Supplementary material for "Continuous-action planning for discounted infinite-horizon nonlinear optimal control with Lipschitz values"

List of main notations

x, X, u, U	state, state space, action, action space
$f, \boldsymbol{u}$	dynamics, sequence of actions
γ, r, ρ, v	discount factor, reward, reward fcn., value
L_f, L_ρ, L_v	Lipschitz constants of respective fcns.
$\mathcal{U}$	box in the space of action sequences
$i, k; i^\dagger, k^\dagger$	box and dimension indices; selected indices
K	number of discretized box dimensions
μ, d	action interval, interval length
n	computation budget
M	number of subintervals for splitting
$\mathcal{T}, \mathcal{T}^*, \mathcal{L}$	tree, near-optimal tree, set of leaves
h, s	depth in tree, number of splits
$b(i), \delta(i)$	b-value, diameter of box i
m	branching factor of near-optimal tree
$h_{\max}(n)$	maximum depth function
τ, c, C	constants
$O(\cdot), \tilde{O}(\cdot)$	bounded by $\cdot$ up to const., log. terms
$\lceil\cdot\rceil$	ceiling (smallest integer still larger than $\cdot$)
diag$[\cdot]$	diagonal matrix with $\cdot$ on diagonal

A Proofs

Proof of Lemma 3: We will prove the inequality from the theorem for any two sequences $\boldsymbol{u}_\infty$ and $\boldsymbol{u}'_\infty$, which directly implies it for the particular case $\boldsymbol{u}'_\infty = \boldsymbol{u}^*_\infty$ from (3). Denote by x_k, x'_k, r_k, r'_k for $k \geq 0$ the state and reward trajectories generated by the two sequences, with $x'_0 = x_0$. Then, by induction:

$$\|x_k - x'_k\| \leq \sum_{j=0}^{k-1} L_f^{k-j} \left|u_j - u'_j\right| \tag{A.1}$$

Indeed, at $k = 1$, we have:

$$\|x_1 - x'_1\| \leq L_f(\|x_0 - x'_0\| + |u_0 - u'_0|) = L_f\,|u_0 - u'_0|$$

and assuming the relation holds at $k-1$, we get:

$$\begin{aligned}\|x_k - x'_k\| &\leq L_f(\left\|x_{k-1} - x'_{k-1}\right\| + \left|u_{k-1} - u'_{k-1}\right|)\\ &\leq L_f \sum_{j=0}^{k-2} L_f^{k-1-j}\left|u_j - u'_j\right| + L_f\left|u_{k-1} - u'_{k-1}\right|\end{aligned}$$

equivalent to (A.1).

Then:

$$\begin{aligned}|v(\boldsymbol{u}_\infty) - v(\boldsymbol{u}'_\infty)| &\leq \sum_{k=0} \gamma^k \left|r_k - r'_k\right|\\ &\leq \sum_{k=0}^{\infty} \gamma^k L_\rho(\|x_k - x'_k\| + |u_k - u'_k|)\\ &\leq L_\rho \sum_{k=0}^{\infty}\gamma^k \sum_{j=0}^{k} L_f^{k-j}\left|u_j - u'_j\right|\\ &= L_\rho \sum_{k=0}^{\infty}|u_k - u'_k| \sum_{j=0}^{\infty}\gamma^{k+j}L_f^j\\ &= \frac{L_\rho}{1-\gamma L_f}\sum_{k=0}^{\infty}\gamma^k\left|u_k - u'_k\right|\end{aligned}$$

Here, we used the Lipschitz continuity of ρ, and then applied (A.1). The three equalities simply rewrite the right hand side, isolating the contribution of each $|u_k - u'_k|$; the middle equality swaps indices k and j, and can be verified by writing out explicitly the summations. The last step holds because $\gamma L_f < 1$. Thus the inequality is proven with $L_v = \frac{L_\rho}{1-\gamma L_f}$. □

Proof of Theorem 6: Our goal is to find an upper bound for the diameter of an arbitrary box i at some depth h. Since we deal with this single box, we will omit its index i for most of the derivation. Indeed, due to dimension selection (7), all the boxes at a given depth have the same shape. Recall function $s(k)$, the number of splits per dimension k, which is clearly decreasing, see Fig. A.1 for an example. In addition s decreases in steps of at most 1. To see this, consider k like in the figure, the first dimension in a constant-s range, which will always be preferred to later dimensions in the same range. To increase the gap to 2, dimension $k-1$ must be expanded before k, which means $\gamma^{k-1}M^{-(s(k)+1)} \geq \gamma^k M^{-s(k)}$, or $M \leq 1/\gamma$. This contradicts Assumption 5(ii).

Denote now the lengths of the ranges in s by $\tau_0, \tau_1, \ldots, \tau_N$ where N is the last, infinitely long range where $s = 0$. Let j be the index of the range starting with k. By direct computation like above, we find that if node k is expanded:

$$\tau_j \geq \frac{\log M}{\log 1/\gamma}, \quad \tau_{j-1} < \frac{\log M}{\log 1/\gamma} \text{ if } j \geq 1 \tag{A.2}$$

Recall that $\tau = \left\lceil \frac{\log M}{\log 1/\gamma}\right\rceil \geq 2$, so (A.2) implies $\tau_j \geq \tau, \tau_{j-1} < \tau - 1$. Keeping in mind that s (and so N, τ_j) depend on h, we prove by induction that at any depth h, we have:

$$\begin{cases}\tau_0 \leq \tau\\ \tau_j \in \{\tau-1, \tau\} \quad \text{for } 1 \leq j < N\\ \tau_N = \infty\end{cases} \tag{A.3}$$

For any τ, the first $h = 3\tau+2$ are done in the same order, shown in the figure. To see this, consider any step and try to expand any other dimension; one finds that (A.2) is violated. Clearly any function s so far obeys (A.3). Consider now an arbitrary $h \geq 3\tau+2$ that satisfies (A.3). To study $h + 1$, four cases must be considered for the next expanded dimension, denoted (a)-(d) inside dotted squares in the figure. Call the next function s', with N' ranges τ'_j.

Case (a): Since $k = 1$ is expanded, $\tau_0 = \tau$ due to (A.2), (A.3). Hence $\tau'_0 = 1$, $\tau'_1 = \tau - 1$, and later ranges remain unchanged. Case (b): We must have $\tau_0 \leq \tau - 1, \tau_1 = \tau$. Hence, $\tau'_0 \leq \tau$, $\tau'_1 = \tau - 1$, and the later ranges are unchanged. Case (c): Some arbitrary k in some range $1 < j < N$ is expanded. We must have $\tau_{j-1} = \tau$, $\tau_j = \tau - 1$, leading to $\tau'_{j-1} = \tau - 1$, $\tau'_j = \tau$, and the other ranges remain the same. Case (d): Finally, when the first undiscretized dimension K is split, we have $\tau_{N-1} = \tau-1$, $\tau_N = \infty$, leading to $N' = N$, $\tau'_{N-1} = \tau$, $\tau'_N = \infty$, with no change to the prior ranges. In all cases, the new function satisfies (A.3), so the induction is complete.

The next step is to find, for fixed h, a lower bound on $s(k)$ under constraints (4), (A.3). If the length K of the box is

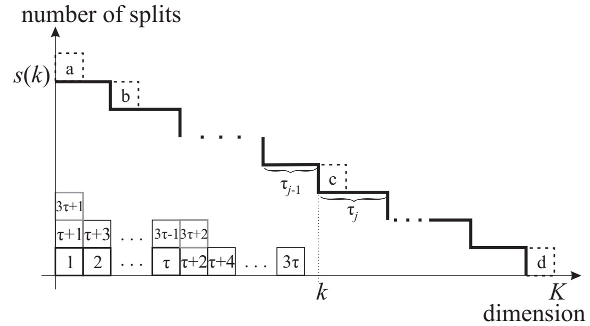

Fig. A.1. An example of a split function is shown as a thick black line. The first unsplit dimension is K, and a dimension k is highlighted with the constant-s ranges to its left and right, τ_{j-1} and τ_j. Each square at a certain value of k indicates a split of dimension k, and the squares stack together to obtain $s(k)$. The first $3\tau + 2$ splits, shown on the bottom-left, always occur in the same order, indicated by indices inside the squares. The dotted squares above $s(k)$ indicate possibilities for the next split of $s(k)$.

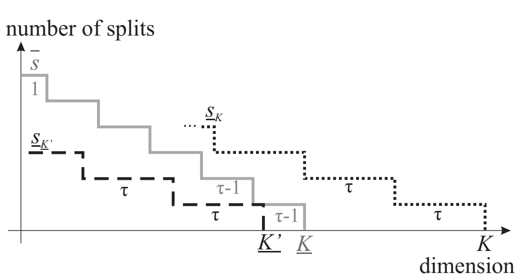

Fig. A.2. Various split function bounds used in the proof, see there for their meaning. Dotted: $\underline{s}_K$ for arbitrary K; gray: $\bar{s}$ yielding $\underline{K}$; dashed: $\underline{s}_{\underline{K}'}$ where $\underline{K}' \leq \underline{K}$.

fixed and (4) is ignored (thereby relaxing the problem), then a lower bound $\underline{s}_K$ on any function for this K is obtained by filling in s with ranges of τ; see Fig. A.2. Now $\underline{s}_K(k)$ is decreasing with K for any k, so we must find a lower bound on K, denoted $\underline{K}$. To this end, we fill in a function $\bar{s}$ with all ranges of $\tau - 1$, except τ_0 which is equal to 1, see again Fig. A.2; while imposing a relaxed version of (4), namely $h \leq \sum_{k=0}^{\infty} \bar{s}(k)$.

Denote N the number of ranges in $\bar{s}$, then by explicit computation we get:

$$h \leq \sum_{k=0}^{\infty} \bar{s}(k) = \frac{(N-1)N}{2}(\tau-1)+N \leq \frac{(N+1)^2}{2}(\tau-1)$$

from which we get $N \geq \sqrt{\frac{2h}{\tau-1}} - 1 =: \underline{N}$. Again by computation and replacing $\underline{N}$:

$$\underline{K} = 2 + (N-1)(\tau-1) \geq 4 - 2\tau + \sqrt{2h(\tau-1)} =: \underline{K}'$$

Since $\underline{s}_K = \left\lceil \frac{K-k}{\tau} \right\rceil \geq \frac{K-k}{\tau}$ for $k < K$, we finally obtain a lower-bound split function as $\underline{s}_{\underline{K}'}$, which at $k < \underline{K}'$ is explicitly:

$$\underline{s}_{\underline{K}'} = \sqrt{2h\frac{\tau-1}{\tau^2}} - \frac{k}{\tau} + (4/\tau - 2)$$

and 0 for $k \geq \underline{K}'$.

To complete the proof, we use this function in the diameter formula:

$$\begin{aligned}
\delta(i) &= \bar{L}_v \sum_{k=0}^{\infty} \gamma^k M^{-s(k)} \\
&\leq \bar{L}_v \left[\sum_{k=0}^{\underline{K}'-1} \gamma^k M^{-\underline{s}_{\underline{K}'}} + \frac{\gamma^{\underline{K}'}}{1-\gamma} \right] \\
&\leq \bar{L}_v \left[c_1 \sqrt{2h(\tau-1)} M^{-\sqrt{2h\frac{\tau-1}{\tau^2}}} + c_2 \gamma^{-\sqrt{2h\tau-1}} \right] \\
&\leq c\sqrt{2h(\tau-1)}\gamma^{\sqrt{2h\frac{\tau-1}{\tau^2}}}
\end{aligned}$$

where we have omitted some tedious derivations. An intermediate step was highlighted to show that some conservativeness is introduced in the tail term, notably by decreasing its rate of convergence via the division by τ^2. Here, c_1, c_2, c denote positive constants whose value is not important in the asymptotic analysis. $\square$

Proof of Lemma 8: For the first part, consider any iteration t, and denote by i_t the box expanded at this iteration. Since the leaf boxes on the current tree $\mathcal{T}$ cover the entire space, there exists a leaf box j containing an optimal solution, for which $b(j) \geq v^*$ by (6). As the expanded box i_t is selected by maximizing the b-value,

$b(i_t) \geq b(j) \geq v^*$. Further, $v(i_t) + \delta_h \geq b(i_t)$, where h is the depth of i_t, and therefore finally $i_t \in \mathcal{T}^*$.

For the second part, among the descendants of i_t there exists a leaf j on the final tree so that $v(j) \geq v(i_t)$, due to Assumption 5(i). Since $\hat{\boldsymbol{u}}$ maximizes the value among the leaves, $v(\hat{\boldsymbol{u}}) \geq v(j) \geq v(i_t)$. Combining this with $b(i_t) \geq v^*$ from the first part, we get $v^* - v(\hat{\boldsymbol{u}})) \leq b(i_t) - v(i_t) = \delta(i_t)$. Since this holds at any iteration, the bound $\delta_{\min} = \min_t \delta(i_t)$ follows. $\square$

Proof of Theorem 9: For a given budget n the branching factor is used to infer a lower bound on the depth reached by OPC. For $m > 1$, take the smallest depth h so that $Cm^{h+1}Mh \geq n$. The left-hand side is chosen since there are less than Cm^{h+1} nodes in the explored tree up to depth h, and each of them takes at most Mh computation to expand, see Fig. A.3. Therefore, we are sure

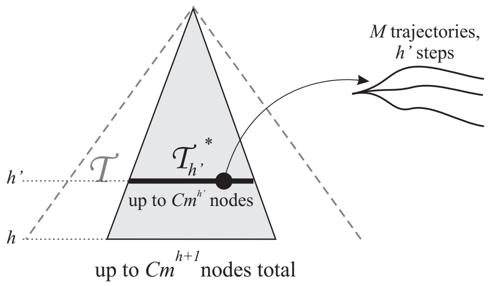

Fig. A.3. To reach depth h, at most $Cm^{h'}$ nodes must be expanded on $\mathcal{T}^*_{h'}$ at depths $h' \leq h$, for a total of Cm^{h+1}. Each of these expansions requires, at worst, simulating M trajectories with h' steps.

that at least a node at depth $h_{\min} = h - 1$ has been expanded, and by solving the inequality we get after some computation:

$$h_{\min} \geq \frac{\log n}{\log m} - \frac{\log \log (n\alpha)^\beta}{\log m}$$

for $\alpha, \beta > 0$ and sufficiently large n. Replacing this in the formula for $\delta_{h_{\min}}$, we obtain:

$$\begin{aligned}
\delta_{\min} \leq \delta_{h_{\min}} &= \tilde{O}\left(\gamma\hat{}\sqrt{2\frac{\tau-1}{\tau^2}\left[\frac{\log n}{\log m} - \frac{\log\log(n\alpha)^\beta}{\log m}\right]}\right) \\
&= \tilde{O}\left(\gamma^{\sqrt{\frac{2(\tau-1)\log n}{\tau^2 \log m}}} \cdot (\frac{1}{\gamma})\hat{}\sqrt{2\frac{\tau-1}{\tau^2}\frac{\log\log(n\alpha)^\beta}{\log m}}\right) \\
&= \tilde{O}\left(\gamma^{\sqrt{\frac{2(\tau-1)\log n}{\tau^2 \log m}}} \cdot (\frac{1}{\gamma})\hat{}(2\frac{\tau-1}{\tau^2}\frac{\log\log(n\alpha)^\beta}{\log m})\right) \\
&= \tilde{O}\left(\gamma^{\sqrt{\frac{2(\tau-1)\log n}{\tau^2 \log m}}} \cdot [\log(n\alpha)]^{\beta'}\right) \\
&= \tilde{O}\left(\gamma^{\sqrt{\frac{2(\tau-1)\log n}{\tau^2 \log m}}}\right)
\end{aligned}$$

where $\beta' > 0$ is a constant and we temporarily use notation $z\hat{}y = z^y$ for readability. In the second step, the square root can be split in this way for sufficiently large n; then we show that the second term enters in fact the logarithmic, negligible part of the expression.

When $m = 1$, take the smallest h so that $Ch^2M \geq n$. Since at most C nodes must be expanded at each depth, this guarantees that some node was expanded at depth $h_{\min} \geq \sqrt{n/MC} - 1$. Replacing this in the diameter formula leads to the desired result.

Note that we did not make explicit the term $\sqrt{2h_{\min}(\tau-1)}$ in the diameter bounds. However, trivial upper bounds for $h_{\min}$ are on the order of $\log n$ in the first case (since a tree with branching factor greater than 1 must asymptotically be explored, larger depths than this cannot be reached), and n in the second case. Both of these remain in the logarithmic part of the bound. $\square$

Proof of Lemma 10: Consider one top-level loop of SOPC in Alg. 2, and denote the number of elapsed such loops by ℓ. The major part of the proof will be to show by induction that, for any $h \leq h_{\max}(n)$, if $\ell \geq C\sum_{h'=0}^{h} m^{h'}$, then at least a box (tree node) containing an optimal sequence at h has been expanded.

For the base case, if $\ell \geq Cm^0 \geq 1$, then the root has been expanded, and it contains the optimal solution by definition. Take now an arbitrary $h < h_{\max}(n)$, and define ℓ_h to be the loop where the first node i_h^* at h that contains an optimal solution was expanded. By the induction hypothesis, $\ell_h \leq C\sum_{h'=0}^{h} m^{h'}$. Node i_h^* has itself some child i_{h+1} that contains an optimal solution. (Note that another optimal node i_{h+1}^* may be expanded before this particular child; we will be interested in i_{h+1}^* below.) Consider now any iteration $\ell_h + \ell'$ where some other node i'_{h+1}, different from i_{h+1}, is expanded at depth $h+1$. Then it must be that:

$$v(i'_{h+1}) \geq v(i_{h+1}) \geq v^* - \delta(i_{h+1}) \geq v^* - \delta_{h+1}$$

where the middle inequality holds because i_{h+1} contains an optimal solution. But then $i'_{h+1} \in \mathcal{T}^*_{h+1}$, and since there are at most Cm^{h+1} nodes in this set, including i_{h+1} which was not yet expanded, it means at most $Cm^{h+1} - 1$ loops can pass before i_{h+1} *must* be expanded. Thus the loop ℓ_{h+1} where i_{h+1}^* is expanded is, at worst, $\ell_h + Cm^{h+1} \leq C\sum_{h'=0}^{h+1} m^{h'}$, and the induction is proven.

Observe next that each loop ℓ expands at most $h_{\max}(n)$ nodes, where each node expansion takes at most

$Mh_{\max}(n)$ model calls. Then, by combining this with the bound on ℓ obtained above, the algorithm is sure to expand an optimal node at $h(n)$ when this is smaller than $h_{\max}(n)$ – or, when $h(n) > h_{\max}(n)$, it expands an optimal node at $h_{\max}(n)$. Thus finally it expands some optimal node $i^*_{\underline{h}}$ at $\underline{h}$, and therefore the solution returned by SOPC satisfies:

$$v(i^*) \geq v(i^*_{\underline{h}}) \geq v^* - \delta_{\underline{h}}$$

i.e. it is $\delta_{\underline{h}}$-optimal.

Note that here we streamlined the proof of the SOO depth bound from [18] so as to take advantage of the fact that, unlike SOO, SOPC always expands a full path down to $h_{\max}(n)$. □

Proof of Theorem 11: Consider first $m > 1$. Since $h(n)$ is the smallest for which (8) holds, we have:

$$n > CMh^2_{\max}(n) \sum_{h'=0}^{h(n)-1} m^{h'} = CMh^2_{\max}(n) \frac{m^{h(n)} - 1}{m - 1}$$

Therefore:

$$h(n) < \frac{1}{\log m} \log \big[\frac{n(m-1)}{CMh^2_{\max}(n)} + 1\big] < c_3 \log n^{1-2\varepsilon}$$

for some constant $c_3 > 0$, where $h_{\max}(n) = n^\varepsilon$ was used. Thus $h(n)$ is logarithmic in n and, for large n, smaller than $h_{\max}(n)$ since the latter is a power of n. Thus, for large n, $\underline{h} = h(n)$ in Lemma 10.

Similarly solving (8) for a *lower* bound on $h(n)$, we get:

$$\begin{aligned} h(n) &\geq \frac{1}{\log m} \log \big[\frac{n(m-1)}{CMh^2_{\max}(n)}\big] - 1 \\ &= \frac{1}{\log m} \left[\log n^{1-2\varepsilon} - \log \frac{eCM}{m-1}\right] \end{aligned}$$

and plugging this into $\delta_{h(n)}$:

$$\begin{aligned} \delta_{h(n)} &= \tilde{O}\left(\gamma\hat{}\sqrt{2\frac{\tau-1}{\tau^2 \log m}\left[\log n^{1-2\varepsilon} - \log \frac{eCM}{m-1}\right]}\right) \\ &= \tilde{O}\left(\gamma\hat{}\sqrt{\frac{2(\tau-1)(1-2\varepsilon)\log n}{\tau^2 \log m}}\right) \end{aligned}$$

where the elimination of the subtracted term holds due to n being large. By Lemma 10 this is also the near-optimality of the algorithm.

When $m = 1$, (8) becomes simply $CMh_{\max}(n)(h(n) + 1) \geq n$, leading via $h_{\max}(n) = n^{1/3}$ to $h(n) \geq \frac{\sqrt[3]{n}}{CM} - 1$. Therefore

$$\underline{h} = \min\left\{\frac{\sqrt[3]{n}}{CM} - 1, \sqrt[3]{n}\right\} \geq \sqrt[3]{n} \min\left\{\frac{1}{CM}, 1\right\} - 1$$

and finally:

$$\begin{aligned} \delta_{\underline{h}} &= \tilde{O}\left(\gamma\hat{}\sqrt{2\frac{\tau-1}{\tau^2}\left[\sqrt[3]{n} \min\left\{\frac{1}{CM}, 1\right\} - 1\right]}\right) \\ &= \tilde{O}\left(\gamma\hat{}\left(n^{1/6}\sqrt{2\tfrac{\tau-1}{\tau^2} \min\left\{\tfrac{1}{CM}, 1\right\}}\right)\right) \end{aligned}$$

which is the desired result. □

B Execution time of planning algorithms

Fig. B.1 shows the execution time of the planning algorithms evaluated in Sec. 5 of the main paper. These results confirm the expectation that the budget n is the most important factor: the execution time is very close to linear in n, with minor differences between the algorithms due to their varying overhead of e.g. searching the tree in different ways. SOPC is the fastest.

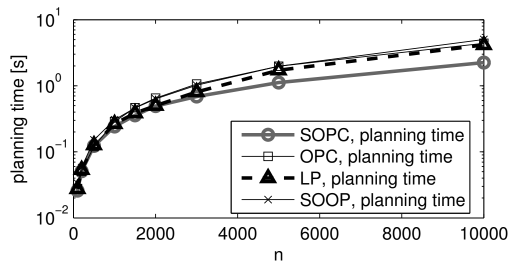

Fig. B.1. Runtime of one call to the planner, averaged over the 50 steps in the trajectory.

C Extension of SOPC to multiple actions

Although our analysis does not cover it, we briefly illustrate an extension of SOPC to 2 control actions. Here, each time step k is associated with 2 intervals rather than just 1. The idea is simple: boxes and steps k are selected for expansion using the same rules as in SOPC, but when a box must be expanded along step k, we split both intervals into M pieces, leading to $M^2 = 9$ child boxes. More refined rules could be given that avoid this direct exponential growth with the number of inputs.

To evaluate this extension, we use the rotational pendulum system from Sec. 5 of the main paper, but a second motor is added to the vertical joint of the pendulum. This motor has the same viscous damping and torque constant as the first one, but its model is simplified in that the torque is linearly related to the voltage $u_2 \in [-9, 9]$ V, rather than dynamically like for the first motor. The unnormalized reward function is $-x^\top \text{diag}[0.5, 0.05, 1, 0.05]x - u^\top \text{diag}[0.2, 0.2]u$. The C++ implementation of SOPC is used with a budget $n = 5 \cdot 10^5$ and $\varepsilon = 0.33$ in $h_{\max}$. The resulting trajectory from the pointing-down state is shown in Fig. C.1. The angles are approximately stabilized, although with worse performance and larger input adjustments than

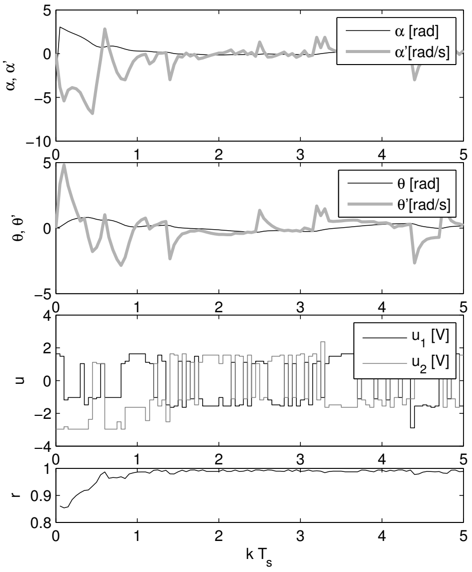

Fig. C.1. Two-input trajectory.

in the single-input case, since this simple extension has exponential complexity in the number of actions (and we increased the budget less than quadratically).