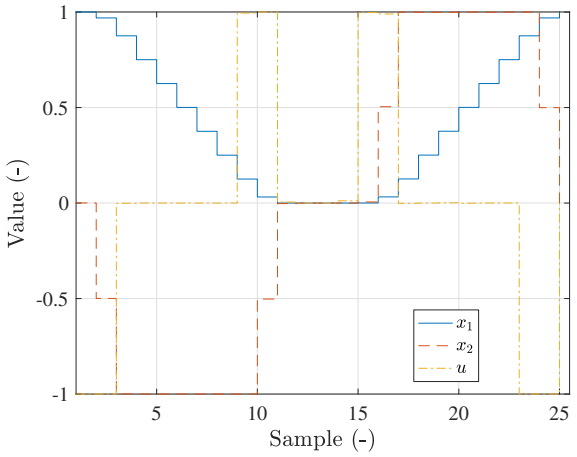

Fig. 3. The state and control trajectory solution for the double integrator (3). This result asymptotically approaches the analytic solution.

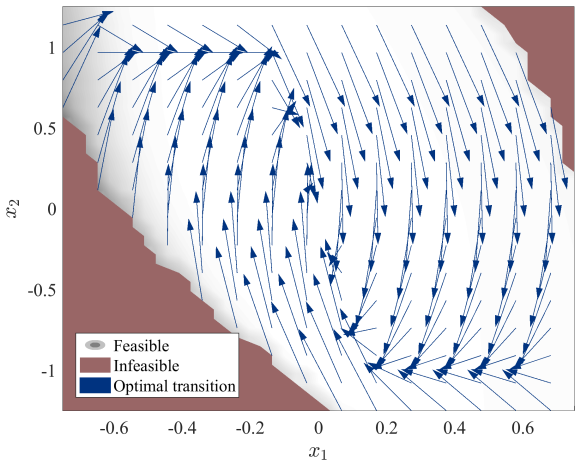

Fig. 4. Optimal trajectory map for the first iteration of the system defined by (3) for sample $k = 10$. Each arrow indicates the optimal state transition from the current sample to the next sample. The feasible set is shown in the greyscale region, with darker areas indicating a larger cost J^*_{13,N_s}.

3.2 Hybrid vehicle example

Determining the optimal control policy for hybrid vehicles is one application that often utilizes DP (Liu and Peng (2008); Pérez et al. (2006)). Typically, this problem is solved for one state variable, namely the SOC, as the search space grows prohibitively quickly with additional state or control variables. However, for a multi-dimensional problem with loosely coupled states with different time scales, the IDP-MP method defined in Section 2 can be effectively used.

Consider a passenger car series-hybrid where the combustion engine crankshaft velocity, ω, and SOC are modeled as state variables. A block diagram illustrating the model is shown in Figure 5, where the internal combustion engine torque, generator torque, and total power demand are model inputs; SOC and ω are state variables; and the SOC and instantaneous fuel consumption $\dot{m}$ are model outputs.

As the crankshaft velocity is loosely coupled and exhibits dynamics several orders of magnitude faster than the SOC,

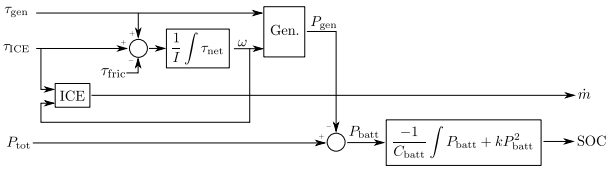

Fig. 5. Simple model of a series-hybrid vehicle.

this problem is amenable to IDP-MP. Solving for model parameters that are representative of a typical passenger car, power requirements given by the US-06 drive cycle (US Environmental Protection Agency (2008)), a 50% SOC at the start and end of the cycle, and solely penalizing fuel consumption gives results shown in Figure 6. (See `https://github.com/lerneaenhydra/dpm` for a full definition of the model set-up.)

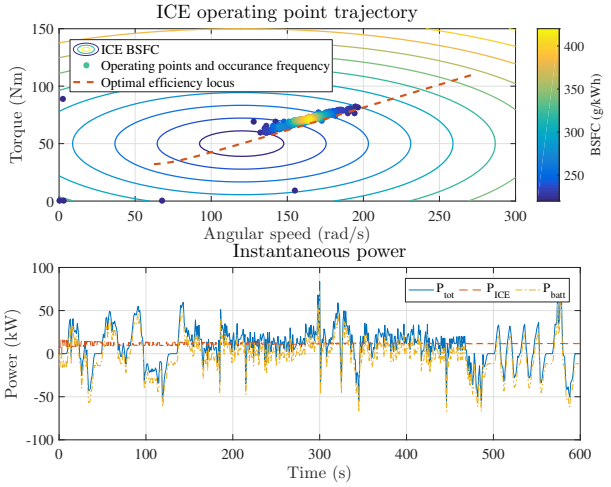

Fig. 6. Internal combustion engine (ICE) operating point and instantaneous subsystem power for the vehicle topology defined in Figure 5 subjected to the US-06 drive cycle. The upper plot shows isolines of the engine's brake specific fuel consumption (BSFC) and operating point at each sample. More frequent points are colored yellow while less frequent points are colored blue. The engine is operated mostly along the optimal-efficiency locus, where the most frequent operating point is 71 Nm and 164 rad/s. The lower plot displays the total, battery, and generator power over time.

This example was solved using both the IDP-MP method as well as a standard IDP method. Here, IDP-MP used 32 times fewer model evaluations than IDP per iteration, and the calculation time was reduced by a similar factor for each iteration. The computational reduction is limited primarily by the quotient of time scales between the two state variables. In this example the battery capacity has been kept as small as possible to let the standard method generate a solution in a reasonable time frame, and a battery time scale on the order of 42 seconds was sufficient (i.e. the battery was sized so that, at maximum power draw, completely depleting the battery from a full SOC takes 42 seconds). The crankshaft displays a time scale on the order of one second, so the state variables time-scales differ by a factor of 42, which matches well with the difference in the number of model evaluations between IDP-MP and IDP. Had the battery time scale instead been on the order of an hour, which is more physically realistic, each IDP-MP iteration would approximately use $1/\left(32/42 \cdot 60 \cdot 60\right) \approx 1/2700$ as many system model evaluations as the standard IDP method (as *reachability* implies the battery energy grid density must be constant while the grid range is increased by a factor of approximately $60 \cdot 60/42 \approx 86$).

The IDP-MP method has been verified for this particular example by comparing the results to those given by a standard IDP method. Figure 7 displays the total cost J^*_{0,N_s} for successive IDP iterations. The mean relative difference between the state trajectories of the 7'th iteration of IDP-MP and 5'th iteration of IDP (which have been selected due to their similar total cost) is 0.764% and is believed to primarily be due to the inherent variable quantization of the states and controls. Even though IDP-MP generates inferior results for a given number of iterations the significantly reduced computation time ensures that a result equally good as that given by IDP is generated after a shorter execution time.

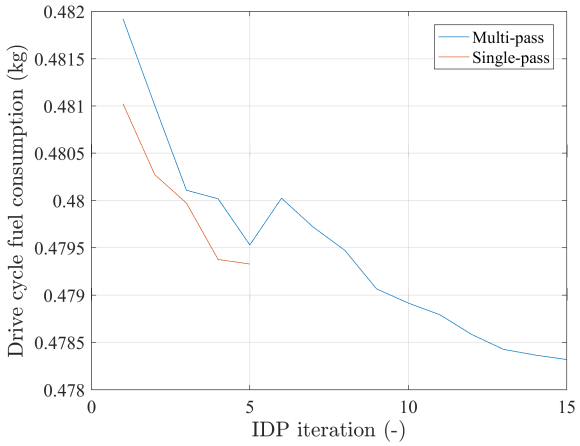

Fig. 7. Net cost (fuel consumption) for the system topology shown in Figure 5 for successive iterations. IDP-MP gives a slightly worse trajectory than IDP for a given iteration, but eventually surpasses IDP while using 1/32 as many model evaluations per iteration. Only five IDP iterations are shown due to the prohibitive execution time, taking approximately ten times as long to calculate as all the IDP-MP iterations.

4. CONCLUSIONS

It has been shown that the IDP-MP method can greatly decrease the computational time required to solve optimal control problems for systems where the system dynamics are both loosely coupled and display significantly different time scales. Typically, the execution time is decreased by a factor on the order of the quotient of the system's time scales. As problems with very different time scales are computationally difficult to solve using IDP, this method is a significant improvement over IDP as these problems are where the largest performance gains with IDP-MP are found. Broadly speaking, the IDP-MP method described in this paper increases the range of optimal control problems that DP/IDP is suited for and, as a result of the large performance improvement, significantly more complex problems can be considered.

REFERENCES

Åström, K.J. and Wittenmark, B. (1997). *Computer-controlled Systems (3rd Ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Bellman, R. and Dreyfus, S. (2015). *Applied Dynamic Programming*. Princeton Legacy Library. Princeton University Press.

Bellman, R. (1956). Dynamic programming and lagrange multipliers. *Proceedings of the National Academy of Sciences*, 42(10), 767–769.

Bertsekas, D.P. (1999). *Nonlinear Programming*. Athena Scientific.

Bertsekas, D.P. (2005). *Dynamic programming and optimal control. Volume I*. Athena Scientific optimization and computation series. Belmont, Mass. Athena Scientific.

E. Bryson, Jr, A. and Ho, Y.C. (1975). *Applied Optimal Control*. Taylor & Francis Group, 270 Madison Avenue, New York, NY.

Elbert, P., Ebbesen, S., and Guzzella, L. (2013). Implementation of Dynamic Programming for n-Dimensional Optimal Control Problems With Final State Constraints. *IEEE Transactions on Control Systems Technology*, 21(3), 924–931.

Liu, J. and Peng, H. (2008). Modeling and control of a power-split hybrid vehicle. *IEEE Transactions on Control Systems Technology*, 16(6), 1242–1251.

Luus, R. (1990). Optimal control by dynamic programming using systematic reduction in grid size. *International Journal of Control*, 51(5), 995–1013.

Pérez, L.V., Bossio, G.R., Moitre, D., and Garcia, G.O. (2006). Optimization of power management in an hybrid electric vehicle using dynamic programming. *Mathematics and Computers in Simulation*, 73(1-4), 244–254.

Sciarretta, A. and Guzzella, L. (2007). Control of hybrid electric vehicles. *IEEE Control Systems*, 27(2), 60–70.

Sundström, O., Ambühl, D., and Guzzella, L. (2010). On Implementation of Dynamic Programming for Optimal Control Problems with Final State Constraints. *Oil & Gas Science and Technology — Revue de l'Institut Franccais du Pétrole*, 65(1), 91–102.

Sundström, O. and Guzzella, L. (2009). A generic dynamic programming matlab function. In *2009 IEEE Control Applications,(CCA) & Intelligent Control,(ISIC)*, 1625–1630. IEEE.

US Environmental Protection Agency (2008). US06 Dynamometer Drive Schedule. `https://www.epa.gov/vehicle-and-fuel-emissions-testing/dynamometer-drive-schedules`. [Online; accessed 2016-10-31].

Wahl, H.G. and Gauterin, F. (2013). An iterative dynamic programming approach for the global optimal control of hybrid electric vehicles under real-time constraints. In *Intelligent Vehicles Symposium (IV), 2013 IEEE*, 592–597. IEEE.