Reference State Trajectory Generation for Output Tracking with Constraints using Search Trees

Jonathan Eden[1], Ying Tan[1], Darwin Lau[2] and Denny Oetomo[1]

***Abstract*— This paper generates a reference state trajectory from a given reference output trajectory for redundant nonlinear affine dynamic systems subject to input, state and output constraints. Extending upon the expansive search tree (*EST*) methodology, a new probabilistic sampling method is proposed to address time dependence and output mappings. It is shown that the methodology is able to find a feasible solution given constraints, if such a solution exists. The use of the proposed methodology is illustrated by considering the end effector tracking of a multi-link cable-driven parallel robot.**

I. Introduction

Tracking a desired output trajectory is a common control task. When a dynamic system is nonlinear, to ensure that internal state is stable, a simple input-output linearisation technique is not enough [1]. For example, it is well-known that for a non-minimum phase system, tracking by input-output linearisation can lead to unstable internal state [1]. A reference state trajectory is therefore always needed to design an appropriate state feedback and stabilise internal state.

The inversion method has been widely used to generate a reference state trajectory from a given reference output. Different methods have been proposed from model-based inversion (linear or nonlinear) to dynamic inversion (see [2] and [3] and the reference therein). In the majority of the literature, constraints acting in the state, input and output spaces have however not been addressed. This work focuses on finding a feasible reference state from a given reference output considering that the state, input and output are all constrained into given sets. This is motivated from the application of multi-link cable-driven robots (*MCDRs*) [4].

Multi-link cable-driven robots (*MCDRs*) are a class of robot characterised by cable actuation. MCDRs are always constrained, since cables only provide force in tension. Additionally, MCDRs typically perform tasks that are defined in terms of an end effector (output) that possesses a lower number of degrees of freedom (*DoFs*) than the mechanism. As a result, output trajectory tracking for MCDRs is an example of generating a feasible reference state.

The determination of a feasible reference state can be reformulated as a searching problem. That is, for a given reference output, search through the domain of all state trajectories to find a corresponding feasible reference state. This type of model-based off-line searching is widely used in robotic path planning (see [5] and the references therein).

For robotic systems with high dimensional state spaces and constraints acting on the input, output and states, probabilistic sampling methods (*PSMs*) are a commonly used searching algorithm [5]. These methods incrementally generate a set of candidate paths (connecting the given initial state to the given end state) over a discretised time domain, whereby a probability distribution is used to determine which possible solution to grow at any given instant. The use of a probability distribution ensures that if a feasible trajectory exists, it will almost surely be identified as the number of increments tends towards infinite (*probabilistically complete*).

Two of the more common PSMs are rapidly-exploring random trees (RRT) [6] and expansive search trees (EST) [7]. Both methods construct a single tree from a common root using random sampling and a predefined tree growth procedure. Conventional PSMs consider the problem of point to point trajectory generation in the state space. As a result, the methods do not typically consider the time dependence or output mappings associated with output tracking. Independent extensions of RRT and EST have considered time based motions [8] or output point to point motions [9]–[12]. These methods typically neglect the effect of constraints and are formulated to only consider either the time dependence or the output mapping. The methods therefore prove ill suited to the generation of feasible reference state trajectories.

In this paper, the generation of a feasible reference state trajectory for the tracking of a predefined reference output is considered for the class of redundant affine systems. The work extends upon existing EST algorithms to find the feasible reference state. A tree based searching methodology is proposed to consider both the time and output dependent aspects of the output tracking problem. It is shown that the methodology is able to find a feasible solution, if such a solution exists. To illustrate the application of the methodology and the effect of the various tuning parameters, the end effector control of an example MCDR is considered.

II. Preliminaries and Problem Formulation

A. Notation

Denote the set of *real* and *non-negative* numbers as $\mathbb{R}$ and $\mathbb{R}_{\geq 0}$, respectively. The notation $\|\mathbf{x}\|$ denotes the Euclidean norm of $\mathbf{x}$. For a given matrix $A \in \mathbb{R}^{n\times m}$, where $m \geq n$, the pseudo-inverse $A^\dagger$ is defined such that $AA^\dagger = I_n$, where I_n denotes an n-dimensional square identity matrix. The null space of matrix A is the set $\mathcal{N}_A = \{\mathbf{x} \in \mathbb{R}^m : A\mathbf{x} = \mathbf{0}\}$. The matrix $N_A \in \mathbb{R}^{m\times(m-n)}$ denotes a matrix whose columns

[1]J. Eden, Y. Tan and D. Oetomo are with the Melbourne School of Engineering, the University of Melbourne, Australia {jpeden@student.,yingt@,doetomo@}unimelb.edu.au,

[2]D. Lau is with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong darwinlau@mae.cuhk.edu.hk

form a basis of the set $\mathcal{N}_A$. Let $\mathcal{A}$ denote a set containing a finite number of elements, the number of elements in $\mathcal{A}$ is given by the cardinality operator card($\mathcal{A}$). The notation $\mathbb{C}^n[t_0, t_1]$ denotes the set of all nth order differentiable and continuous functions defined over the domain $[t_0, t_1]$.

B. Problem Formulation

Consider a nonlinear dynamic system of the form

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}) + G(\mathbf{x})\mathbf{u}, \quad \mathbf{x}(t_0) = \mathbf{x}_0, \\ \mathbf{y} &= \mathbf{h}(\mathbf{x}), \end{aligned} \tag{1}$$

where $\mathbf{x}, \mathbf{x_0} \in \mathcal{X} \subset \mathbb{R}^n$ is the n-dimensional state and initial state, respectively, $\mathbf{u} \in \mathcal{U} \subset \mathbb{R}^m$ is the m-dimensional input, $\mathbf{y} \in \mathcal{Y} \subset \mathbb{R}^p$ is the p-dimensional output, $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^n$ is the state mapping, $G : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ is the input mapping and $\mathbf{h} : \mathbb{R}^n \to \mathbb{R}^p$ is the output mapping. Let the system (1) be both state-to-output redundant and input-to-state redundant such that $m > n > p$. The problem formulation is stated as

Problem Statement

Given: a dynamic system (1), a reference output trajectory $\mathbf{y}_{ref}(\cdot) \in \mathcal{Y} \cap \mathbb{C}^1[t_0, t_f]$ and a non-negative scalar $\varepsilon \geq \varepsilon^*$,

Determine: an input trajectory $\mathbf{u}(\cdot) \in \mathcal{U} \cap \mathbb{C}^0[t_0, t_f]$ such that the resulting state space trajectory satisfies $\mathbf{x}(\cdot) \in \mathcal{X} \cap \mathbb{C}^1[t_0, t_f]$ and the resulting output trajectory satisfies $\mathbf{y}(\cdot) \in \mathcal{Y} \cap \mathbb{C}^1[t_0, t_f]$, in addition to the output tracking constraint

$$\max_{t\in[t_0,t_f]} (\|\mathbf{y}(t) - \mathbf{y}_{ref}(t)\|) \leq \varepsilon. \tag{2}$$

Remark 1 (Tracking Parameter ε^*): The parameter ε^* represents the minimum tracking error that can feasibly be achieved by (1) for given $\mathcal{X}$, $\mathcal{U}$ and $\mathcal{Y}$. If $\varepsilon^* = 0$, then the reference output can be tracked without any error. To provide freedom in generating the reference state, it is possible to sacrifice the tracking performance ($\varepsilon > \varepsilon^*$) thereby also reducing the computational costs. ∘

Remark 2 (Polytopic Constraint Sets): For simplicity of presentation, the constraint sets $\mathcal{X}$, $\mathcal{U}$ and $\mathcal{Y}$ are assumed to be polytopic such that they can be represented as

$$A_x\mathbf{x} \leq \mathbf{b}_x, \ (3) \qquad A_u\mathbf{u} \leq \mathbf{b}_u, \ (4) \qquad A_y\mathbf{y} \leq \mathbf{b}_y, \ (5)$$

respectively. The constraint sets are therefore all convex but not necessarily compact. Similar analysis can be applied for other types of constraints with appropriate modification. ∘

Remark 3 (Local Control Methods): Due to the constraints and redundancy, reactive and/or finite horizon control may not ensure that the reference output remains feasible over the interval $[t_0, t_f]$. As a result, careful reference state generation over the complete interval $[t_0, t_f]$ is needed. ∘

Remark 4 (Sampling and Zero Order Holds): The formulation is in continuous-time. In practice, an appropriate discretisation is always used for the system trajectories of (1). Due to sampling, the continuity of the state and output trajectories may be lost. Filtering might therefore be needed to smooth the obtained feasible state trajectory. ∘

C. Expansive Search Trees

Tree based planners represent a class of search algorithms in which a tree is constructed to connect reference states to one another without specifying the time instant to reach the new state. One methodology of these planners that has been extensively used is that of expansive search trees (*ESTs*).

Let a tree leaf $\mathfrak{L}_i$ denote a sampled state $\mathbf{x}_i$*, the goal set $\mathcal{G}$ denote the set of all possible terminating states and the tree denote the set of tree leafs such that $\mathfrak{T} = \{\mathfrak{L}_i\}_{i=1}^{\kappa(\mathfrak{T})}$, where $\kappa(\mathfrak{T})$ is the current number of tree leafs. Algorithm 1 outlines the typical constrained EST algorithm. It can be seen that tree is initially populated by a single initial leaf $\mathfrak{L}_0 = \mathbf{x}_0$ and it continues to grow until it reaches the goal set $\mathcal{G}$. The process of growing the tree then consists of five steps: 1) The tree is sampled to obtain a new leaf to grow from $\mathfrak{L}_{sample}$ through the operation `sample_tree`. This is achieved through the construction and evaluation of a probability distribution $\mathfrak{p}[\cdot]$†. 2) A target leaf $\mathfrak{L}_{target}$ for the growth to move towards is determined by sampling the state space from the sampled leaf $\mathfrak{L}_{sample}$. 3) The set of possible inputs is sampled, through the operation `sample_input(...)`, to determine the input to be applied. This sampling allows for input constraints to be considered and ensures that the system dynamics are always satisfied. 4) A growth method, defined by the `grow_tree(...)` operation, is used to grow the tree from $\mathfrak{L}_{sample}$ using the input $\mathbf{u}$ which is held constant for a period of Δt seconds. 5) Finally, the candidate leaf is checked using the `is_valid(...)` operation to determine if the new leaf violates constraints on the valid set of states. If it does not violate the constraints, the new leaf is added to the tree.

Algorithm 1 Constrained EST Algorithm [7]

```
T ← {L_0};
while T ∩ G = ∅ do
    L_sample ← sample_tree(T);
    L_target ← sample_target(L_sample);
    u ← sample_input(L_sample, L_target);
    L_new ← grow_tree(L_sample, u, Δt);
    if (is_invalid(L_new)) then
        T ← T ∪ {L_new};
return T;
```

Remark 5 (Application of Algorithm 1 to the problem): The constrained EST algorithm (Algorithm 1), is conventionally applied to connect, without any time constraints, a given initial state $\mathbf{x}_0 \in \mathbb{R}^n$ with a final state $\mathbf{x}_f \in \mathbb{R}^n$ while satisfying the system dynamics and any other actuation constraints. In contrast, this paper considers the problem of tracking the reference trajectory $\mathbf{y}_{ref}(\cdot)$ given the initial state $\mathbf{x}_0 \in \mathbb{R}^n$ and constraints on the input, state, output system dynamics. The constrained EST formulation therefore proves incapable of satisfying the

*This notation is adopted for brevity. In practice, the leaf would also include the parent state $\mathbf{x}_p$ and the input required to go from $\mathbf{x}_p$ to $\mathbf{x}_i$.

†A tree probability distribution $\mathfrak{p}$ is a mapping $\mathfrak{p}[\cdot] : \mathfrak{T} \to [0, 1]$ where $\sum_{i=1}^{\kappa(\mathfrak{T})} (\mathfrak{p}[i]) = 1$ and $\kappa(\mathfrak{T})$ corresponds to the number of leafs in tree $\mathfrak{T}$.

problem due its lack of consideration for 1) the lower dimensional output space, 2) the ε tracking bound and 3) the time dependence of the tracking problem. ∘

III. EST Inspired Reference State Generation for Output Tracking

As noted in Remark 5, the constrained EST algorithm (Algorithm 1) does not provide a framework for identifying feasible state trajectories due to it not considering time dependence or the output space. Algorithm 2 presents a novel EST inspired methodology which can be used to solve the problem. It can be seen that the changes between Algorithm 2 and the classical constrained EST algorithm (Algorithm 1) reflect the need to consider both the time and the output.

Algorithm 2 EST Inspired Algorithm for Identifying Feasible State Trajectories

$\overline{\mathfrak{T}} \leftarrow \{\overline{\mathfrak{L}}_0\}$;
while $\overline{\mathfrak{T}} \cap \overline{\mathcal{G}} = \emptyset$ **do**
  $\overline{\mathfrak{L}}_{sample} \leftarrow$ `sample_tree`$(\overline{\mathfrak{T}})$;
  $\dot{\mathbf{y}}_{target} \leftarrow$ `determine_reference_velocity`$(\overline{\mathfrak{L}}_{sample})$;
  $\mathbf{n} \leftarrow$ `sample_null_space`$(\overline{\mathfrak{L}}_{sample}, \dot{\mathbf{y}}_{target})$;
  $\overline{\mathfrak{L}}_{new} \leftarrow$ `grow_tree`$(\overline{\mathfrak{L}}_{sample}, \boldsymbol{\gamma}, \Delta t)$;
  if (`is_invalid`$(\overline{\mathfrak{L}}_{new})$) **then**
    $\overline{\mathfrak{T}} \leftarrow \overline{\mathfrak{T}} \cup \{\overline{\mathfrak{L}}_{new}\}$;
return $\overline{\mathfrak{T}}$;

To incorporate the time component, Algorithm 2 makes use of the augmented tree $\overline{\mathfrak{T}} = \cup_i \{\overline{\mathfrak{L}}\}_i$. This new tree is composed of augmented leafs $\overline{\mathfrak{L}} = (\mathbf{x}, t)$ which include the current time information. As a result, the initial leaf is now given by $\overline{\mathfrak{L}}_0 = (\mathbf{x}_0, t_0)$ and the augmented goal set $\overline{\mathcal{G}}$ is defined to represent the desired end point of the reference state trajectory such that $\overline{\mathcal{G}}(t_f) = \{(\mathbf{x}, t) \in \mathbb{R}^{n+1} : \|\mathbf{y}_{ref}(t_f) - \mathbf{h}(\mathbf{x})\| \leq \varepsilon,\, t = t_f\}$.

To consider the output, Algorithm 2 modifies the tree growth methodology through the use of a reference output derivative $\dot{\mathbf{y}}_{target}$, determined using the operation `determine_reference_velocity`$(\overline{\mathfrak{L}}_{sample})$, as well as the replacement of the input with the new null space decision variable $\mathbf{n} \in \mathbb{R}^{m-p}$ which is randomly chosen using the `sample_null_space(...)` operation. Additional changes lie in the sampling (or `sample_tree(...)` operation) which must now consider the time information, as well as the method of tree growth which must now consider the set constraints (3)-(5), in addition to the output tracking constraint (2). A more detailed consideration of these two changes is provided in the subsequent subsections.

A. Selection of Tree of Elements

Algorithm 3 expands upon the `sample_tree` procedure of Algorithm 2. The key component of this process corresponds to the generation of the probability distribution $\mathfrak{p}[\cdot]$. Once this has been constructed, the sampling procedure is concluded by using the distribution to sample the tree.

Algorithm 3 `sample_tree` Algorithm

$\mathfrak{p}[\cdot] \leftarrow$ `generate_probability_distribution`$(\mathfrak{T})$;
$i \leftarrow$ `sample_probability_distribution`$(\mathfrak{p}[\cdot])$;
$\mathfrak{L}_{sample} \leftarrow \mathfrak{T}[i]$;
return $\mathfrak{L}_{sample}$;

In this work a hierarchical probability distribution (*HPD*) $\mathfrak{p}[\cdot]$ is proposed to separate the effect of the time and state leafs components on the total probability of the leaf being sampled. Let $\iota_t(\cdot) : \overline{\mathfrak{T}} \to \mathbb{R}$ denote the mapping that takes in a leaf and returns its time component, $\iota_{\mathbf{x}}(\cdot) : \overline{\mathfrak{T}} \to \mathbb{R}^n$ denote the mapping that takes in a leaf and returns its state component, $\tau(\overline{\mathfrak{T}})$ denote the number of different unique time steps currently in the tree such that $\tau(\overline{\mathfrak{T}}) = \frac{\max_{i \in \{0,\ldots,\kappa(\overline{\mathfrak{T}})-1\}} \left(\iota_t\left(\overline{\mathfrak{T}}[i]\right)\right) - t_0}{\Delta t} + 1$, where $\kappa(\mathfrak{T})$ denotes the number of tree leafs $\mathfrak{T}$ and let $\zeta(t_i, \overline{\mathfrak{T}})$ denote the number of leafs at time t_i such that $\zeta(t_i, \overline{\mathfrak{T}}) = \text{card}\left(\overline{\mathfrak{T}}_{t_i}\right)$, where $\overline{\mathfrak{T}}_{t_i} = \left\{\mathfrak{L} \in \overline{\mathfrak{T}} : \iota_t(\mathfrak{L}) = t_i\right\}$. The HPD is then assigned such that its elements are given by

$$\mathfrak{p}[i] = \mathfrak{a}^t\left[\iota_t\left(\overline{\mathfrak{T}}[i]\right)\right] \mathfrak{b}^{\mathbf{x}}_{\iota_t(\overline{\mathfrak{T}}[i])}\left[\mathfrak{o}^{\mathbf{x}}_{\iota_t(\overline{\mathfrak{T}}[i])}[\overline{\mathfrak{T}}[i]]\right], \tag{6}$$

where $\mathfrak{a}^t[\cdot] : \{0, \ldots, \tau(\overline{\mathfrak{T}}) - 1\}\Delta t \to (0, 1]$ is a monotonically increasing sequence of $\tau(\overline{\mathfrak{T}})$ elements such that $\sum_{i=0}^{\tau(\overline{\mathfrak{T}})-1} \mathfrak{a}[i] = 1$, each of the $\mathfrak{b}^{\mathbf{x}}_t[\cdot] : \{0, \ldots, \zeta(t, \overline{\mathfrak{T}}) - 1\} \to (0, 1]$ is a monotonically decreasing sequence of $\zeta(t, \overline{\mathfrak{T}})$ elements such that $\sum_{i=0}^{\zeta(t,\overline{\mathfrak{T}})-1} \mathfrak{b}_t[i] = 1$ and each of the $\mathfrak{o}^{\mathbf{x}}_t[\cdot] : \overline{\mathfrak{T}}_t \to \{0, \ldots, \zeta(t, \overline{\mathfrak{T}}) - 1\}$ is a ordering mapping which assigns to a leaf its position in the reordered set $\mathfrak{T}_t = \left\{\overline{\mathfrak{L}}_{\alpha_1}, \ldots, \overline{\mathfrak{L}}_{\alpha_{\zeta(t,\overline{\mathfrak{T}})}}\right\}$, which is ordered such that $\left\|\mathbf{y}_{ref}(t) - \mathbf{h}\left(\iota_{\mathbf{x}}\left(\overline{\mathfrak{L}}_{\alpha_1}\right)\right)\right\| \leq \ldots \leq \left\|\mathbf{y}_{ref}(t) - \mathbf{h}\left(\iota_{\mathbf{x}}\left(\overline{\mathfrak{L}}_{\alpha_{\zeta(t,\overline{\mathfrak{T}})}}\right)\right)\right\|$.

Figure 1 provides a pictorial representation of $\mathfrak{p}[\cdot]$. It can be seen that the hierarchical approach (6) allows for separate costs to be placed onto the time and state components.

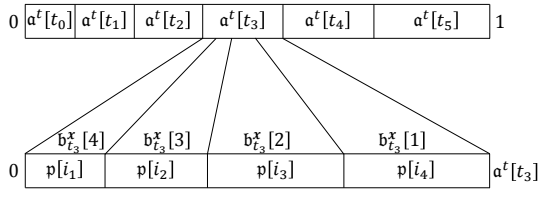


Fig. 1: Pictorial representation of the distribution $\mathfrak{p}[\cdot]$

Remark 6 (Dual layer interpretation): The HPD $\mathfrak{p}[\cdot]$ is effectively comprised of two layers. In the first layer, $\mathfrak{p}[\cdot]$ is broken up based upon the different time instances that are considered in the tree, where the total probability assigned to the time instant t_i is given by $\mathfrak{a}^t[t_i]$. The use of the monotonically increasing sequence $\mathfrak{a}^t[\cdot]$ ensures that leafs with a larger time step are more likely to be chosen to grow from. The particular selection of $\mathfrak{a}^t[\cdot]$ specifies the speed of the tree's forward growth. The choice of this

parameter provides a balance between convergence speed and the feasibility of the solution.

The second layer breaks up the probability assigned to each time step $\mathfrak{a}^t[t_i]$ into different probabilities for each of the leafs with time step t_i based upon their state. Through this layer, the bias of the tree towards the use of states with lower tracking error can be set. Here, the use of flatter monotonically decreasing sequence $\mathfrak{b}_t^{\mathbf{x}}[\cdot]$ results in all of the different leafs being considered as roughly equivalent. In contrast, the use of a sharply decreasing $\mathfrak{b}_t^{\mathbf{x}}[\cdot]$ results in growth emanating from the leaf with lower tracking error. ∘

B. Tree Growth

For EST algorithms, tree growth looks to extend the tree from the selected leaf $\mathfrak{L}_{sample}$ over a single time period while satisfying the constraints. Algorithm 2 therefore makes use of the altered decision variable $\mathbf{n} \in \mathbb{R}^{m-p}$, where this variable provides a coordinate for the affine space subset of the input space $\mathbb{R}^m$ which can satisfy both (1) and (2). To compute an appropriate null space, the following procedure is considered: First, the output mapping $\mathbf{h}(\cdot)$ is differentiated to derive a relationship between the feasible state derivative $\dot{\mathbf{x}}$ and the reference output derivative $\dot{\mathbf{y}}_{ref}$. Second the system dynamics is used to relate the feasible state derivative with the inputs that can produce those derivatives. The procedure presented focuses on the input constraint (4) as this constraint can be considered through the null space. The state and output constraints (3) and (5) are considered in Remark 8.

The output tracking relationship

$$\mathbf{y}_{ref}(t) = \mathbf{h}(\mathbf{x}(t)), \tag{7}$$

represents a desired relationship between the state and the reference output. Taking the first derivative of (7), it can therefore be seen that

$$\dot{\mathbf{y}}_{ref}(t) = H(\mathbf{x})\dot{\mathbf{x}}, \tag{8}$$

where $H : \mathbb{R}^n \to \mathbb{R}^{p\times n}$ is given by $H = \nabla_{\mathbf{x}}\mathbf{h}(\mathbf{x})$. During EST tree growth from $\mathfrak{L}_{sample}$, $\mathbf{x}$ is known. Given the output reference velocity $\dot{\mathbf{y}}_{ref}(t)$, the allowable state velocities can therefore be determined as a function of an output to state null vector $\mathbf{n}_1 \in \mathbb{R}^{n-p}$ by inverting (8) such that

$$\dot{\mathbf{x}} = H(\mathbf{x})^\dagger \dot{\mathbf{y}}_{ref}(t) + N_H(\mathbf{x})\mathbf{n}_1. \tag{9}$$

With knowledge of the allowable state velocities, the allowable inputs can be determined using the system dynamics (1). Substituting (9) into (1) and making $\mathbf{u}$ the subject of the resulting equation, it can be seen that the allowable input can be expressed in terms of the output to input null space vector $\mathbf{n} = \left[\mathbf{n}_1^T, \mathbf{n}_2^T\right]^T \in \mathbb{R}^{m-p}$. This results in

$$\begin{aligned}\mathbf{u} &= G(\mathbf{x})^\dagger \left(H(\mathbf{x})^\dagger \dot{\mathbf{y}}_{ref}(t) + N_H(\mathbf{x})\mathbf{n}_1 - \mathbf{f}(\mathbf{x})\right) + N_G(\mathbf{x})\mathbf{n}_2,\\ &= \boldsymbol{\xi}(\mathbf{x},t) + \Gamma(\mathbf{x})\mathbf{n},\end{aligned} \tag{10}$$

where $\boldsymbol{\xi}(\mathbf{x},t) = G(\mathbf{x})^\dagger \left(H(\mathbf{x})^\dagger \dot{\mathbf{y}}_{ref}(t) - \mathbf{f}(\mathbf{x})\right)$ and $\Gamma(\mathbf{x}) = \begin{bmatrix} G(\mathbf{x})^\dagger N_H(\mathbf{x}) & N_G(\mathbf{x})\end{bmatrix}$.

The use of the null space vector $\mathbf{n}$ and output velocity $\dot{\mathbf{y}}_{ref}$ as the decision and reference variables, respectively, allows for the constraints (1) and (7) to be locally satisfied at every instance. Furthermore, using the allowable input relationship $A_u\mathbf{u} \leq \mathbf{b}_u$, the input constraint can be expressed in terms of $\mathbf{n}$ through the inequality

$$A_u\Gamma(\mathbf{x})\mathbf{n} \leq \mathbf{b}_{\mathbf{u}} - A_u\boldsymbol{\xi}(\mathbf{x},t). \tag{11}$$

Due to the nonlinear dynamics (1), local satisfaction of the constraint (7) will not result in exact tracking. The use of the null space vector may therefore lead to a violation of the tracking condition (2). This can be negated in Algorithm 2 by including condition (2) into the `is_valid(...)` operation. This approach is however reactive, where the tree growth will not consider tracking error until after the condition is violated. Tracking error compensation can also be incorporated into Algorithm 2 through the modification of the variable $\dot{\mathbf{y}}_{target}$. Using a proportional control based compensation strategy, the target output velocity is given by

$$\dot{\mathbf{y}}_{target} = \dot{\mathbf{y}}_{ref} + K\left(\mathbf{y}_{ref} - \mathbf{y}\right), \tag{12}$$

where $K \in \mathbb{R}^{p\times p}$ is a constant positive definite matrix.

Remark 7 (Impact of the Gain Matrix K): The gain matrix K allows for the simulation feedback to be used to compensate for the effects of discretisation and nonlinearity during the state trajectory generation. This methodology, however adds a tuning requirement where the tracking performance is influenced in a nonlinear fashion by K as a result of the nonlinearity of the system model. ∘

Remark 8 (Considering State and Output Constraints): The state and output constraints (3) and (5) can be considered within Algorithm 2 in two different manners: Firstly, through the use of discretisation, the constraints can be approximated as additional constraints on the null space. In this approach, the output-to-state velocity relationship (8), the system dynamics (1) and allowable input mapping (10) would be numerically integrated with the known state value and substituted into the constraints (3) and (5) to identify null space linear inequalities. Alternatively, the constraints can also be considered within the final validation step through the `is_valid(...)` operation. ∘

C. Discussion

The HPD elements (6) are assigned such that the leaf selection probability is always non-zero. This feature, when combined with the random null space vector selection, ensures that the search considers all possible joint space trajectories which satisfy the system dynamics (1) and output tracking constraint (2). Algorithm 2 is therefore always guaranteed to eventually find a feasible state trajectory after a sufficiently large number of leafs have been grown.

Algorithm 2 possesses three tuning parameters: the positive definite matrix $K \in \mathbb{R}^{p\times p}$, the monotonically increasing sequence $\mathfrak{a}^t[\cdot]$ and the monotonically decreasing sequences $\mathfrak{b}_t^{\mathbf{x}}[\cdot]$. The selection of K compensates for discretisation and is discussed further in Remark 7. In contrast, the sequences $\mathfrak{a}^t[\cdot]$ and $\mathfrak{b}_t^{\mathbf{x}}[\cdot]$ serve to bias the growth of the tree based upon the time information and state information, respectively.

Remark 9 (Tuning Parameter ε): $\varepsilon \geq \varepsilon^*$ represents a possible tuning parameter. The selection of this parameter, discussed in Remark 1, affects how easy the problem is to solve and therefore alters the computational performance. ∘

IV. Illustrative Example

A. Multi-Link Cable-Driven Parallel Robot System Model

This example considers the end effector tracking of the MCDR shown in Figure 2. This MCDR is characterised by cable actuation, where the cables only transmit force $\mathbf{f} \in \mathbb{R}^6$ when their force is above a minimum tension level and below a maximum safety tension such that $10N \leq f_i \leq 100N$ for all $i \in \{1, \ldots, 6\}$. The end effector pose $\mathbf{y}$ is 3 dimensional ($p = 3$) and the 4 total DoFs of the mechanism are described by the pose vector $\mathbf{q} \in \mathbb{R}^4$. All simulations are conducted using the CASPR cable robot software [13].

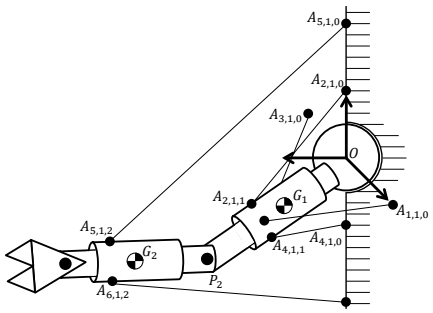


Fig. 2: The Example 4 DoF 2 link CDPR

For the BioMuscular Arm robot parameters[‡], the MCDR equation of motion and end effector mapping can be expressed using the methodology of [4] as

$$M(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) + \mathbf{G}(\mathbf{q}) = -L(\mathbf{q})^T\mathbf{f}, \tag{13}$$

$$\mathbf{y} = \mathbf{j}(\mathbf{q}), \tag{14}$$

$$10\mathbf{1}_4 \leq \mathbf{f} \leq 100\mathbf{1}_4 \tag{15}$$

where $M(\cdot) : \mathbb{R}^4 \to \mathbb{R}^{4\times4}$ is the symmetric positive definite inertia matrix, $\mathbf{C}(\cdot,\cdot) : \mathbb{R}^4 \times \mathbb{R}^4 \to \mathbb{R}^4$ is the Coriolis/centrifugal force vector, $\mathbf{G}(\cdot) : \mathbb{R}^4 \to \mathbb{R}^4$ is the gravitational vector, $L(\cdot) : \mathbb{R}^4 \to \mathbb{R}^{6\times4}$ is the cable Jacobian matrix and $\mathbf{j}(\cdot) : \mathbb{R}^4 \to \mathbb{R}^3$ is the end effector mapping.

Choosing the state representation $\mathbf{x} = \left[\mathbf{x}_1^T, \mathbf{x}_2^T\right]^T = \left[\mathbf{q}^T, \dot{\mathbf{q}}^T\right]^T$, the input $\mathbf{u} = \mathbf{f}$ and the output as the end effector pose, the system dynamics can be expressed in the form

$$\dot{\mathbf{x}} = \begin{bmatrix}\dot{\mathbf{x}}_1 \\ \dot{\mathbf{x}}_2\end{bmatrix} = \begin{bmatrix}\mathbf{x}_2 \\ \mathbf{f}(\mathbf{x}_1, \mathbf{x}_2) + G(\mathbf{x}_1)\mathbf{u}\end{bmatrix},$$
$$\mathbf{y} = \mathbf{h}(\mathbf{x}_1),$$
$$A_u\mathbf{u} \leq \mathbf{b}_u, \tag{16}$$

where $\mathbf{f}(\mathbf{x}_1, \mathbf{x}_2) = -M(\mathbf{x}_1)^{-1}\left(\mathbf{C}(\mathbf{x}_1, \mathbf{x}_2) + \mathbf{G}(\mathbf{x}_1)\right)$, $G(\mathbf{x}_1) = -M(\mathbf{x}_1)^{-1}L(\mathbf{x}_1)^T, \mathbf{h}(\mathbf{x}_1) = \mathbf{j}(\mathbf{x}_1), A_u = [-I_4\ I_4]^T$ and $\mathbf{b}_\mathbf{u} = \left[-10\left(\mathbf{1}_4^T\right)\ 100\left(\mathbf{1}_4^T\right)\right]^T$.

[‡]The mechanism parameters can be found at the repository `https://github.com/darwinlau/CASPR` in the BMArm subfolder

While the MCDR dynamics (16) does not satisfy $m > n$, it is noted that due to the double integrator system dynamics, the system can be completely controlled through its acceleration. Algorithm 2 is therefore applied by using $\ddot{\mathbf{y}}_{ref}$ in place of $\dot{\mathbf{y}}_{ref}$ and by double integrating the acceleration term of the MCDR dynamics. This means that the applied reference input must also satisfy the condition $\mathbf{y}_{ref}(\cdot) \in \mathbb{C}[t_0, t_f]$.

B. End Effector Tracking Example

Figure 3a depicts this example's reference end effector motion. This motion corresponds to a straight line motion of the end effector in the y_3 direction. Throughout this motion the other end effector DoFs are held constant.

Using the reference output trajectory (Figure 3a), with the sampling period $\Delta t = \frac{1}{150}$s and the controller (12), where $K = \text{diag}(200I_4, 2I_4)$, it was observed that Algorithm 2 could identify feasible state trajectories for $\varepsilon \geq 7 \times 10^{-4}$m. Figures 3b - 3d depict the solution input trajectory, pose component of the state trajectory and velocity component of the state trajectory, respectively, for one candidate solution in which ε was set to 5×10^{-3}m.

These trajectories were obtained using the sequences

$$\mathfrak{a}^t[t_i] = \frac{1-\mu}{1-\mu^{\tau(\overline{\mathfrak{T}})}}\mu^{\left(\tau(\overline{\mathfrak{T}}) - \frac{t_i}{\Delta t}\right)}, \tag{17}$$

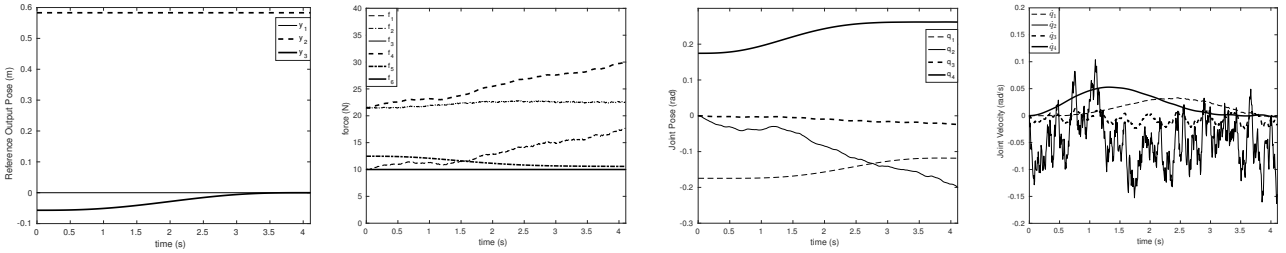
$$\mathfrak{b}_{t_i}^{\mathbf{x}}[j] = \frac{1-\eta}{1-\eta^{\zeta(t_i,\overline{\mathfrak{T}})}}\eta^j, \tag{18}$$

where $\mu = 0.95$ and $\eta = 0.9$, for all sampled time instances. From these results, it is observed that Algorithm 2 is able to identify solutions that satisfy the constraints and that the resulting trajectories possess small variations between sampling time such that they can be implemented in practice.

C. Tuning Parameter Variation

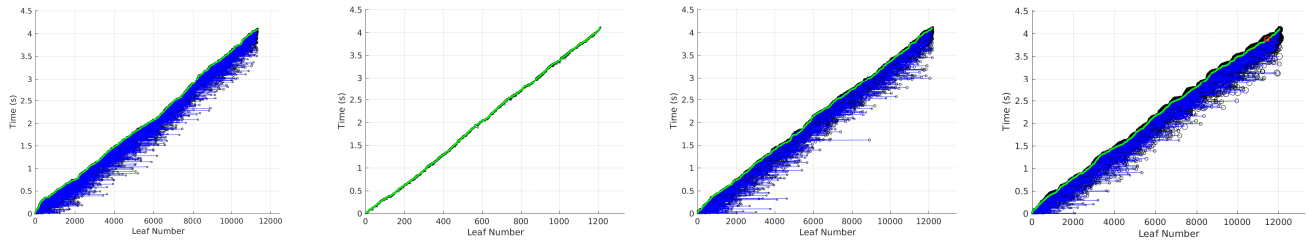
The controller gain K is very important to keep some robustness with respect to sampling. Due to space limitation, only the role of $\mathfrak{a}^t[\cdot]$ and $\mathfrak{b}_{t_i}^{\mathbf{x}}[\cdot]$ is discussed. Figure 4a provides a visualisation of the tree used in identifying the candidate trajectories of Figures 3b - 3d. This figure depicts the leaf number i against the leaf time iteration $\iota_t(\overline{\mathfrak{T}}[i])$, where the blue lines indicate the parent leaf relationships, the circle radii indicate the relative magnitudes of the tracking error with respect to the threshold ε and the green line indicate the solution branch used in Figures 3b - 3d. Using Figure 4a it can be observed that the tree considered 11332 leafs in identifying the solution. Furthermore, due to the relatively flat sequence $\mathfrak{a}^t[\cdot]$, it can also be observed that a wide range of leafs can be selected at any point in the tree growth. This is indicated in Figure 4a through the length of the blue lines. Despite this feature, the solution however primarily makes use of the first leaf identified at a given time instant t_i. This results from the relatively flat $\mathfrak{b}_t^{\mathbf{x}}[\cdot]$ sequences.

To investigate the effect of varying $\mathfrak{a}^t[\cdot]$, Figure 4b shows the tree growth using (17) with $\mu = 0.5$, while the sequences $\mathfrak{b}_t^{\mathbf{x}}[\cdot]$ are unchanged. From this figure, it can be seen that the sharper $\mathfrak{a}^t[\cdot]$ sequence has resulted in more direct tree growth that made use of only 1213 leafs. This however means that



(a) Reference End Effector Pose (b) Input Trajectory (c) Joint Pose Trajectory (d) Joint Velocity Trajectory

Fig. 3: Identified Solution Trajectory



(a) $(\mu, \eta) = (0.95, 0.9)$ (b) $(\mu, \eta) = (0.5, 0.9)$ (c) $(\mu, \eta) = (0.95, 0.1)$ (d) $\varepsilon = 0.001$m

Fig. 4: Tree Growth for Different Algorithm Parameters

there is approximately two leafs for every time instant, such that there is little exploration performed. This limits the tree's capability to avoid constraint violation.

In contrast to Figure 4b, Figure 4c considers the effect on the tree growth of varying the sequences $\mathfrak{b}_t^{\mathbf{x}}[\cdot]$ while keeping $\mathfrak{a}^t[\cdot]$ as in (17), where in this case (18) is used with $\eta = 0.1$. From this result, it is observed that the sequences $\mathfrak{b}_t^{\mathbf{x}}[\cdot]$ have little effect on the total number of leafs used in the candidate solution identification. Compared to Figure 4a, the effect of the $\mathfrak{b}_t^{\mathbf{x}}[\cdot]$ sequences is however observed in the lower number of horizontal lines. This is reflective of the sharper $\mathfrak{b}_t^{\mathbf{x}}[\cdot]$ sequences, in which tree growth is most likely to make use of the current best identified solution. This is also noted in the green final solution branch which compared to Figure 4a makes use of a larger number of leafs that do not correspond to the first leaf identified at a given time instant.

Finally, Figure 4d indicates the effect of shrinking ε. It is observed (through the red circles) that the growth has resulted in leafs that violate (2). As a result, of the probabilistic nature of the solver, a feasible candidate solution has however been identified through further sampling.

V. Conclusion

In this paper, the identification of a feasible reference state trajectory with which to track a given reference output for redundant nonlinear control affine dynamic systems was considered. An expansive search tree (*EST*) inspired algorithm was proposed, which could identify a feasible solution if one exists while considering constraints. The methodology was illustrated on an example problem of end effector tracking for an MCDR, in which the effect of the different algorithm tuning parameters could be observed. Future work will look to incorporate the optimisation of metrics into the algorithm and to apply the methodology onto hardware platforms.

Acknowledgements

The work was supported by the grants from the Early Career Scheme sponsored by the the Research Grants Council (Reference No. 24200516).

References

[1] H. K. Khalil, *Nonlinear systems*. Prentice-Hall, 2002.

[2] C. Byrnes and A. Isidori, "Output regulation for nonlinear systems: an overview," *International journal of robust and nonlinear control*, vol. 10, no. 5, pp. 323–337, 2000.

[3] A. V. Pavlov, N. van de Wouw, and H. Nijmeijer, *Uniform output regulation of nonlinear systems: a convergent dynamics approach*. Springer Science & Business Media, 2006.

[4] D. Lau, D. Oetomo, and S. K. Halgamuge, "Generalized modeling of multilink cable-driven manipulators with arbitrary routing using the cable-routing matrix," *IEEE Transactions on Robotics*, vol. 29, pp. 1102–1113, 2013.

[5] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[6] ——, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[7] D. Hsu, R. Kindel, J.-C. Latombe, and S. Rock, "Randomized kinodynamic motion planning with moving obstacles," *The International Journal of Robotics Research*, vol. 21, no. 3, pp. 233–255, 2002.

[8] A. Sintov and A. Shapiro, "Time-based rrt algorithm for rendezvous planning of two dynamic systems," in *Proc. IEEE Int. Conf. Rob. Autom.*, 2014, pp. 6745–6750.

[9] D. Bertram, J. Kuffner, R. Dillmann, and T. Asfour, "An integrated approach to inverse kinematics and path planning for redundant manipulators," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2006, pp. 1874–1879.

[10] M. V. Weghe, D. Ferguson, and S. S. Srinivasa, "Randomized path planning for redundant manipulators without inverse kinematics," in *IEEE-RAS Int. Conf. Humanoid Robot.*, 2007, pp. 477–482.

[11] A. Shkolnik and R. Tedrake, "Path planning in 1000+ dimensions using a task-space voronoi bias," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2009, pp. 2061–2067.

[12] M. Behnisch, R. Haschke, and M. Gienger, "Task space motion planning using reactive control," in *Proc. IEEE/RSJ Int. Conf. Intell. Robot. Syst.*, 2010, pp. 5934–5940.

[13] D. Lau, J. Eden, Y. Tan, and D. Oetomo, "Caspr: A comprehensive cable-robot analysis and simulation platform for the research of cable-driven parallel robots," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2016, pp. 3004–3011.