Discrete-time Contraction Constrained Nonlinear Model Predictive Control using Graph-based Geodesic Computation

Journal:	*AIChE Journal*
Manuscript ID	AIChE-22-24953
Wiley - Manuscript type:	Research Article
Date Submitted by the Author:	01-Apr-2022
Complete List of Authors:	Wei, Lai; University of New South Wales, School of Chemical Engineering McCloy, Ryan; University of New South Wales, School of Chemical Engineering Bao, Jie; University of New South Wales, School of Chemical Engineering Cranney, Jesse; Australian National University, College of Science
Keywords:	Stability design, Contraction theory, Graph theory, Discrete-time nonlinear systems, Nonlinear model predictive control

Discrete-time Contraction Constrained Nonlinear Model Predictive Control using Graph-based Geodesic Computation*

Lai Wei[1], Ryan McCloy[1], Jie Bao[1]†, Jesse Cranney[2]

[1] School of Chemical Engineering, University of New South Wales

UNSW, Sydney, NSW 2052, Australia

[2] Research School of Astronomy and Astrophysics,

The Australian National University, ACT, Australia

Abstract

Modern chemical processes need to operate around time-varying operating conditions to optimize plant economy, in response to dynamic supply chains (e.g., time-varying specifications of product and energy costs). As such, the process control system needs to handle a wide range of operating conditions whilst optimizing system performance and ensuring stability during transitions. This article presents a reference-flexible nonlinear model predictive control approach using contraction based constraints. Firstly, a contraction condition that ensures convergence to any feasible state trajectories or setpoints is constructed. This condition is then imposed as a constraint on the optimization problem for model predictive control with a general (typically economic) cost function, utilizing Riemannian weighted graphs and shortest path techniques. The result is a reference flexible and fast optimal controller that can trade-off between the rate of target trajectory convergence and economic benefit (away from the desired process objective). The proposed approach is illustrated by a simulation study on a CSTR control problem.

Keywords:Nonlinear model predictive control, stability design, contraction theory, discrete-time nonlinear systems, graph theory.

1 Introduction

Chemical processes are traditionally designed for and operated at certain steady-state operating conditions. Nowadays, supply chains are increasingly dynamic and the process industry needs to shift towards more agile, cost-effective and flexible process operations, in response to the fluctuations in market demand for products with different specifications, and the costs and supply of raw materials and energy. Real-time optimization (RTO) has become a common practice to improve industrial

*This work was partially supported by Australian Research Council Discovery Project DP210101978.

†Corresponding author: j.bao@unsw.edu.au.

process operations for optimal process economy in response to the dynamics of supply chains [1]. Typically, process control is conducted in hierarchical layers, each with varying response times and control objectives. Plant scheduling[2] (medium to long term planning) is combined with an RTO layer (based on slow process changes, e.g., variability in raw material compositions and product specifications) to determine economical operating targets. The lower-level control system then seeks to drive the process to these operation targets, providing short-term process performance. As such, the control system needs to be able to dynamically track time-varying setpoints determined by the RTO/scheduling layer, whilst optimizing system performance and ensuring stability during transitions [3].

When an RTO/scheduling layer is employed to generate feasible time-varying setpoints, it is naturally befitting to adopt an optimal controller, such as the widely popular Model Predictive Control (MPC) paradigm [4, 5, 6], to also optimize the transitions between feasible setpoints [7]. This becomes increasingly important when the transition period is long (e.g., power modulation of aluminum smelting cells to deal with the intermittency of renewable energy typically requires the operation setpoint to be changed once every 12 hours but it would take about 15 to 25 hours to reach a new steady state). One major obstacle to incorporating nonlinear MPC (NMPC) for process control is to ensure stability (both in terms of process safety and operating target tracking) subject to cost functions which reflect process economy [8]. Since the economic cost functions are generally not positive definite, many existing stability conditions in traditional MPC [9] cannot be applied. This has led to the Lyapunov-based NMPC designs [10], whereby the economic optimization problem is solved subject to an additional stability constraint, and closed-loop stability is explicitly ensured by offline Lyapunov based control design. However, as the stability condition is only valid for a specific equilibrium, the control algorithm for dynamic operating targets requires offline redesign (e.g., with a new Lyapunov function) whenever the setpoint is updated [8]. Due to this inflexibility with respect to reference changes, the Lyapunov-based NMPC approach is impractical when driven by an RTO/scheduling layer.

Since the RTO/scheduling layer will generate time-varying *a priori* unknown setpoints, stability guarantees for optimal control require a condition that is reference-independent, e.g., those based on incremental stability [11, 12]. As a consequence, an effective NMPC requires an incremental stability constraint or equivalent (analogously to the Lyapunov-based NMPC approach [8]). Introduced by Lohmiller and Slotine [13], contraction theory facilitates stability analysis and control of nonlinear systems with respect to arbitrary, time-varying (feasible) references without redesigning the control algorithm [14, 15, 16]. One useful feature of contraction theory is that it can be used to analyze the incremental stability of nonlinear systems and simultaneously synthesize a controller that ensures offset free tracking of feasible target trajectories using control contraction metrics (or CCMs [14]). This has motivated increased interest for contraction-based NMPCs [17, 18], offering significant flexibility over Lyapunov-based alternatives, allowing tracking time-varying setpoints without redesigning the control algorithm.

Despite their advantages, the existing continuous-time contraction-based NMPC approaches [17, 18] suffer from computational issues, e.g., due to nested optimization during the prediction step, and

also introduce conservatisms due to sampling. As a result, it is often impractical (if not impossible) to impose a contraction constraint on model predictions beyond the first step, reducing suitability for real-time control of industrial processes. The authors' preliminary work on discrete-time contraction analysis and controller synthesis [19, 20, 21] removes the need for Lipschitz continuity conditions to handle sampling issues and thus allows for more practical implementation of contraction-based (digital) control. Moreover, in general, real-time contraction-based control requires the fast online computation of *geodesics* (i.e., the shortest path from the current state to the operating targets), which involves solving a nonlinear optimization problem at each time-step (analogously to NMPC) in the absence of structural exploits [22]. This further motivates the employment of discrete-time contraction-based methods (permitting a computation window) in parallel to the development of efficient algorithms.

Contraction theory-based control design requires computation of the geodesics and geodesic distances on curved surfaces, which is unavoidable when considering the highly nonlinear industrial processes (directly impacting the state manifold curvature). This can be a challenging task, compared to Euclidean distance computation. Driven by the broad range of applications across digital geometry processing, computer graphics and imaging, a wide variety of sophisticated geodesic computational methods have been developed in recent literature [23]. Of particular interest, graph-based geodesic computational methods provide an approximated path solution (through path discretization and numerical search) and permit a trade-off in accuracy for speed. Graph-based approaches are a befitting choice for real-time nonlinear control due to their implicit numerical stability and efficient online solution, which is achieved by transforming the online optimization problem into an online numerical search problem using pre-computed path information stored offline. This speed-up is naturally also advantageous when considering the already computationally heavy burden of contraction-constrained NMPC.

We propose a novel discrete-time contraction constrained NMPC in this work. By exploiting the discrete-time structure of a contraction-based stability constraint, significant computational burdens are relaxed by lifting restrictions on geodesic computations within the MPC optimization problem, enabling the practical and scalable implementation of a stability constraint for the full prediction horizon. Inspired by the benefits posed in the 2-dimensional surface examples[24], a novel Riemannian graph-based method is developed for geodesic calculation by solving a shortest path problem, tailored for contraction-based control, greatly improving practical feasibility with respect to existing contraction-based control methods (including non-MPC approaches). Analysis of the proposed method is also provided, including quantification of approximation error and mesh configuration, with additional extensions to further trade-off between accuracy and speed. The result is a practical discrete-time contraction constrained MPC, capable of optimally stabilizing processes in real-time to time-varying operational targets generated by an RTO layer.

The remainder of the paper is structured as follows. Section 2 formulates the overall problem and approach. Section 3 reviews discrete-time contraction theory. Section 4 develops the contraction constrained NMPC and Section 5 constructs the graph-based geodesic approach. Section 6 analyses

and discusses the proposed method. Section 7 presents a simulation study with conclusions drawn in Section 8.

Notations. Denote by $f_k = f(x_k)$ for any function f, $\mathbb{Z}$ represents the set of all integers, $\mathbb{Z}^+$ represents the set of positive integers, $\mathbb{R}$ represents set of real numbers.

2 Problem Formulation and Approach

2.1 Control Problem Formulation

We consider a discrete-time nonlinear system in control affine form as,

$$x_{k+1} = f(x_k) + g(x_k)u_k, \tag{1}$$

where $x_k \in \mathcal{X} \subseteq \mathbb{R}^n$ and $u_k \in \mathcal{U} \subseteq \mathbb{R}^m$ are the state and control vectors, respectively, $\mathcal{X}$ and $\mathcal{U}$ represent the restricted (or permissible) state and control spaces, and, f and g are smooth functions. We define a feasible triplet time-varying target sequence $(x_k^*, u_k^*, x_{k+1}^*)$ for system (1), i.e., satisfying

$$x_{k+1}^* = f(x_k^*) + g(x_k^*)u_k^*, \tag{2}$$

and consider the discrete-time control input, $u_k = u_k(x_k, x_k^*, u_k^*)$. The process operation targets, i.e., x^*, are often determined and updated by the RTO in response to variations in raw materials, product specifications, and market demand to optimize the plant economy at these steady-state target operation conditions.

The NMPC tracks the time-varying setpoints, i.e., $x_k \to x_k^*$ and is also often required to optimize process economy during transitions from one setpoint to another, via the minimization of an economic stage cost function, $\ell(k+i)$, i.e., a NMPC solves the following optimization problem,

$$\begin{aligned} &\min_{\hat{u}} \sum_{i=0}^{N_\ell} \ell(\hat{x}_i, \hat{u}_i), \\ &\text{s.t.} \quad \hat{x}_0 = x_k, \quad \hat{x}_{i+1} = f(\hat{x}_i) + g(\hat{x}_i)\hat{u}_i, \\ &\qquad \hat{u}_i \in \mathcal{U}, \quad \hat{x}_i \in \mathcal{X}, \end{aligned} \tag{3}$$

where N_ℓ is the prediction horizon, $\hat{x}_i$ and $\hat{u}_i$ are the respective i-th step state and control predictions and x_k is the current state measurement. Assuming feasibility of the optimization problem (3), i.e., the optimal input trajectory, $\hat{\mathbf{u}}^{opt} = (\hat{u}_0^{opt}, \hat{u}_1^{opt}, \cdots, \hat{u}_{N_\ell-1}^{opt}) \in \mathcal{U}^{N_\ell}$, can be computed satisfying (3). The NMPC is then implemented in a receding horizon fashion, by applying the first control action $\hat{u}_0^{opt}$ to system (1) until the next time instant, $k+1$, i.e.,

$$u(k) = \hat{u}_0^{opt} : [k, k+1). \tag{4}$$

Clearly, a NMPC requires a contraction based constraint to ensure reference-independent stability.

The generalized control objectives can be summarized as follows:

P1 To maintain reference-independent stability, i.e., $x_k \to x_k^*$, as $k \to \infty$, where x^* is a time-varying (externally updated) operation target;

P2 To minimize the stage cost $\ell(x, u)$ over a certain prediction horizon, N_ℓ, subject to constraints on the state, $x \in \mathcal{X}$, and control input, $u \in \mathcal{U}$.

2.2 Approach

To realize the control objectives in Section 2.1, we propose a novel NMPC with stability constraint, constructed via contraction theory, to ensure both reference convergency and optimal process economy. Under the contraction theory framework, stability is determined using the shortest path or *geodesic* between the current state and operation target. To improve the real-time amenability of the approach, the geodesic computation problem is addressed via a new fast graph-based method. The generalized control approach can then be summarized as:

i Formulate a contraction condition as a reference-independent stability constraint;

ii Transform the geodesic calculation from an optimization problem to a graph-based search problem;

iii Implement a discrete-time contraction constrained NMPC using graph-based geodesics.

The resulting discrete-time contraction constrained NMPC is capable of optimally stabilizing processes to time-varying operational targets generated by an RTO layer. In addition, the proposed approach offers a practical digital implementation through scalable and efficient computational methods.

3 Contraction Theory Overview

Since the process operation target is time varying, the control design requires rigorous stability conditions, e.g. incremental stability [11]. Contraction theory [13, 14] provides a differential framework that can be used to assess incremental stability properties, based on *differential dynamics*. As such, this section presents an overview of contraction theory.

Considering a discrete-time nonlinear system of the form (1), the corresponding differential dynamics are defined as

$$\delta_{x_{k+1}} = A\delta_{x_k} + B\delta_{u_k}, \tag{5}$$

where $A := \frac{\partial(f(x_k)+g(x_k)u_k)}{\partial x_k}$ and $B := \frac{\partial(f(x_k)+g(x_k)u_k)}{\partial u_k}$ are Jacobian matrices of functions f and g in (1) respectively, $\delta_{u_k} := \frac{\partial u_k}{\partial s}$ and $\delta_{x_k} := \frac{\partial x_k}{\partial s}$ are vectors in the tangent space $T_x\mathcal{U}$ at u_k and tangent space $T_x\mathcal{X}$ at x_k respectively (see Figure 1), which can be understood as the infinitesimal variations (parameterized by s) along any solution pair (x, u) of (1). Defining the state trajectories of (1) as $\check{x} := (x_0, \cdots, x_k, \cdots)$ for $k = 0, \cdots, \infty$, a group of state trajectories can then be parameterized by s and thus described as

$$\check{x}(s) = (x_1(s), x_2(s), \cdots, x_k(s), \cdots), \tag{6}$$

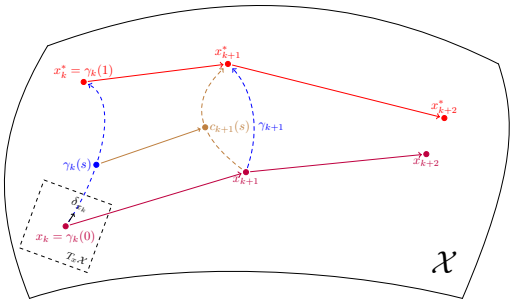


Figure 1: Illustration of s-parameterized trajectories in Riemannian space.

where $0 \leq s \leq 1$. For example, the trajectory $\check{x}$ is associated with $s = 0$, and $\check{x}^*$ is associated with $s = 1$, as shown in Figure 1.

Contraction theory studies the incremental stability properties of system (1), based on the analysis of differential dynamics (5). Practically, if (5) is exponentially stable, then the solution x_k is locally exponentially stable. Furthermore, if (5) is exponentially stable for all solutions x, then any pair of solutions will converge to each other. For rigorous analysis, we can utilize mathematical tools from Riemannian geometry[25]. Firstly, a Riemannian metric $M(x)$ is a matrix function $M : \mathbb{R}^n \to \mathbb{R}^n \times \mathbb{R}^n$ satisfying $M(x) = M^\top(x) > 0$ for all $x \in \mathbb{R}^n$. A metric $M(x)$ is said to be uniformly bounded if there exist $\alpha_2 \geq \alpha_1 > 0$ such that $\alpha_1 I \leq M(x) \leq \alpha_2 I$, $\forall x \in \mathbb{R}$.

Definition 3.1. A symmetric positive-definite function $V(x, \delta_x)$ is considered as the infinitesimal squared length distance, given by

$$V(x, \delta_x) = \delta_x^\top M(x) \delta_x. \tag{7}$$

Subsequently, System (1) is contracting with rate $0 < \beta < 1$, with respect to a uniformly bounded discrete-time control contraction metric (DCCM) $M(x)$, such that $V(x, \delta_x)$ in (7) satisfies

$$V_{k+1} - V_k \leq -\beta V_k < 0, \quad \forall x \in \mathcal{X}, \forall \delta_x \in T_x\mathcal{X}. \tag{8}$$

The function $V(x, \delta_x)$ can be understood as a differential Lyapunov function for the differential dynamics (5). Then, following some additional definitions from Riemannian geometry, incremental stability can be established.

Definition 3.2. For a smoothly varying, uniformly bounded metric $M(x)$, and a smooth curve $c(s) : [0, 1] \to \mathcal{X}$ (see, Fig. 1), connecting any pair of points, $x, x^* \in \mathcal{X}$ (such that $c(0) = x$ and $c(1) = x^*$), we define the Riemannian distance, $d(x, x^*)$, and energy, $E(x, x^*)$, as[25]

$$\begin{aligned} d_c(x, x^*) = d(c) &:= \int_0^1 \sqrt{\delta_{c(s)}^\top M(c(s)) \delta_{c(s)}} ds, \\ E(x, x^*) = E(c) &:= \int_0^1 \delta_{c(s)}^\top M(c(s)) \delta_{c(s)} ds, \end{aligned} \tag{9}$$

where $\delta_{c(s)} := \frac{\partial c(s)}{\partial s}$. Furthermore, there exists a bounded (not necessarily unique) minimum length

curve or geodesic, γ, connecting any two points, e.g. x and x^*, defined as

$$\gamma(s) := \arg\min_{c(s)} d(x, x^*), \tag{10}$$

where the geodesic distance is defined as $d_\gamma(x, x^*) := d(\gamma)$.

While γ is generally a curve, it can be a straight line joining x to x^* if the metric M is "flat" (i.e., not a function of x). The existence of such a curve can be guaranteed as M is uniformly bounded [14, Lemma 2]. Furthermore, the geodesic computation (10) can be converted into convex nonlinear optimization problem [22].

For brevity, contraction analysis via DCCMs involves searching jointly for a controller and the metric that describes the contraction properties of the resulting closed-loop system. It was shown that the existence of a contraction metric for a nonlinear system is sufficient for globally stabilizing every forward-complete solution of that system[14] . To see this, consider the generic differential state feedback law,

$$\delta_{u_k} = K(x_k)\delta_{x_k}. \tag{11}$$

The relationship between differential stability of (5) and the incremental stability of (1) given differential controller (11) is then given as follows.

Theorem 3.3. *[20] For a discrete-time nonlinear system* (1) *which satisfies*

$$(A_k + B_k K_k)^\top M_{k+1}(A_k + B_k K_k) - (1-\beta)M_k < 0, \tag{12}$$

the system is contracting with respect to a uniformly bounded, positive definite DCCM, $M(x_k)$, and it is exponentially incrementally stable, i.e., for any pair of solutions x_k, x_k^*,

$$|x_k - x_k^*| \leq Re^{-\lambda k \Delta_t}|x_0 - x_0^*|, \tag{13}$$

for some constants R and λ, where Δ_t denotes a discrete-time interval.

By integrating (11) along the geodesic, γ (10), one particular feasible tracking controller, can be defined as

$$u_k = u_k^* + \int_0^1 K\left(\gamma(s)\right) \frac{\partial \gamma(s)}{\partial s}\, ds. \tag{14}$$

Note that this particular formulation is reference-independent since the target trajectory variations do not require a structural redesign of the feedback controller and is naturally befitting to the flexible process operation paradigm. Moreover, the discrete-time control input, u_k (14), is a function with arguments (x_k, x_k^*, u_k^*) and hence the control action is computed from the current state and target trajectory.

If matrix functions A and B in the differential dynamics (5) can be approximated as polynomial functions of x and u, the DCCM can be determined offline is by using Sum of Squares (SoS) programming [20]. The contraction condition in (12) can be transformed into an equivalent and tractable

state-dependent LMI [19]

$$\Phi = \begin{bmatrix} W_{k+1} & A_k W_k + B_k L_k \\ (A_k W_k + B_k L_k)^\top & (1-\beta) W_k \end{bmatrix} > 0, \tag{15}$$

where $W_k := M_k^{-1}$, $W_{k+1} := M_{k+1}^{-1} = M^{-1}(f(x_k) + B(x_k)u_k)$, and $L_k := K_k W_k$ are the unknown polynomial matrix functions. The DCCM $M(x)$ can be synthesized by solving the following SOS programming problem for all $x_k \in \mathcal{X}$ and $u_k \in \mathcal{U}$:

$$\begin{aligned} &\min_{l_c, w_c, r} \ r \\ &s.t. \ \phi^\top \Phi \phi - rI \in \Sigma(x_k, u_k, \phi), \\ &r \geq 0, \end{aligned} \tag{16}$$

where w_c, l_c are polynomial coefficients of the matrix elements in $W_k(x)$, $W_{k+1}(x)$ and $L_k(x)$, respectively. Furthermore, ϕ is a vector of monomials, r is a small positive relaxation parameter and $\Sigma(x_k, u_k, \phi)$ is a non-negative polynomial sum of squares function of x_k, u_k and ϕ. Solution to (16) yields polynomial matrices $W(x), L(x)$, and hence the DCCM $M(x)$ with controller gain $K(x)$. Other DCCM synthesis methods were developed for more general differential dynamics (not limited to polynomial systems), e.g., a machine learning-based approach can be found in Wei et al.[21].

In summary, the resulting implication from contraction analysis is that the length of the minimum path (i.e., geodesic) between any two trajectories (e.g., the plant state, x, and desired state, x^*, trajectories), with respect to the metric $M(x)$, decreases with time. It is worth noting that for contraction-based control, the optimal or even *complete* trajectories (e.g., between the current state and target setpoint) are not needed, only the desired feasible setpoints are required. Suppose that a feasible complete set of reference trajectories was, in fact, available (typically a non-trivial task), then the same contraction based controller could be additionally used to drive the system to such references, and without structural redesign (simply update the geodesic information in (14)). The contraction-based control methods do not require redesigning the control algorithm as the reference changes (setpoint or otherwise), unlike the Lyapunov-based control designs.

4 Discrete-time Contraction Constrained NMPC

In this section, contraction conditions are translated into stability constraints which are then imposed on a control optimization problem. Since one of the main objectives is to achieve "optimal" operation (w.r.t. minimization of a stage cost, see the control objectives, **P1**, **P2**, of Section 2.1), we then wish to choose the "most optimal" controllers which also satisfy the criteria **P1**. Our approach is to define a set of controllers which satisfy **P1** and search amongst those for the most optimal. This process involves reformulating contraction conditions as a constraint in u and imposing it on a NMPC (23).

We note here that the NMPC (23) does not require pre-computation of the controller given in

(14) (and by extension the control reference u^*), i.e., control solutions to the optimization problem in (23) are not dependent on (14), but rather the existence of this control law which satisfies the contraction condition in (12) provides a certificate for problem feasibility subject to a contraction constraint (physical constraints are addressed in later sections). The use of NMPC in this form is additionally attractive when considering RTO-driven setpoints, for which, the control reference, u^*, is practically unavailable without additional computational burdens (via solution to an additional nonlinear optimization problem).

4.1 Discrete-time Contraction Constraints via DCCMs

As in the continuous-time contraction-based NMPC approaches[26, 27], the contraction constraints can be imposed on a NMPC to ensure reference flexible stability. By exploiting the structure of DCCMs, we propose here a discrete-time contraction constraint as a reference flexible and scalable condition, overcoming existing computational drawbacks and ensuring stability for the full prediction horizon.

Proposition 4.1. *For the system* (1)*, with differential dynamics* (5)*, provided a DCCM,* $M(x)$*, can be found satisfying* (12)*, the exponential convergence property of the system state,* x*, to the reference,* x^**, with rate* β*, can be expressed as a condition on the Riemannian energy of the geodesics at each time step, i.e.,*

$$E(\gamma_{i+1}) \leq (1-\beta)E(\gamma_i), \tag{17}$$

implies exponential incremental stability of x_i *to* x_i^**. This condition can be imposed as the contraction constraint*

$$r(\hat{x}_i, x_i^*, \hat{u}_i, \beta) := d\left(c_{i+1}\right) - (1-\beta)^{(i+1)/2}\, d\left(\gamma_0\right) \leq 0, \tag{18}$$

for any non-geodesic path, $c_i = c\left(x_i, x_i^*\right)$ *satisfying* $\gamma_{i+1} \leq c_{i+1} \leq \gamma_i \leq c_i$*.*

Proof. For (1) with (5), the existence of a DCCM, $M(x)$, satisfying (12), ensures it is exponentially incrementally stable (contracting w.r.t. $M(x)$) under Theorem 3.3. Then, consider (x_i, x_{i+1}) and (x_i^*, x_{i+1}^*) as two solution pairs of system (1). The shortest path connecting each of these solutions at prediction step i, denoted by the geodesic $\gamma_i(s)$ in (10), is defined with $\gamma_i(0) = x_i$ and $\gamma_i(1) = x_i^*$, and $c_{i+1}(s)$ denotes a path (not necessarily a geodesic) at the next time step, connecting $c_{i+1}(0) = x_{i+1}$ and $c_{i+1}(1) = x_{i+1}^*$. Hence, from (1), we have $c_{i+1}(0) = x_{i+1} = f(x_i) = f(\gamma_i(0))$ and $c_{i+1}(1) = x_{i+1}^* = f(x_i^*) = f(\gamma_i(1))$ as shown in Figure 1, or, without loss of generality,

$$c_{i+1}(s) = f(\gamma_i(s)). \tag{19}$$

From (9), (10) and the Hopf-Rinow Theorem, (e.g., in Manchester and Slotine[14]), we have

$$E(\gamma) = d(\gamma)^2 \leq d(c)^2 \leq E(c), \tag{20}$$

which, given (12) and (19), yields the decreasing Riemannian energy condition in (17), i.e.,

$$E(\gamma_{i+1}) \leq \int_0^1 \frac{\partial c_{i+1}(s)}{\partial s}^\top M(c_{i+1}(s)) \frac{\partial c_{i+1}(s)}{\partial s} ds \leq \int_0^1 (1-\beta) \frac{\partial \gamma_i(s)}{\partial s}^\top M(\gamma_i(s)) \frac{\partial \gamma_i(s)}{\partial s} \leq (1-\beta) E(\gamma_i). \tag{21}$$

Then, consider a non-geodesic path, $c_i = c\,(\hat{x}_i, x_i^*)$, defined as

$$d(\gamma_{i+1}) \leq d(c_{i+1}) \leq d(\gamma_i) \leq d(c_i), \tag{22}$$

which, from (17) and (9), gives (18). □

Remark 1. The information required for the computation of $d(\gamma)$ is available from the current state measurement and knowledge of the system model and reference dynamics. In addition, these conditions define a set (or the conditional property) of controllers which guarantee the system is contracting (under Theorem 3.3), whereby one particular feasible control solution is given by the state-feedback control structure proposed (14).

Whilst the contraction condition in (17) can alone be imposed as an incremental stability constraint on a NMPC (3), it requires solution of a nested optimization problem, e.g., at each i-th step in the prediction horizon solving (10) to obtain γ_i for $i = 0, \cdots, N_\ell$. To avoid the computational complexities involved with predicting geodesics, γ_i, for $i = 1, \cdots, N_\ell$, which are also dependent on the $\hat{u}_{i-1}$ solution, a common approach is to only consider the contraction constraint for a single step forward (requiring only the current step and previous step geodesic information), and removing the dependency of future control actions on compounding nested optimization problems[17, 18]. To ensure that the system is contracting for the full prediction horizon, the proposed alternative is to consider bounded future non-geodesic paths, c_i and exploit the discrete-time iterative structure. Relative to (17), the condition in (18), is significantly less computationally exhaustive since it does not require the geodesic computation at future steps in the prediction horizon yet still imposes the desired convergence property.

The contraction constraint of Proposition (18), has a reference-independent structure and is a nonlinear function affine in the control action, $\hat{u}$. To see this, consider that at each i-th prediction step, the Riemannian distance, $d(c_i)$, depends on the path $c(\hat{x}_i, x_i^*)$, which is generated iteratively using the model $\hat{x}_{i+1} = f(\hat{x}_i) + g(\hat{x}_i)\hat{u}_i$ and the reference trajectory x_i^* (obtained via RTO), which depend on the current state measurement, x_0, and predicted control sequence $\hat{u} = (\hat{u}_0, \hat{u}_1, \cdots, \hat{u}_{N_\ell - 1})$. In the following section, details for imposing this constraint on a NMPC will be provided.

4.2 Discrete-time Contraction Constrained NMPC

To optimize system economy, via minimization of a cost function, a NMPC which also ensures exponential incremental stability (between the state x and target x^* trajectories) solves the optimization

problem

$$\begin{aligned} &\min_{\hat{u}} \sum_{i=0}^{N_\ell} \ell(\hat{x}_i, \hat{u}_i), \\ \text{s.t.} \quad &\hat{x}_0 = x_k, \quad \hat{x}_{i+1} = f(\hat{x}_i) + g(\hat{x}_i)\hat{u}_i, \\ &\hat{u}_i \in \mathcal{U}, \quad \hat{x}_i \in \mathcal{X}, \\ &r(\hat{x}_i, x_i^*, \hat{u}_i, \beta) \leq 0, \end{aligned} \tag{23}$$

where ℓ denotes an economic cost function, x_k^* the desired target state at time step k and r is the contraction constraint in (18). The contraction-based stability constraint, r (18), can be imposed as part of the optimization problem (23) by:

1. Initialize the optimization by updating the current step metric $M(\hat{x}_k)$, and computing the geodesic $\gamma_0 = \gamma(\hat{x}_0, x_0^*)$ and corresponding Riemannian distance $d(\gamma(\hat{x}_0, x_k^*))$ using the current step state measurement $\hat{x}_0 = x_k$ and setting $i = 0$.
2. Choose a control input $\hat{u}_i$.
3. Compute the next step metric $M(\hat{x}_{i+1})$, path $c(\hat{x}_{i+1}, x_{i+1}^*)$ and corresponding Riemannian distance $d(c(\hat{x}_{k+1}, x_{k+1}))$ using the state update $\hat{x}_{i+1} = f(\hat{x}_i) + g(\hat{x}_i)\hat{u}_i$.
4. Evaluate the condition in (18) to see if the chosen $\hat{u}_i$ is a contracting control input. If (18) is satisfied update $i = i + 1 \leq N_\ell$, and repeat from step 2.

Note that the target trajectory information is captured by (18) and there is no introduced conservatism in the solutions $\hat{u}_i$. Moreover, one such controller that satisfies (18) is the one given by (14), which guarantees exclusive feasibility of the contraction constraint. An additional consideration for the state and control (i.e., input) constraints (which would impose restrictions on each step above) will be addressed in the following section.

In step 3, the next step path $c_{i+1} = c(\hat{x}_{i+1}, x_{i+1}^*)$ can be determined by: (1) using the geodesic, i.e., $c_{i+1} = \gamma_{i+1}$; or (2) implementing the next-step path, c_{i+1}, as an optimization decision variable, i.e., by using $\min_{\hat{u}, c(\hat{u})}$ in (23), the MPC will additionally search for a path c_{i+1} (given $\hat{u}_i$ from step 2) which satisfies (18). In case (1), this is equivalent to using the complete contraction constraint, i.e., $r = d(\gamma_{i+1}) - (1-\beta)^{1/2} d(\gamma_i) \leq 0$ from (9),(17),(18); and for case (2), using the relaxed contraction constraint in (18). This a trade-off to achieve optimality (in the sense of convergency) at the cost of computational complexity, by using a geodesic at each prediction step. Both computational methods can be implemented in conjunction with the graph-based path approach proposed in following sections.

It is also important to note here that step 1 includes the calculation of the geodesic at each time step k (i.e., at each state measurement x_k), which means that the optimization problem (23) includes a second optimization problem (regardless of the choice for the next-step path c_{i+1} at step 3). One possible approach to reduce the complexity introduced by geodesic calculations, is to use a weighted objective function that embeds the geodesic calculation in the cost function. However, a weighted objective function approach results in an objective trade-off between the geodesic length and the

economical cost, which may lead to infeasible geodesic computations since there is no guarantee for path completeness. To solve this problem, we develop an alternative graph-based tool in the following section to efficiently calculate the geodesic and with user-defined precision. It will also be shown that this development permits the implementation of the non-relaxed contraction constraint in (17) (cf. (18)) for the full horizon, enhancing the geodesic prediction capabilities and hence practicality of general contraction-constrained NMPC approaches.

5 Graph-based State Constrained Geodesic Length Computation

For effective real-time contraction-based control, the section presents a graph-based geodesic computation method, whereby the geodesic optimization problem in continuous space (10) is transformed into a constrained search problem (see e.g., in do Carmo[25]) in discrete space for numerical solution. First, the standard optimization based geodesic calculation method will be introduced, followed by the development of a novel graph-based tool.

5.1 Geodesic Length Calculation via Optimization

A typical geodesic calculation method for use with contraction-based control [19, 21] involves solving a numerical optimization problem. From (9) and (10), we have the following expression for computing the geodesic length,

$$d_\gamma(x, x^*) = \min_c \int_0^1 \frac{\partial c(s)}{\partial s}^T M(c(s)) \frac{\partial c(s)}{\partial s} ds. \tag{24}$$

where (see Section 3) $c(s)$ is an s-parameterized smooth curve connecting x ($s = 0$) to x^* ($s = 1$). Since (24) is an infinite dimensional problem over all smooth curves, without explicit analytical solution, the problem must be discretised to be numerically solved. Note that the integral can be approximated by discrete summation provided the discrete steps are sufficiently small. As a result, the geodesic (24) can be numerically calculated by solving the following optimization problem,

$$\begin{aligned} d_{\bar{\gamma}}(x, x^*) = \min_{\Delta\tilde{x}_s \in \mathcal{X}} & \sum_{i=1}^{N_{\Delta s}} \Delta\tilde{x}_{s_i}^T M(\tilde{x}_i) \Delta\tilde{x}_{s_i} \Delta s_i \\ s.t. \quad & \tilde{x}_1 = x, \ \tilde{x}_{N_{\Delta s}} = x^*, \end{aligned} \tag{25}$$

where $\bar{\gamma}(x, x^*) \approx \gamma(x, x^*)$ represents the numerically approximated geodesic, x and x^* are the endpoints of the geodesic, $\tilde{x}_i$ represents i-th point on a discrete path in the state space, $\Delta\tilde{x}_{s_i} := \Delta\tilde{x}_i / \Delta s_i \approx \partial c(s)/\partial s$ can be interpreted as the displacement vector discretised with respect to the s parameter, $\Delta\tilde{x}_s := (\Delta\tilde{x}_{s_1}, \ldots, \Delta\tilde{x}_{s_{N_{\Delta s}}})$ is the discretised path joining x to x^* (i.e., discretization of c(s) in (24)), all Δs_i are small positive scalar values chosen such that $\sum_{i=1}^{N_{\Delta s}} \Delta s_i = 1$, $N_{\Delta s}$ is the chosen number of discretization steps (of s), $\tilde{x}_i = \sum_{j=1}^{i} \Delta\tilde{x}_{s_j} \Delta s_j + x$ represents the numerical state evaluation along the geodesic.

This geodesic calculation method is an optimization problem. In other words, the NMPC (optimization) problem, (23) will contain another (embedded) optimization problem if we use the geodesic calculation method (25). This will increase the computation burden significantly, which is not feasible for existing tools. To overcome this issue, a graph-based geodesic calculation method will be introduced.

5.2 Geodesic Length Calculation via Weighted Graphs

In subsequent sections, an alternative graph-based method is developed such that (24) is approximated as a search problem over the discretised state space to reduce the online computation burden, i.e., by solving a single-pair weighted shortest-path problem for a distance-weighted state-space graph. We begin with some brief definitions from graph theory.

Definition 5.1 (Weighted Graph)**.** A weighted graph G is an ordered pair (V, H) corresponding to the set of vertices, V, and the set of edges, H, along with a weight function $h : H \to \mathbb{R}$ specifying the weight of each edge.

Definition 5.2 (Shortest Path)**.** Let $G = (V, H)$ be a weighted graph with h as its weight function. Then a path between $v_1, v_p \in V$ is the sequence of vertices $(v_1, v_2, \cdots, v_p)$ where $\{v_i, v_{i+1}\} \in H$ for all $i = 1, \cdots, p-1$. The sequence is a shortest path when it minimizes the sum

$$\sum_{i=1}^{p-1} h(v_i, v_{i+1}). \tag{26}$$

The proposed graph-based geodesic approach involves the following main steps, with additional discussion and detail provided in following subsections.

Offline:

1. Describe a discrete state space mesh $\mathcal{M} \subseteq \mathcal{X}$, where the i-th state entry of $\mathcal{M}$ is denoted by x^i.
2. Determine the $n_{v_i} =: \deg(x^i)$ neighbouring state entries, x^j for $j = 1, \cdots, n_{v_i}$, for each state entry $x^i \in \mathcal{M}$.
3. Compute the geodesic length, $d_\gamma(x^i, x^j)$, from a state entry, x^i, to the $j = 1, \cdots, n_{v_i}$ neighbouring entries, for each $x^i \in \mathcal{M}$.
4. Construct an undirected edge-weighted graph, $G(V, H)$, where each n-dimensional vertex, v_i, denotes a state entry $x^i = (x_1, \cdots, x_n)$ on the mesh $\mathcal{M}$ such that $V = v_1, \cdots, v_\phi$; each neighbouring vertex $v_j = x^j$ forms a connecting edge with v_i for $i = 1, \cdots, n_{v_j}$; and the edge weights, $h_{i,j}$, are defined as the geodesic distance between connected vertices (neighbouring state entries), $h_{i,j} = d_\gamma(x^i, x^j)$, such that

$$H = \begin{bmatrix} h_{1,1} & h_{1,2} & \cdots & h_{1,\phi} \\ h_{2,1} & h_{2,2} & \cdots & h_{2,\phi} \\ \vdots & \vdots & \ddots & \vdots \\ h_{\phi,1} & h_{\phi,2} & \cdots & h_{\phi,\phi} \end{bmatrix}, \tag{27}$$

(a) Uniformly distributed mesh.

(b) Non-uniformly distributed mesh.

Figure 2: Mesh distribution.

where the weights for unconnected vertices (non-neighboring entries) are zero.

Online:

1. Solve a weighted shortest-path problem for the distance-weighted state-space graph $G(V, H)$, where the path $P = (v_1, \cdots, v_p)$ is the shortest path (i.e., graph geodesic) from $v_1 = x_k$ to $v_p = x_k^*$ and minimizes the sum $\sum_{i=1}^{p-1} h_{i,i+1}$ (i.e., geodesic distance).

5.2.1 Determining the State Space Mesh and Vertices

One method to construct the state space mesh, $\mathcal{M}$, is to discretize the constrained state space by hypercubes (n-dimensional 'cubes') with known edge widths defined using the Euclidean distance

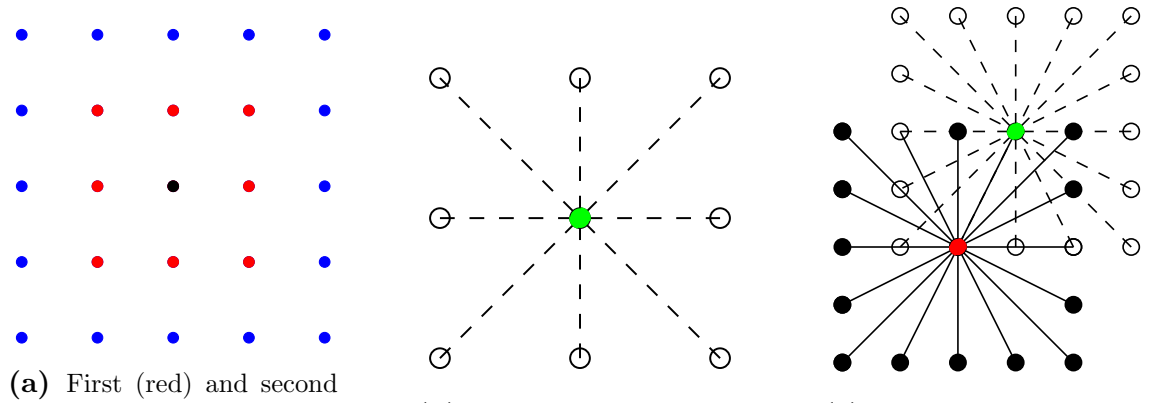
$$d_e(x^i, x^j) = ||x^i - x^j||_2, \tag{28}$$

where the i-th state entry of $\mathcal{M}$ is denoted by x^i, i.e., the state space mesh forms a set of vertices, each with a unique state value, defined as

$$V := \{v_1, v_2, \cdots, v_\phi\} := \mathcal{M} = \{x^1, x^2, \cdots, x^\phi\}. \tag{29}$$

For a uniformly distributed or linearly discretised state space mesh (see Figure 2a for a two dimensional example), the edge widths of each hypercube are fixed, i.e., $d_e(v_i, v_j) = d_e(x^i, x^j) = \bar{d}_e, \forall i \neq j = 1, \cdots, \phi$. Alternatively, the mesh can be constructed using a nonlinear discretization or non-uniform distribution (see the two dimensional example in Figure 2b), using hypercubes of varying edge widths. A non-uniform distribution permits finer detail in state regions of interest , e.g., areas with more nonlinearity, in the neighborhood of the target reference, or known areas of regular operation.

A finer mesh, formed with small Euclidean widths d_e and a relatively large number of state entries (i.e., large ϕ), offers increased accuracy, whilst also increasing computational burdens. Furthermore, a higher dimension of the dynamical system creates higher dimensional hypercubes. By imposing a nonlinear mesh, the trade-off between computation complexity and accuracy can be effectively managed. Additional considerations could also be made for alternative shapes to the hypercube,



(a) First (red) and second (blue) tier neighboring vertices.

(b) Nodal neighbors (first tier) and (connecting) edges.

(c) Connection between vertices (second tier).

Figure 3: Nodal neighbors and connections of vertices.

such as simplexes. Importantly, regardless of the chosen mesh distribution, path feasibility can be implicitly ensured by constructing the mesh using only permissible state values, i.e., $\mathcal{M} \subseteq \mathcal{X}$. Geodesic calculations strictly adhere to the state constraints by only considering vertices in the graph.

5.2.2 Choosing Nodal Neighbors and Vertex Connections

A nodal neighbor, v_j, of a vertex, v_i, is any vertex connected to v_i resulting in an undirected set of adjoining edges, E, defined as

$$E = \{\{v_i, v_j\} | v_i, v_j \in V, i \neq j, j = 1, \cdots, n_{v_i}, i = 1, \cdots, \phi\}, \tag{30}$$

whereby the degree of the corresponding vertex is defined as the number of its nodal neighbors, i.e., $n_{v_i} =: \deg(x^i)$, where E_i represents the set of nodal neighbors of vertices v_i. Under this construction, there is a great deal of freedom in selecting nodal neighbors for any vertex (state entry) on the state space mesh. Consider one specific vertex, $v_1 \in V$. By construction (see Section 5.2.1), there exists several layers of hypercubes about v_1, as shown in the two dimensional example of Figure 3a, where the first layer consists of the red vertices and the second layer of blue vertices. Suppose that vertices in the first (red) layer are selected as nodal neighbors of v_1, then, it will form the connected structure shown in Figure 3b. Then, by also including the vertices in the second (blue) layer as nodal neighbors provides additional directions from $x^1 = v_1$ to traverse the state space of $\mathcal{M}$. Thus, accuracy in the sense of direction can be improved by using nodal neighbors in an increasing number of layers (simultaneously with mesh refinement), see the structure in Figure 3c. However, as more nodal neighbors of v_1 are considered, to increase possible search directions, the computational complexity naturally increases. Consider for example, the memory storage and computational requirements associated when every other vertex, $v_i \neq v_1$, is considered, i.e., $n_{v_i} = \phi - 1$ for $i = 1, \cdots, \phi$. Thus, selection of nodal neighbors (together with the mesh construction) requires consideration for the available storage, required accuracy and resulting computational complexity.

5.2.3 Assigning Edge Weights

Given the vertex, $v_i \in V$, any non-neighboring vertex $v_\ell \in V$ is not connected to v_i (see Section 5.2.2) and hence the corresponding edge weight is zero. Then, for any neighboring vertices, $v_j \in V$ the edge weights are defined using geodesic length (10), i.e., the weights, H, are assigned as

$$H(i,j) = \begin{cases} h(v_i, v_j) = h(v_j, v_i) = d_{\tilde{\gamma}}(v_i, v_j) & \forall \{v_i, v_j\} \in E \\ h(v_i, v_j) = 0 & \forall \{v_i, v_j\} \notin E \end{cases}, \tag{31}$$

where $d_{\tilde{\gamma}} = d_{\tilde{\gamma}}(v_i, v_j)$ is computed as follows. If the Euclidean distance between nodal neighbors v_i to v_j is small, i.e., $d_e(v_i, v_j) \leq \epsilon$, it is reasonable to assume the neighborhood from v_j to v_i is approximately linear (metrically flat), and hence the geodesic length can be calculated as the average

$$d_{\tilde{\gamma}}(v_j, v_i) \approx \frac{1}{2} \left(M(v_i) + M(v_j) \right) d_e(v_i, v_j). \tag{32}$$

If the Euclidean distance between nodal neighbor v_j to v_i is not small, i.e., $d_e(v_i, v_j) \geq \epsilon$, the geodesic length can be calculated via the summation of the solved optimal argument (state constrained geodesic path) in (25), i.e.,

$$\begin{aligned} d_{\tilde{\gamma}}(v_j, v_i) = & \min_{\Delta \tilde{x}_s} \sum_{i=1}^{N} \Delta \tilde{x}_{s_i}^T M(\tilde{x}_i) \Delta \tilde{x}_{s_i} \Delta s_i \\ & s.t. \;\; \tilde{x}_1 = v_i, \; \tilde{x}_N = v_j, \; \Delta \tilde{x}_{s_i} \in \mathcal{X}. \end{aligned} \tag{33}$$

One particular advantage to computing the weights as in (32), (33), is that as the mesh gets increasingly fine, solutions to the optimization problem (25) invite increasing numerical errors associated with the finite precision of the computations, which can be alleviated via the introduction of ϵ.

5.2.4 Online Shortest Path Length Computation

For the undirected edge-weighted graph, $G(V, H)$, with vertices V (29) and edge weights H (31), the shortest-path (i.e., geodesic distance) can be found via a number of fast and efficient methods (see, e.g., the foundational approach by Dijkstra [28]). First we consider the scenario when the current state, x and target state x^* fall exactly on a a vertex. In this case, the shortest path, $P = (v_1, \cdots, v_p) \approx \gamma(x, x^*)$ (see Definition 5.2), for $v_1 = x_k$ to $v_p = x_k^*$, minimizes the cumulative sum of connecting edge weights, which forms the graph-approximated geodesic distance given by

$$d_{\tilde{\gamma}}(x, x^*) \approx d_{\tilde{\gamma}}(v_1, v_p) = \sum_{i=1}^{p-1} h(v_i, v_{i+1}). \tag{34}$$

For the alternative scenario, consider when either or both the current and target state do not fall exactly on a vertex. In this case, the start and end points each belong to a hypercube of n_x (for the current state x) and n_{x^*} (for the target state x^*) neighboring vertices denoted by $V_{n_x} = \{v_1, \cdots, v_{n_x}\}$ and $V_{n_{x^*}} = \{v_1, \cdots, v_{n_{x^*}}\}$ (note that the vertices have been reassigned indices for ease of notation and do not necessarily represent the first n_x or n_{x^*} vertices of V). See for example, Figure 4, which

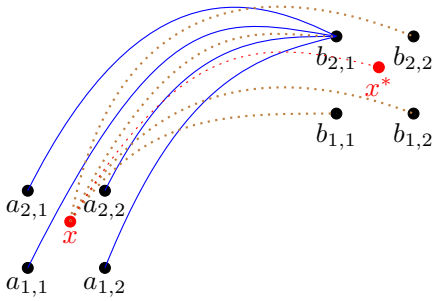


Figure 4: Geodesic calculation via convex combination using Method (ii) in (35).

shows a two dimensional example, where $n_x = 4$, $n_{x^*} = 4$ and $V_{n_x} = \{a_{1,1}, a_{1,2}, a_{2,1}, a_{2,2}\}$, $V_{n_{x^*}} = \{b_{1,1}, b_{1,2}, b_{2,1}, b_{2,2}\}$. We then offer two methods for computing the geodesic:

$$\begin{aligned} &\text{(i)} \qquad d_{\bar{\gamma}}(x, x^*) \approx d_{\bar{\gamma}}(v_x, x) + \sum_{i=1}^{p-1} h(v_i, v_{i+1}) + d_{\bar{\gamma}}(v_{x^*}, x^*) \\ &\text{(ii)} \qquad d_{\bar{\gamma}}(x, x^*) \approx \sum_{i=1}^{n_x} \sum_{j=1}^{n_{x^*}} \alpha_{i,j}(x, x^*) d_{\bar{\gamma}}(v_i, v_j) \end{aligned} \tag{35}$$

where $\alpha_{i,j}(x, x^*) \in [0, 1]$, $\sum \alpha_{i,j}(x, x^*) = 1$ and v_x and v_{x^*} are the closest vertices to x and x^* respectively, whereby the corresponding distances are calculated using

$$\begin{aligned} d_{\bar{\gamma}}(v_x, x) &= \min_{v_i \in V_{n_x}} \frac{1}{2} \left(M(v_i) + M(x) \right) d_e(v_i, x), \\ d_{\bar{\gamma}}(v_{x^*}, x^*) &= \min_{v_j \in V_{n_{x^*}}} \frac{1}{2} \left(M(v_j) + M(x^*) \right) d_e(v_j, x^*). \end{aligned} \tag{36}$$

In (35), method (i) exploits the idea that the spaces formed by V_{n_x} and $V_{n_{x^*}}$ are metrically flat and adds a small Riemannian length to the graph-based shortest path $P = (v_1, \cdots, v_p)$, from $v_1 = v_{n_x}$ to $v_p = v_{n_{x^*}}$, resulting in a complete path from x to x^*. In method (ii), a linear convex combination of all shortest paths (see Figure 4) connecting each of vertices of the hypercubes containing x and x^* (i.e., for each $v_i \in V_{n_x}$ and $v_j \in V_{n_{x^*}}$) is computed.

Method (i) provides the ability to calculate and store the path length together with the path explicitly, and the shortest path algorithm only needs to be executed once. In comparison, method (ii) provides a more accurate path length calculation by using the convex combination. This, however, comes at the cost of an upper limit of calculating 2^{2n} graph-based shortest paths and increases the complexity of computing the geodesic path. Consequently, when the mesh is sufficiently fine, both methods can provide accurate results if the mesh is not fine, resulting in a non metrically flat space formed by V_{n_x} and $V_{n_{x^*}}$, method (ii) provides improved accuracy relative to (i). Finally, we importantly note that both methods in (35) collapse to (34) when the current and target state fall precisely on a vertex.

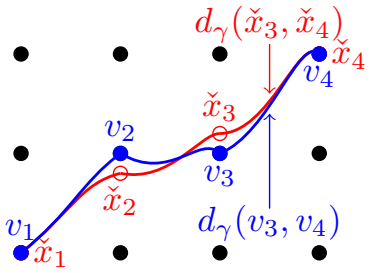


Figure 5: Actual geodesic path (red) vs approximated geodesic path (blue).

6 Analysis and Discussion

6.1 Analysis of the Graph-based Geodesic Length Computational Method

In this section, we will discuss the sufficiency of the proposed graph-based geodesic method. We begin by considering paths when the start and end points fall precisely on vertices in the graph, i.e., $v_1 = x, v_p = x^*$. The approximation error caused by breaking the geodesic into segments associated with the proposed graph-based geodesic length, $d_{\bar{\gamma}}$, can be quantified as,

$$d_{\bar{\gamma}}(v_1, v_p) - d_\gamma(v_1, v_p) = \sum_{i=1}^{p-1} \left(d_\gamma(v_i, v_{i+1}) - d_\gamma(\check{x}_i, \check{x}_{i+1})\right) = \sum_{i=1}^{p-1} \left(h(v_i, v_{i+1}) - d_\gamma(\check{x}_i, \check{x}_{i+1})\right), \tag{37}$$

where $\check{x}_i \in \check{P} \in \mathcal{X}$ are the state points (not necessarily on the mesh) along the geodesic which intersect with the first edge of each hypercube (see Figure 5). This expression (37) highlights several important features for the proposed method regarding the selection of nodal neighbors and the mesh spacing. Note, that provided each point $\check{x}_i$, falls precisely on a vertex, i.e., $\check{x} \in V$, then $P = \check{P}$ and the graph-based solution for geodesic length is exact.

One might then consider the effects of increasing the number of nodal neighbors. In Figure 5, we can see that each of the vertices in the graph-based solution are not equal to the actual geodesic at each vertex, i.e., $P \neq \check{P}$. Suppose then that instead of choosing the first-tier (see Figure 3a) of the neighboring vertices, we chose to connect all available nodes on the mesh. Then, the shortest graph-based path, $P = (v_1, v_2) = (x, x^*) = (\check{x}_1, \check{x}_2) = \check{P}$, produces the exact geodesic path length (using the corresponding edge weight). We can then derive an appropriate approximation bound using this concept, whereby

Increasing the number of nodal neighbors results in reducing the distance between the nodal neighbors and the geodesic path (intersected at the first edge of the corresponding hypercube). See for example Figure 7 where the introduced grey node, v_1, is closer to the blue geodesic. We then observe that as the number of nodal neighbors increases, i.e. increasing Y_i, the graph-based geodesic length approximation error is reduced. This, however, is limited by the available number of vertices, as a consequence of the fineness of the mesh, and hence leads to the following.

Lemma 6.1. *An upper bound on the error associated with the graph-based geodesic length is a function*

of mesh spacing and number of nodal neighbors, i.e.,

$$\sum_{i=1}^{p-1}(d_{\bar{\gamma}}(v_i,v_{i+1})-d_{\gamma}(\check{x}_i,\check{x}_{i+1}))\leq\sum_{i=1}^{p-1}\left((\overline{m}_i-\underline{m}_i)d_e(v_i,v_{i+1})+O\left(\overline{m}_i\frac{D_i\sqrt{n}}{Y_i}\right)\right), \tag{38}$$

where D_i denotes the width of the hypercube containing the set of v_i's nodal neighbors E_i (30), Y_i represents the tier of the nodal neighbors of v_i (see Figure 7), and the metric M is bounded by $\underline{m}_i I \leq M_i \leq \overline{m}_i I$ in that hypercube.

Proof. The geodesic distance between $(\check{x}_i,\check{x}_{i+1})$, with metric $\underline{m}_i I \leq M_i \leq \overline{m}_i I$, is bounded by

$$\underline{m}d_e(\check{x}_i,\check{x}_{i+1})\leq d_\gamma(\check{x}_i,\check{x}_{i+1})\leq\overline{m}_i d_e(\check{x}_i,\check{x}_{i+1}). \tag{39}$$

From definition of Riemannian space and the triangular inequality, we have

$$\begin{aligned}
&d_\gamma(\check{x}_i,\check{x}_{i+1})+d_\gamma(v_i,\check{x}_i)+d_\gamma(v_{i+1},\check{x}_{i+1})\geq d_\gamma(v_i,v_{i+1})\\
&d_\gamma(\check{x}_i,\check{x}_{i+1})\geq d_\gamma(v_i,v_{i+1})-d_\gamma(v_i,\check{x}_i)-d_\gamma(v_{i+1},\check{x}_{i+1})\\
&d_\gamma(\check{x}_i,\check{x}_{i+1})\geq \underline{m}_i d_e(v_i,v_{i+1})-\overline{m}_i d_e(v_i,\check{x}_i)-\overline{m}_i d_e(v_{i+1},\check{x}_{i+1})
\end{aligned} \tag{40}$$

We have $d(v_i,v_{i+1})\leq\overline{m}_i d_e(v_i,v_{i+1})$, thus

$$d(v_i,v_{i+1})-d(\check{x}_i,\check{x}_{i+1})\leq(\overline{m}_i-\underline{m}_i)d_e(v_i,v_{i+1})+\overline{m}_i d_e(v_i,\check{x}_i)+\overline{m}_i d_e(v_{i+1},\check{x}_{i+1}). \tag{41}$$

Furthermore, suppose the nodal hypercube width Euclidean length is d_e, the largest Euclidean distance of $d_e(v_i,\check{x}_i)$ or $d_e(v_{i+1},\check{x}_{i+1})$ is $\sqrt{n\frac{d_e^2}{(2Y_i)^2}}$ where n is the dimension of the system state x,

$$d(v_i,v_{i+1})-d(\check{x}_i,\check{x}_{i+1})\leq(\overline{m}_i-\underline{m}_i)d_e(v_i,v_{i+1})+2\overline{m}_i\sqrt{n\frac{D_i^2}{(2Y_i)^2}}, \tag{42}$$

which implies

$$d_{\bar{\gamma}}(v_i,v_{i+1})-d_\gamma(\check{x}_i,\check{x}_{i+1})\leq(\overline{m}_i-\underline{m}_i)d_e(v_i,v_{i+1})+O\left(\overline{m}_i\frac{D_i\sqrt{n}}{Y_i}\right). \tag{43}$$

By interpolating between the start and end points, (x,x^*), of the path (i.e., linearly connecting geodesic pieces inside the mesh) using (43), we have the inequality (38). □

Remark 2. By increasing the number of layers (and hence number) of nodal neighbors, Y_i, the term $O\left(\overline{m}_i\frac{D_i\sqrt{n}}{Y_i}\right)$ can be made arbitrarily small. In addition, by refining the mesh spacing, the area of each hypercube is smaller and hence $\overline{m}_i-\underline{m}_i$ is smaller since the variation of the metric M across that smaller surface is also smaller (i.e., the surface is more "flat"). Thus, by increasing the number of nodal neighbors and increasing the mesh precision, the right-hand side of (38) (graph-based path length approximation error) can be greatly reduced.

Importantly, the inequality in (38) shows that the approximation error of the graph-based geodesic

length is upper bounded by a function of mesh spacing, which reduces as the mesh becomes increasingly fine, i.e., as p increases, the flatness of each hypercube increases and hence the metric function approaches a point-wise (non-meshed) variation. Subsequently, when the start and end points of the geodesic do not fall exactly on the mesh, we have the following.

Corollary 6.2. *An upper bound on the error associated with the graph-based geodesic length, where x and x^* do not fall on vertices in the mesh, is defined as*

$$\begin{aligned}&\left\|\sum_{i=1}^{p-1}\left(d_{\tilde{\gamma}}(v_i,v_{i+1})-d_{\gamma}(\check{x}_i,\check{x}_{i+1})\right)-d_{\gamma}(x,\check{x}_1)-d_{\gamma}(x^*,\check{x}_p)\right\| \\ &\leq\left\|\sum_{i=1}^{p-1}\left((\overline{m}_i-\underline{m}_i)d_e(v_i,v_{i+1})+O\left(\overline{m}_i\frac{D_i\sqrt{n}}{Y_i}\right)\right)\right\|+\left\|\overline{m}_x\sqrt{n}D_x+\overline{m}_{x^*}\sqrt{n}D_{x^*}\right\|.\end{aligned} \tag{44}$$

where D_x, D_{x^*}, are the widths of the smallest hypercubes containing x and x^*, defined similarly to D_i in Lemma 6.1.

Proof. From (38) and the triangle inequality, it is easy to see,

$$\begin{aligned}&\left\|\sum_{i=1}^{p-1}\left(d_{\tilde{\gamma}}(v_i,v_{i+1})-d_{\gamma}(\check{x}_i,\check{x}_{i+1})\right)-d_{\gamma}(x,\check{x}_1)-d_{\gamma}(x^*,\check{x}_p)\right\| \\ &\leq\left\|\sum_{i=1}^{p-1}\left(d_{\tilde{\gamma}}(v_i,v_{i+1})-d_{\gamma}(\check{x}_i,\check{x}_{i+1})\right)\right\|+\left\|d_{\gamma}(x,\check{x}_1)+d_{\gamma}(x^*,\check{x}_p)\right\| \\ &\leq\left\|\sum_{i=1}^{p-1}\left((\overline{m}_i-\underline{m}_i)d_e(v_i,v_{i+1})+O\left(\overline{m}_i\frac{D_i\sqrt{n}}{Y_i}\right)\right)\right\|+\left\|\overline{m}_x\sqrt{n}D_x+\overline{m}_{x^*}\sqrt{n}D_{x^*}\right\|,\end{aligned} \tag{45}$$

where the maximum difference between x and $\check{x}_1$ is the longest line in the hypercube of dimension n defined as $\overline{m}_x\sqrt{n}D_x$, and similarly for x^*. □

Remark 3. As D_x and D_{x^*} become arbitrarily small (as the mesh spacing is reduced), the geodesic length approximation error from x and x^* to their closest vertices also tends to zero.

In summary, for a chosen state-space mesh spacing and a specific number of nodal neighbors, the geodesic length can be found using a graph-based method with sufficient precision (at the cost of memory storage and computational burden). Moreover, as the precision increases, the graph-based geodesic length (see Section 5.2) approaches the optimization value and hence the true geodesic length (as the discretization step size of s reduces – see Section 5.1).

6.2 Obtaining Permissible Geodesic Paths

The geodesic distance represents the value of distance between two points, however, the geodesic path contains information of the exact position of each point along the geodesic. From (9) and (10), a geodesic path is defined as (i.e., the argument of (24)),

$$\gamma(x,x^*)=\arg\min_c\int_0^1\frac{\partial c(s)}{\partial s}^T M(c(s))\frac{\partial c(s)}{\partial s}ds. \tag{46}$$

A numerically calculated geodesic (see Section 5.1) can then obtained by storing the argument of the solution to the numerical optimization problem in (25), i.e.,

$$\begin{aligned} \bar{\gamma}(x, x^*) = \underset{\Delta\tilde{x}_s \in \mathcal{X}}{\arg\min} \sum_{i=1}^{N} \Delta\tilde{x}_{s_i}^T M(\tilde{x}_i) \Delta\tilde{x}_{s_i} \Delta s_i \\ s.t. \quad \tilde{x}_1 = x, \ \tilde{x}_N = x^*, \end{aligned} \tag{47}$$

where $\bar{\gamma}(x, x^*) \approx \gamma(x, x^*)$ represents the numerically approximated geodesic, x and x^* are the end-points of the geodesic. Note that (47) is the discretization of (46) with $\Delta\tilde{x}_{s_i}$ and Δ_{s_i} as the discretizations of $\frac{\partial c(s)}{\partial s}$ and δ_s respectively, whereby the constraints in (25) ensure that the discretised path connecting the start, x, and end, x^*, state values align with the continuous integral from $s = 0$ to $s = 1$. Hence, as Δ_{s_i} approaches 0, i.e., for an infinitesimally small discretization step size, the approximated discrete summation in (25) converges to the smooth integral in (24). Moreover, imposing additional state constraints, $\Delta\tilde{x}_s \in \mathcal{X}$, also ensure that the connecting path only considers permissible state values (i.e., hard limits on the states). As already discussed, solving the (47) is an infinite dimensional problem which cannot guarantee a solution, the geodesic can be not accurate enough if the step size is not limited to some small value, furthermore, the complexity of this algorithm makes it extremely hard to implement in a NMPC scheme as the NMPC will need to solve an optimization problem (geodesic calculation) inside another optimization (NMPC).

To overcome this issue, the geodesic can be calculated using the graph based method in (35). If we use method (i) in (35), the resulting path is simply the straight line connecting the sequence of vertices and end points, $(x, v_1, \cdots, v_p, x^*)$. If we use the convex combination method (ii), we will have $N_x \times N_{x^*}$ multiple paths, the resulting geodesic path will be represented by the convex combination of all the paths as follows,

$$\bar{\gamma}(s) \approx \sum_{i=1}^{n_x} \sum_{j=1}^{n_{x^*}} \alpha_{i,j}(x, x^*) \bar{\gamma}_{i,j}(s), \tag{48}$$

where $\bar{\gamma}_{i,j}(s)$ represents the s-parameterized path, $(x, v_1^{i,j}, \cdots, v_p^{i,j}, x^*)$ where $v_i^{i,j}$ represents the (i, j) solution in method ii, such that $\bar{\gamma}_{i,j}(0) = x$ and $\bar{\gamma}_{i,j}(1) = x^*$.

Analogously to geodesic length, as the precision of the graph increases, the graph-based geodesic path approaches the optimization solution (47) and hence a true geodesic path (46), as the discretization step size of s reduces. Additional routines are also available for improving the geodesic path precision *a posteriori* (see, e.g., Newton-Raphson iterative methods [24]), which can assist with offsetting the computational complexity (associated with reducing the step size of s) and memory storage requirements (associated with increasing mesh refinement and nodal neighbor selection).

Using the calculated path, a particular control implementation (although not required here) is readily available from (14). We also note that by employing a graph-based geodesic computation method, the nested optimization issues associated with predicted geodesics, by imposing (17) on (3), is additionally overcome. Recall that the nested optimization problem at each i-th step in the prediction horizon requires solving (10) via (47) to obtain γ_i for $i = 0, \cdots, N_\ell$. Computing the geodesics using a

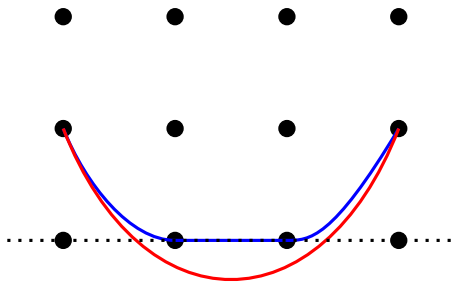


Figure 6: Unconstrained and graph-approximated geodesic paths.

graph-based approach naturally relaxes implementation restrictions (replacing a nested optimization NMPC problem with a paired "optimization – shortest-path" NMPC problem), facilitating the use of geodesics instead of any shorter path c_i (cf. (18)), offering tighter constraints (see (22)) and hence improved performance (in terms of convergency). Consequently, this approach also improves the prediction capabilities of existing contraction-constrained NMPCs [17, 18].

6.3 State and Control Constraint Handling

If the DCCM can be synthesized offline using (16) $\forall x \in \mathcal{X}, \forall u \in \mathcal{U}$, the existence of a contraction-based controller which is valid in the desired operating region can be ensured. However, this does not ensure the feasibility of the NMPC problem in (23) during online control, i.e., simultaneously satisfying the contraction constraint with both the state and control constraints (an open problem for contraction-based control).

In the proposed approach, implementation of the graph-based geodesic computation method during online control (via solution to the contraction constrained NMPC problem) ensures both geodesic boundedness and path feasibility (in terms of geodesic convexity). As stated in Section 5.2, this is an inherent property due to offline construction of a mesh which only considers the state values within the permissible ranges. Consequently, solutions to the shortest path problem (over that mesh) yield graph-based geodesic paths and hence geodesic distances that also satisfy the state constraints. This can be illustrated in Figure 6, where the red and blue curves are the unconstrained and graph-approximated geodesic paths respectively and the dotted line is a state constraint. This assists the MPC to generate optimal solutions subject to the state constraints by directly avoiding paths that are not compliant with them. This is reflected in the following statement.

Lemma 6.3. *For the system* (1)*, with graph* $G(V, H)$*, with vertices* V (29) *and edge weights* H (31)*, a geodesic obtained via Section 6.2 is bounded and satisfies the state constraints.*

Proof. For two points $x, x^* \in \mathcal{X}$, the geodesic $\gamma(x, x^*)$ is bounded by a Euclidean ball, i.e., the maximum geodesic length $d_\gamma(x, x^*)$ for any points along the s-parameterized path is upper bounded as $d_\gamma(\gamma(s), x^*) \leq \overline{m} d_e(x, x^*)$. Furthermore, we have $\underline{m} d(\gamma(s), x^*) \leq d_\gamma(\gamma(s), x^*) \leq \overline{m} d_e(x, x^*)$. Thus, for any points on the path, the geodesic is in a ball with radius $\overline{m}/\underline{m} d_e(x, x^*)$. Since $V \in \mathcal{X}$, the

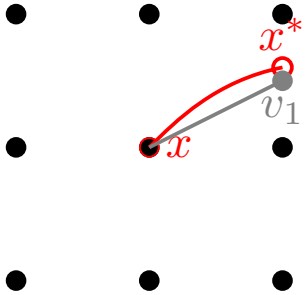


Figure 7: Using additional vertices (v_1) for improved approximation.

approximated geodesic path is the sequence of state values belonging to the intersection of state constraints and all Euclidean bounding balls. □

To deal with control constraints in contraction-based control, less aggressive contraction rates can be searched offline, resulting in smaller control action magnitudes, and hence comply with the system constraints. One possible way (adapted from our previous results [17]) to ensure feasibility of the locally constrained NMPC problem is to search for a uniform feasible convergence rate β_m for a given metric $M(x)$ as follows

$$\begin{aligned} & r(x, x^*, u, \beta) \leq 0 \qquad (18) \\ \text{s.t.} \quad & 0 < \beta_m \leq \beta, \\ & x, x^* \in \mathcal{X}, \ u \in \mathcal{U}. \end{aligned} \tag{49}$$

The feasibility problem (49) can be solved offline by implementing a line search on β_m and a sampling method for constraint verification. Assuming there exists a feasible contraction rate, β_m in (49), set $\beta = \beta_m$ from (49) when imposing the contraction constraint (18) on the NMPC in (23). Consequently, by considering β as a tunable variable in (49), a reduction in magnitude to satisfy constraint feasibility will result in longer convergence times (in the sense of state to reference trajectories). To further to reduce conservatisms introduced by a globally relaxed convergence rate, β_m, state-dependent solutions to (49) can be stored, resulting in a state-dependent contraction rate $\beta_m = \beta_m(x)$. By searching across the entire state space grid in $\mathcal{X}$, a state region and control constraint feasible contraction constraint could then be imposed, from (18), as

$$d\left(c_{i+1}\right) - \left(1 - \beta(x)\right)^{i+1/2} d\left(\gamma_0\right) \leq 0, \tag{50}$$

which further allows for non-uniform distributions and reduces the conservatism introduced by forcing a reduced convergence rate across the entire state space. Consequently, relative to existing approaches, the proposed graph-based method not only provides reduced online computation burden it also provides a flexible way to assist with constraint handling.

Table 1: Process Parameters

F	$4.998[m^3/h]$	κ_{10}	$3 \times 10^6[h^{-1}]$
V_r	$1[m^3]$	κ_{20}	$3 \times 10^5[h^{-1}]$
R	$8.314[KJ/kmolK]$	κ_{30}	$3 \times 10^5[h^{-1}]$
T_{A0}	$300[K]$	E_1	$5 \times 10^4[KJ/kmol]$
C_{A0}	$4[kmol/m^3]$	E_2	$7.53 \times 10^4[KJ/kmol]$
ΔH_1	$-5.0 \times 10^4[KJ/kmol]$	E_3	$7.53 \times 10^4[KJ/kmol]$
ΔH_2	$-5.2 \times 10^4[KJ/kmol]$	σ	$1000[kg/m^3]$
ΔH_3	$-5.4 \times 10^4[KJ/kmol]$	c_p	$0.231[KJ/kgK]$

7 Illustrative Example

Consider a well mixed, nonisothermal CSTR where 3 parallel irreversible elementary exothermic reactions take place of the form $A \to B$, $A \to C$, $A \to D$ [29], whereby the process can be modeled as follows,

$$\begin{aligned} x_{1_{k+1}} &= x_{1_k} + \Delta_t \left(\frac{F}{V_r}(C_{A0} - x_{1_k}) + \sum_{i=1}^{3} \kappa_{i0} e^{\frac{-E_i}{Rx_{2_k}}} x_{1_k} \right) \\ x_{2_{k+1}} &= x_{2_k} + \Delta_t \left(\frac{F}{V_r}(T_{A0} - x_{2_k}) - \sum_{i=1}^{3} \psi_i(x_{1_k}) + u_k \right), \end{aligned} \tag{51}$$

where $\psi_i(x_{1_k}) = \frac{\Delta H_i}{\sigma c_p} k_{i0} e^{\frac{-E_i}{Rx_{2_k}}} x_{1_k}$, the feed to the reactor consists of reactant A with molar concentration C_{A0} at flow rate F and temperature T_{A0}. This process model consists of two states, where $x_1 = C_A$ denotes the concentration of reactant A, and $x_2 = T$ denotes the temperature of the reactor. The states x_1 and x_2 are controlled by manipulating $u = \frac{Q}{\sigma C_p V_r}$, where Q represents the rate of heat input/removal. For the remaining process variables, V_r denotes the volume of the reactor, $\Delta H_i, \kappa_{i0}, E_i$ denotes the enthalpies, preexponential constants and activation energies of the three reactions, respectively, c_p and σ denote the heat capacity and density of the fluid in the reactor, respectively. For simulation purposes, the system model is normalized in the range of operation with a sampling period of $\Delta_t = 0.05\ h$. All parameters are shown in Table 1. The corresponding $W = M^{-1}$ and $L = KW$ can be synthesized with contraction rate $\beta = 0.01$ using SOS programming as stated in (16) (see our recent work[20] for more details). A graph $G(V, H)$ is constructed with respect to the metric synthesized, i.e., $M = W^{-1}$. This graph is constructed with uniformly distributed mesh with width length 0.01 using structure in Figure 2a, the graph covers the constrained area of state $x \in \mathcal{X} := [0.1, 1]$, the control is constrained as $u \in \mathcal{X} := [0, 4]$, and the topology of the edges are constructed as tier 1 (8 nodal neighbors) shown in Figure 3. The storage size of the uncompressed data is 21MB (.mat file). The computational time of the graph generation is 20.26s using parallel computing tools in Matlab on a Windows PC (i7 9700k 32GB RAM).

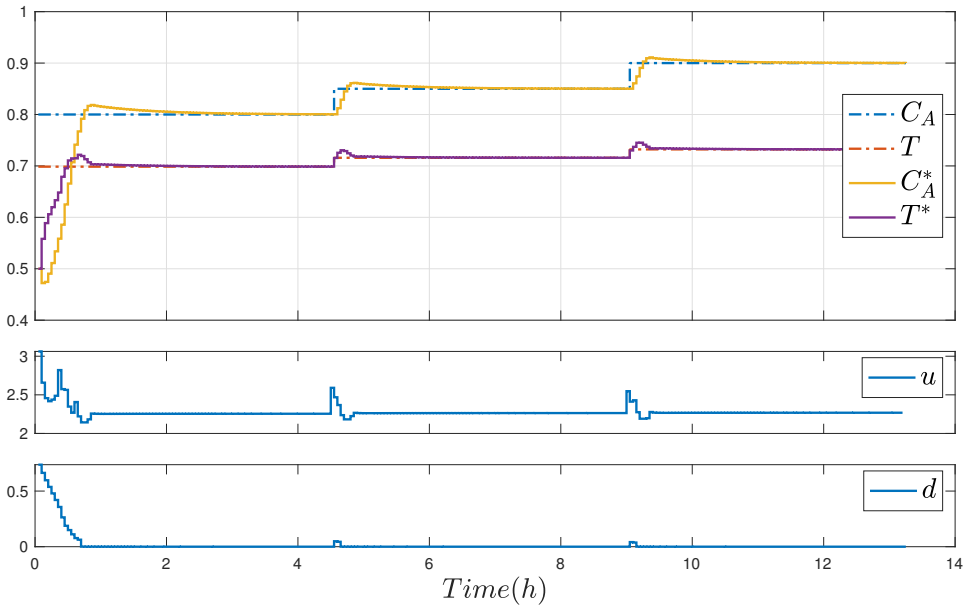


Figure 8: Simulation results of proposed discrete-time contraction constrained NMPC.

The NMPC as in (23) is constructed with a predictive horizon of 10 steps as follows,

$$\begin{aligned} &\min_{\hat{u}} \sum_{i=0}^{10} \|u_i\|, \\ &\text{s.t.} \quad \hat{x}_0 = x_k, \quad \hat{x}_{i+1} = f(\hat{x}_i) + g\hat{u}_i, \\ &\qquad \hat{u}_i \in \mathcal{U}, \quad \hat{x}_i \in \mathcal{X} \\ &\qquad r(\hat{x}_i, x_i^*, \hat{u}_i, \beta) \leq 0, \end{aligned} \tag{52}$$

where f and g correspond to the model in (51). The simulation parameters are as follows, the initial condition of state is $x_0 = (C_{A0}, T_0) = (0.5, 0.5)$, with setpoints of $(x_1^*, x_2^*) = (C_A^*, T^*) = (0.8000, 0.6986), (0.8500, 0.7158), (0.9000, 0.7321)$ on different time intervals as shown in Figure 8.

During online control, the geodesic is calculated using (32), (33), satisfying the contraction constraint in (18). As we can see in Figure 8, the states converge to the reference without any error for both states (upper plot) and the Riemannian distance, $d(x, x^*)$ (lower plot), is decreasing as per Proposition 4.1.

One possible alternative to imposing a stability (contraction) constraint is to use a weighted tracking-based objective function[9], which embeds the stability conditions into the cost function. However, weighted objective functions, even those which incorporate soft constraints, shifts the cost function representation away from a true economic cost and can result in an objective trade-off between target tracking and economy, potentially leading to infeasible computations without guarantee of convergence, especially when these objectives are competing [8]. This was verified via numerical study (not shown here) for (51), with (52) modified for use with the stage cost $\ell_i = \|x_i - x_i^*\| + \|u_i\|$. Clearly, minimization of the tracking cost $\|x_i - x_i^*\|$ (requiring $u \to u^* \neq 0$) is conflicting with min-

imizing energy input costs $\|u_i\|$ (requiring $u \to 0$) and results in poor performance or instability. Consequently, imposing stability constraints are naturally required [8]. Whilst defensible for a single operating point, it is unrealistic to design a Lyapunov-based stability constraint for all possible operation targets, warranting employment of contraction-based stability constraints whose reference flexibility is more befitting to RTO-driven control.

8 Conclusion

In this article, a discrete-time contraction constrained NMPC approach was developed. To ensure convergence to any feasible operation targets dictated by an RTO layer, a contraction based stability constraint was constructed and imposed on the NMPC. This condition utilizes Riemannian weighted graphs to solve the shortest path problem required for contraction. The resulting contraction constrained NMPC is reference flexible and can trade-off between the rate of target trajectory convergence and a general cost (such as an economic cost). The embedded graph-based geodesic computation method can significantly reduce the computational load for contraction theory based control and improve its applicability for real-time implementations.

References

[1] Bonvin D., ed. *Real-Time Optimization*. MDPI, Basel, Switzerland. 2018.

[2] Baldea M, Du J, Park J, Harjunkoski I. Integrated production scheduling and model predictive control of continuous processes. *AIChE Journal* 2015; 61(12): 4179–4190.

[3] Daoutidis P, Lee JH, Harjunkoski I, Skogestad S, Baldea M, Georgakis C. Integrating operations and control: A perspective and roadmap for future research. *Computers & Chemical Engineering* 2018; 115: 179–184.

[4] Qin SJ, Badgwell TA. A survey of industrial model predictive control technology. *Control Engineering Practice* 2003; 11(7): 733–764.

[5] Al Seyab R, Cao Y. Nonlinear model predictive control for the ALSTOM gasifier. *Journal of Process Control* 2006; 16(8): 795–808.

[6] Rangaiah G, Saha P, Tade M. Nonlinear model predictive control of an industrial four-stage evaporator system via simulation. *Chemical Engineering Journal* 2002; 87: 285–299.

[7] Rawlings JB, Mayne DQ, Diehl M. *Model Predictive Control: Theory, Computation, and Design*. Nob Hill Publishing Madison, WI. 2017.

[8] Ellis M, Liu J, Christofides P. *Economic Model Predictive Control: Theory, Formulations and Chemical Process Applications*. Advances in Industrial ControlSpringer International Publishing. 2017.

[9] Rawlings J, Mayne D. *Model Predictive Control: Theory and Design*. Nob Hill Publishing, LLC. 2009.

[10] Heidarinejad M, Liu J, Christofides PD. Economic model predictive control of nonlinear process systems using Lyapunov techniques. *AIChE Journal* 2012; 58(3): 855–870.

[11] Angeli D. A Lyapunov approach to incremental stability properties. *IEEE Transactions on Automatic Control* 2002; 47(3): 410–421.

[12] Santoso H, Hioe D, Bao J, Lee PL. Operability analysis of nonlinear processes based on incremental dissipativity. *Journal of Process Control* 2012; 22(1): 156–166.

[13] Lohmiller W, Slotine JJE. On contraction analysis for non-linear systems. *Automatica* 1998; 34(6): 683–696.

[14] Manchester IR, Slotine JJE. Control contraction metrics: Convex and intrinsic criteria for nonlinear feedback design. *IEEE Transactions on Automatic Control* 2017; 62(6): 3046–3053.

[15] Lopez BT, Slotine JJE. Contraction metrics in adaptive nonlinear control. `https://arxiv.org/abs/1912.13138e`; 2019.

[16] Wang R, Bao J. Distributed plantwide control based on differential dissipativity. *International Journal of Robust and Nonlinear Control* 2017; 27(13): 2253–2274.

[17] McCloy R, Wang R, Bao J. Differential dissipativity based distributed MPC for flexible operation of nonlinear plantwide systems. *Journal of Process Control* 2021; 97: 45–58.

[18] Xiao G, Yan Y, Bao J, Liu F. Robust distributed economic model predictive control based on differential dissipativity. *AIChE Journal* 2021; 67(6): e17198.

[19] Wei L, Mccloy R, Bao J. Control Contraction Metric Synthesis for Discrete-time Nonlinear Systems. *IFAC-PapersOnLine* 2021; 54(3): 661-666. *IFAC Symposium on Advanced Control of Chemical Processes ADCHEM 2021.*

[20] Wei L, McCloy R, Bao J. Contraction Analysis and Control Synthesis for Discrete-time Nonlinear Processes. `https://arxiv.org/abs/2112.04699`; 2021.

[21] Wei L, McCloy R, Bao J. Discrete-time Contraction-based Control of Nonlinear Systems with Parametric Uncertainties using Neural Networks. `https://arxiv.org/abs/2105.05432`; 2021.

[22] Leung K, Manchester IR. Nonlinear stabilization via control contraction metrics: A pseudospectral approach for computing geodesics. In: IEEE. ; 2017: 1284–1289.

[23] Crane K, Livesu M, Puppo E, Qin Y. A Survey of Algorithms for Geodesic Paths and Distances. `https://arxiv.org/abs/2007.10430`; 2020.

[24] Baek J, Deopurkar A, Redfield K. Finding Geodesics on Surfaces. Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, Tech. Rep.; 2007.

[25] do Carmo M. *Riemannian Geometry.* Mathematics (Boston, Mass.) Birkhäuser. 1992.

[26] Wang R, Manchester IR, Bao J. Distributed economic MPC with separable control contraction metrics. *IEEE Control System Letters* 2017; 1(1): 104–109.

[27] Long Y, Liu S, Xie L, Johansson KH. Distributed nonlinear model predictive control based on contraction theory. *International Journal of Robust and Nonlinear Control* 2018; 28(2): 492–503.

[28] Fredman ML, Tarjan RE. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)* 1987; 34(3): 596–615.

[29] Christofides PD, Liu J, De La Pena DM. *Networked and distributed predictive control: Methods and nonlinear process network applications*. Springer Science & Business Media. 2011.