Constructing Constrained Control Lyapunov Function and Synthesizing Controller from Null Controllable Region

Tuan Minh Nguyen

Abstract: This paper is an introductory tutorial on how to use the null controllable region (NCR) of a linear time-invariant system to approximate the system's constrained control Lyapunov function (CCLF) and from there synthesize a stabilizing controller for the system. Throughout the paper, every new concept will be illustrated with simple demonstrations. Additionally, only 2D and 3D systems will be considered for easy visualization of the system's evolution. I also provide an electronic supplement that contains well-documented Matlab codes for all examples presented in this paper. My goal is to provide the reader with the implementation perspective of the method rather than its maths and theories.

1. Introduction

In any engineering system, input constraints will always arise due to the physical limitations of actuators such as motors and servos. Neglecting the input constraints when designing a feedback controller might lead to the resulting closed-loop system underperforming or even unstable. Most commonly used controllers such as PID and LQR controllers do not explicitly consider the input constraints into their designs, but rather integrate these constraints later in the form of patching. More advanced controllers like model predictive control (MPC) can directly integrate the actuator limits by formulating them into constraints of an optimization problem. The feasibility and stability of the model predictive controller, however, are not always guaranteed, and the controller's performance strongly relies on the estimation of the infinite horizon cost or "tail-cost" as a Control Lyapunov Function (CLF). In general, finding a valid CLF is non-trivial, even for linear systems, and a good CLF candidate needs to take into account not only the system's dynamics but also the input constraints. Moreover, designing a CLF for multiple-input multiple-output (MIMO) systems with input constraints that can guarantee stability and maximize the region of attraction still remains an open problem.

In essence, if we can somehow estimate the valid CLF for a given system, then we can not only enforce system stability but also maximize the state-space region within which there always exists an admissible control that stabilizes the system. Such maximal stability-region is called null controllable region (NCR) which has been thoroughly studied for linear time-invariance (LTI) systems as in [1]. Furthermore, it has been shown in [2] that a constrained control Lyapunov function (CCLF) can be derived from the system's NCR, and the controller synthesized from such CCLF is guaranteed to stabilize the system in the presence of input constraints and maximize the region of attraction to be approximately equal to the system's NCR.

My primary goal for this paper is not to focus on the detailed maths but to convey how the maths are implemented in constructing the system's NCR, estimating the CCLF from the NCR, and finally synthesizing a stabilizing controller from the resulting CCLF. Throughout the paper, every new concept will be illustrated with simple demonstrations, and, for easy visualization of the system's evolution, only 2D and 3D systems will be considered. I also provide an electronic supplement that contains well-documented Matlab codes for all examples presented in this paper.

The rest of the paper is organized as follows. In Section 2, I will review the linear system description and discuss the methods in constructing the NCR. In Section 3, I will present a way to numerically estimate CCLF from NCR and then synthesize a controller from the resulting CCLF which can achieve stability from the entire NCR. Section 4 would focus on synthesizing a stabilizing controller from the system's NCR. In Section 5, more examples and simulations are presented to illustrate Sections 2,3&4 as well as to compare the proposed CCLF-base controller to the linear quadratic regulator (LQR) controller.

2. Preliminaries

This section describes the class of system dynamics that will be considered, and review the existing method for NCR construction.

2.1 Linear System

We consider the linear time-invariant multi-input multi-state systems describe by:

$$\dot{x} = Ax + Bu \quad \forall\, u \in U$$

, where $x \in R^n$ is the state vector, $u \in R^m$ is the input vector, A is an $n \times n$ matrix, and B is a $n \times m$ matrix. Each control input $u_i \; \forall \; i = 1,\ldots, m$ is bounded and only takes values within the set $U_i := \{u_{i,min} \leq u_i \leq u_{i,max}\} \; \forall \; i = 1,\ldots, m$, where $u_{i,min}$ and $u_{i,max}$ are the corresponding lower and upper bound of u_i respectively. In this paper, we will only consider the case where each control input u_i is bounded symmetrically. Without loss of generality, we assume that $u_{i,min} = -1 \;\&\; u_{i,max} = 1 \; \forall \; i = 1,\ldots, m$. If this is not the case, then the matrix B can be adjusted to "absorb" the true symmetric input constraints. For example, consider the following 2D systems with 2 control inputs, and the input constraints are $U_1 := \{-1 \leq u_1 \leq 1\}$ and $U_2 := \{-2 \leq u_2 \leq 2\}$

$$\dot{x} = Ax + \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} u \xrightarrow[B]{Adjusted} \dot{x} = Ax + \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} u$$

After the matrix B is adjusted, the new constraint for the second input u_2 becomes $U_2 := \{-1 \leq u_2 \leq 1\}$

2.2 NCR

An initial state $x(0)$ is called null controllable if there exists some admissible control action $u(t) \in U$ such that the system's trajectory starting from $x(0)$ will reach the origin in finite time $x(T) = 0 \; \forall \, T \in [0, \infty)$. The region that encompasses of all the null controllable initial state is called the null controllable region (NCR), denoted as C, and the boundary of the NCR will be denoted as ∂C. Given an anti-stable LTI system (eigenvalues of matrix A is on the open left-half plane e.g. real(eig(A)) > 0), an explicit characterization of the single-input NCR is well defined [1]. If a LTI system is multi-input $u \in R^m$, then it can be rewritten such that the overall control actions are decomposed into a "linear combination" of individual of control input $u_i \in R \; \forall \; i = 1,\ldots, m$. Consequently, the original system can be partitioned into m single-input subsystems whose NCR can be found individually $C_i \; \forall \; i = 1,\ldots, m$. Then, the overall NCR can then be calculated by taking the minkowski sum of all the single-input subsystems's NCRs $C = C_1 \oplus C_2 \oplus \ldots \oplus C_m$. To illustrate this, consider the following 2D LTI system with 2 control actions:

$$\dot{x} = Ax + \begin{bmatrix} \alpha_1 & \beta_1 \\ \alpha_2 & \beta_2 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

$$\dot{x} = Ax + \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} u_1 + \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} u_2$$

$$\begin{cases} \dot{x} = Ax + \begin{bmatrix} \alpha_1 \\ \alpha_2 \end{bmatrix} u_1 \xrightarrow[NCR]{Find} C_1 \\ \dot{x} = Ax + \begin{bmatrix} \beta_1 \\ \beta_2 \end{bmatrix} u_2 \xrightarrow[NCR]{Find} C_2 \end{cases}$$

$$C = C_1 \oplus C_2$$

In general, an LTI system might also contain eigenvalues on both the left and right half-plane. When this happens, we can use the Jordan decomposition to partition the original system into a stable subsystem whose corresponding A_{stable} matrix is such that $eig(A_{stable}) \equiv eig(A)$ on the closed left-half plane, and an anti-stable subsystem whose corresponding $A_{unstable}$ matrix is such that $eig(A_{unstable}) \equiv eig(A)$ on the open left-half plane. Then, the NCR analysis can be done on the anti-stable subsystem. For example, consider the following system which is already in the Jordan form:

$$\dot{x} = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix} x + \begin{bmatrix} B_1 \\ B_2 \end{bmatrix} u$$

$$, where\ A_1 = A_{stable} \in R^{n_1 \times n_1}, B_1 \in R^{n_1 \times m},$$

$$A_2 = A_{unstable} \in R^{n_2 \times n_2}, B_2 \in R^{n_2 \times m}$$

$$n = n_1 + n_2$$

$$\begin{cases} Stable\ subsystem & : \dot{x_1} = A_1 x_1 + B_1 u \\ Anti\text{-}stable\ subsystem & : \dot{x_2} = A_2 x_1 + B_2 u \end{cases}$$

Assume that the LTI system is (A,B) controllable, then:

(a) For the stable subsystem whose dimension is $n_1 \leq n$, the NCR would span the entire state-space of the stable subsystem e.g. $C_{stable} = R^{n_1}$.

(b) For the anti-stable subsystem whose dimension is $n_2 \leq n$, the NCR is a bounded convex set containing the origin e.g. $C_{unstable} \subset R^{n_2}$.

(c) For the original system, the overall NCR would be a tube-like shape e.g. $C = C_{stable} \times C_{unstable} = R^k \times C_{unstable}$

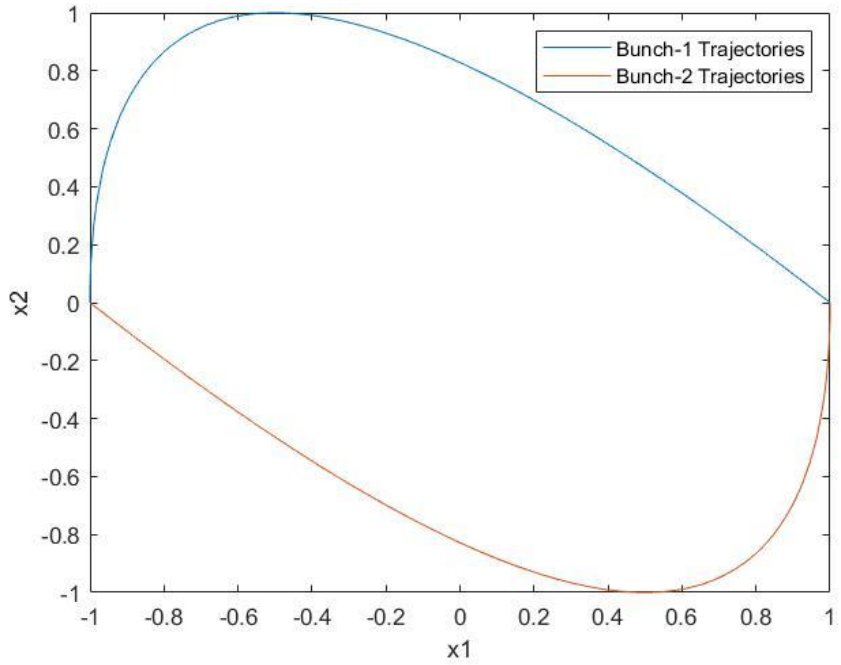
The derivation for the NCR's boundary of an anti-stable LTI system with input constraints $U_i := \{u_{i,min} \leq u_i \leq u_{i,max}\}\ \forall\ i = 1,\ldots, m$ can be found in [1]. For systems with only real eigenvalues, ∂C can be computed as follows:

$$\partial C = \left\{ \pm \left[\sum_{i=1}^{n-1} 2(-1)^i e^{-A(t-t_i)} + (-1)^n I \right] A^{-1} b : 0 = t_1 \leq t_2 \leq \ldots \leq t \leq \infty \right\}$$

Notice that every $t \geq t_{n-1}$ that is inputted into the above equation results in a point on the extremal trajectory evolving along ∂C. Essentially, ∂C is covered by two bunches of trajectories. Bunch-1 consists of trajectories that starts from the point $A^{-1}b$, while Bunch-2 consists of trajectories that starts from the point $-A^{-1}b$. The evolutions of these trajectories depend on the choices of t_{n-1}. For example, If the system is two-dimensional (n = 2), then we only have one $t_{n-1} = t_1 = 0$, and thus there are only 2 possible trajectories, one from each bunch. If the system's demonsion is more than 2, then t_{n-1} can be chosen to be any value as long as $t_{n-1} \geq 0$, and thus we would have an infinite number of possible trajectories all of which cover ∂C. The Matlab files *Demo1* and *Demo2* include the demo codes for constructing ∂C for 2D and 3D systems respectively. In Demo1, we consider the 2D system:

$$\dot{x} = \begin{bmatrix} 0 & -0.5 \\ 1 & 1.5 \end{bmatrix} x + \begin{bmatrix} 0 \\ -1 \end{bmatrix} u$$

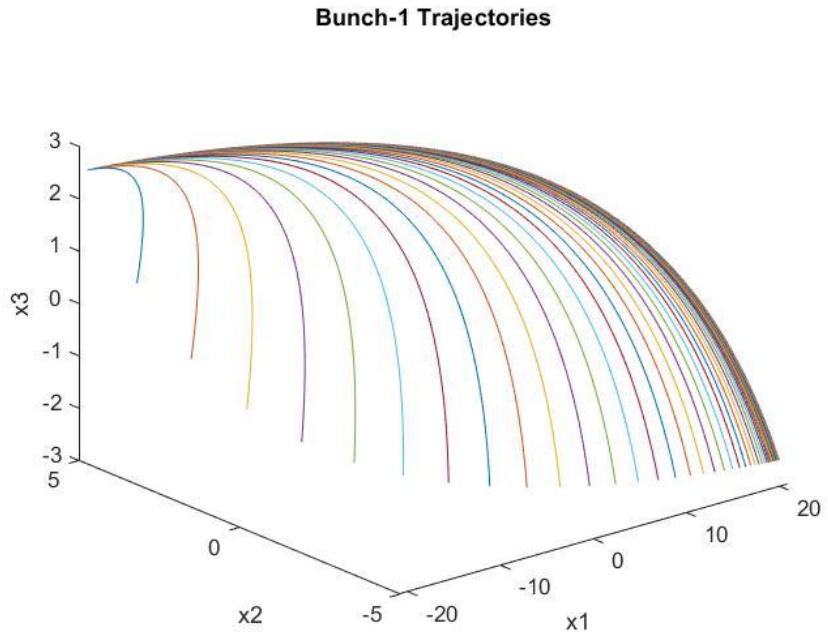
, and its ∂C is:

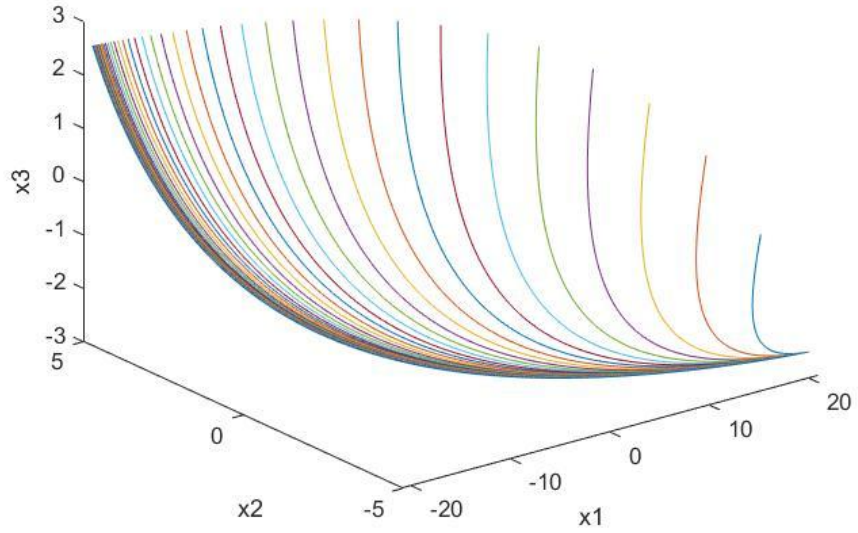


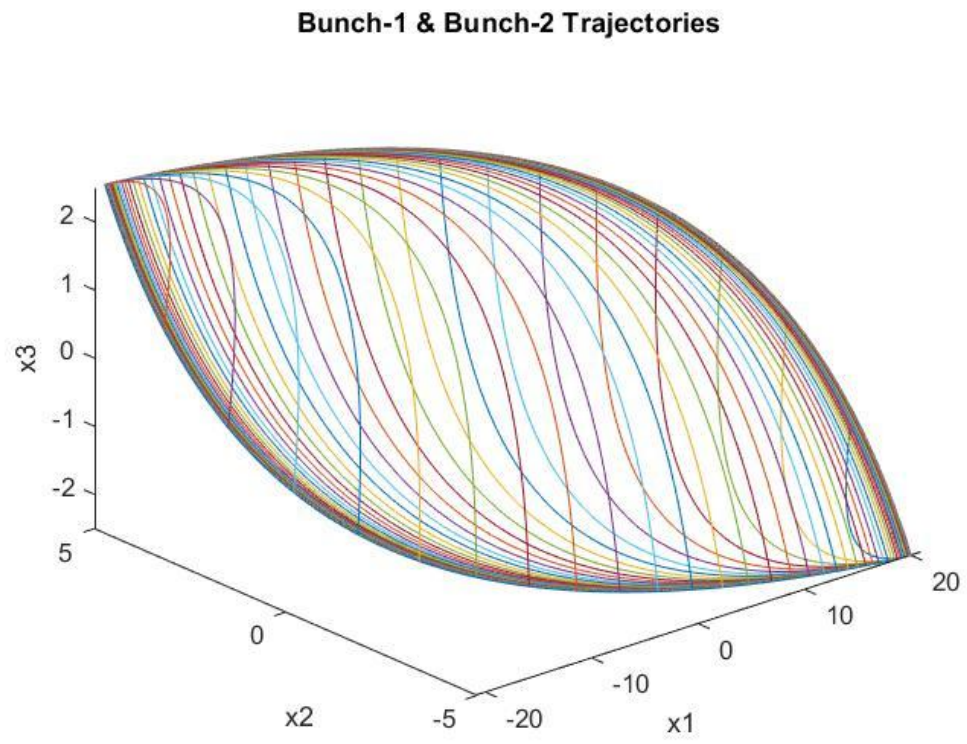
In Demo2, we consider the 3D system:

$$\dot{x} = \begin{bmatrix} 0.2 & 1 & 0 \\ 0 & 0.2 & 0 \\ 0 & 0 & 0.4 \end{bmatrix} x + \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} u$$

, and its ∂C is:







For systems with a mixture of real and imaginary eigenvalues, there is no closed-form equation for ∂C and, for the sake of convenience, we would not consider such systems in this paper. Nevertheless, the NCR analysis for systems with real and imaginary eigenvalues can be found in chaper 2.5 in [1].

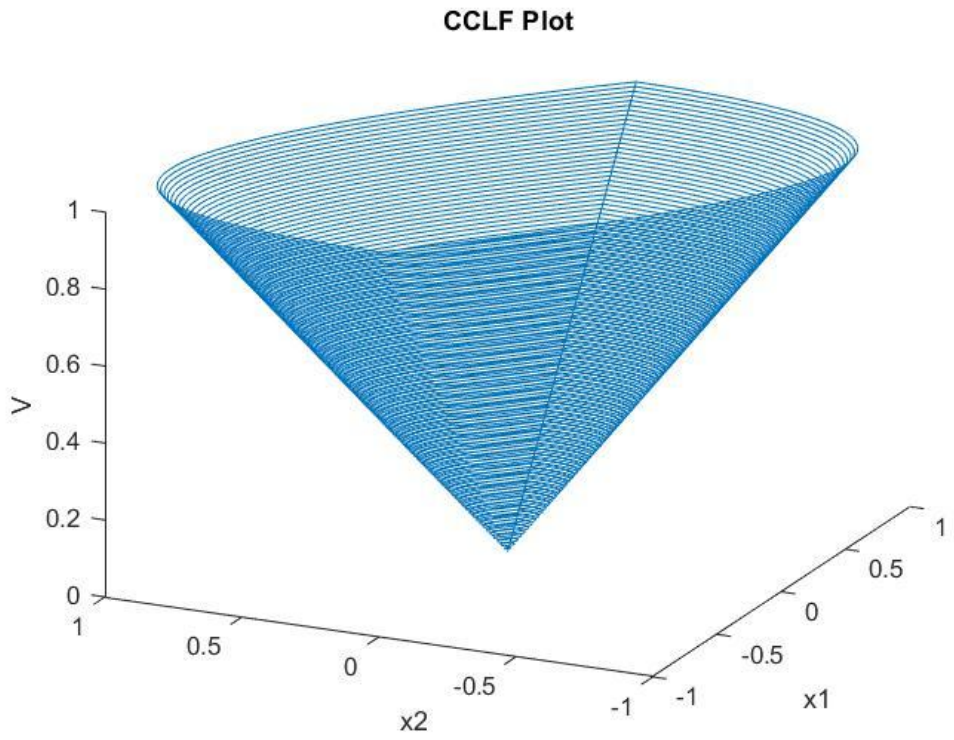
3. Numerically Estimating CCLF from NCR

Given an anti-stable linear system, a valid constrained control Lyapunov function (CCLF) for the system is the Minkowski functional of the system's null controllable region (NCR) [2][3].

$$V(x) \equiv \alpha \quad s.t \quad x \in \alpha(\partial C)$$

The above equation essentially means that that given the system's current state x, the value of the CCLF V(x) equals to the positive scalar α such that scaling the system's NCR with α will result in the state x to be located right on the boundary of the scaled NCR e.g. $x \in \alpha(\partial C)$. Notice that scaling ∂C with α is equivalent to multiplying every points on ∂C by α. Therefore, to find α given x, we can first formulate a "look-up table"

which stores all the scaled values of $\alpha(\partial C)$ with respect to a chosen range of α and then interpolate the given point x with points in the table to estimate the value of α. The look-up table can be easily constructed using the Matlab's scatteredInterpolant function which is demonstrated in *Demo3*. *Demo3* first uses the Matlab's boundary function to uniquely extract the boundary points of the system's ∂C. The boundary points are then multiplied by the α values ranging from 0 to 1 wih 0.01 increments. The resulting scaled boundary points are stored in the Matlab's scatteredInterpolant function which serves as the numerical approximation (lookup table) for the CCLF V(x). The plot of the system's CCLF are shown in the figure below:



4. Synthesizing Control Law from CCLF

Since the V(x) derived in Section 3 is a valid CCLF [2] and thus a valid control Lyapunov function (CLF), based on the definition of CLF, there would always exist some control action u such that time derivative of V(x) is non-positive e.g. $\dot{V}(x,u) \leq 0$, which means that the system is stable. Additionally, if we can determine a control u such that $\dot{V}(x,u) < 0 \; \forall \; x \neq 0$, then the system is guaranteed to be asymptotically stable.

Time derivative of V(x) along the LTI system's dynamics:

$$\dot{V}(x,u) = \frac{dV(x)}{dx}\frac{dx}{dt} = \frac{dV(x)}{dx}\dot{x} = \frac{dV(x)}{dx}(Ax + Bu) = \frac{dV(x)}{dx}Ax + \frac{dV(x)}{dx}Bu$$

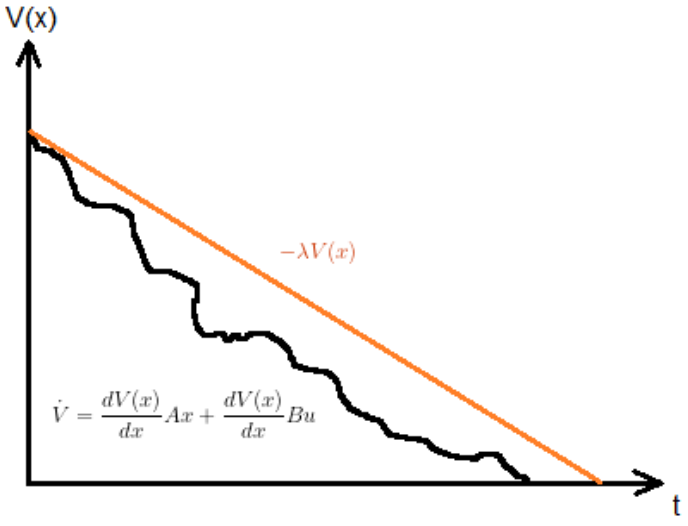
To ensure stability, we need to find an input u such that:

$$u = \underset{u \in U}{argmin} \|u\|_2$$
$$s.t. \;\; \frac{dV(x)}{dx}Ax + \frac{dV(x)}{dx}Bu \leq 0$$

To ensure asymptotic stability and to control the rate at which trajectory x(t) approach the origin, we modify the constraint such that:

$$u = \underset{u \in U}{argmin} \|u\|_2$$
$$s.t. \;\; \frac{dV(x)}{dx}Ax + \frac{dV(x)}{dx}Bu \leq -\lambda V(x)$$

, where λ is a positive scalar and $-\lambda V(x)$ is the upper-envolope of $\dot{V}(x,u)$



Notice that the larger the lambda λ the steeper the slope of the orange line, and thus the faster trajectory x(t) converges to the origin since V(x) would approach zero faster. However, if we choose λ to be very large, then there might be no admissible $u \in U$ that can satisfy the constraint $\dot{V}(x,u) \leq$ -λV, and thus the optimization problem would be infeasibile. Therefore, to ensure the feasibility of the problem, we introduce slack variable s that result in the relaxed constraint $\dot{V}(x,u) \leq$ -$\lambda V + s$. The overall control law u(x) will have the following form:

$$u(x(t)) = \underset{u,s}{argmin} \|u\|_2 + Ms$$

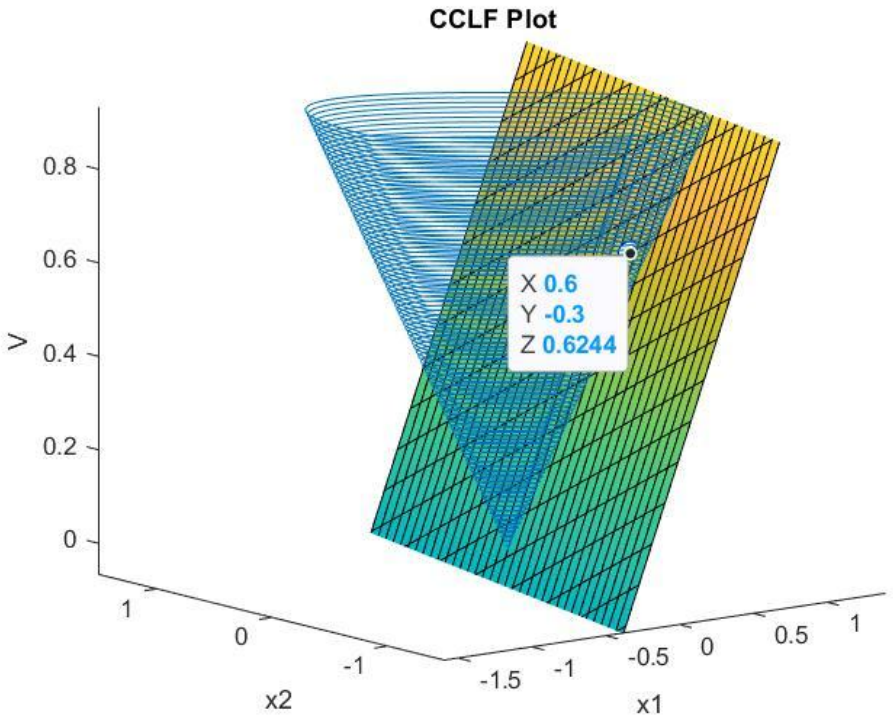
$$s.t \quad \frac{dV(x)}{dx}Ax + \frac{dV(x)}{dx}Bu \leq -\lambda V(x) + s$$

$$x = x(t)$$
$$u \in U$$
$$s \geq 0$$

, where M is a very big number, and x(t) is the system's current state

Notice that when we plug in x=x(t), the optimization problem become a quadratic program which is fast to solve and thus suitable for online-optimzation.

However, finding $\frac{dV(x)}{dx}$ is not straight forward since we only have the numerical approximation of V(x) from Section 3. Therefore, to estimate $\frac{dV(x)}{dx}$ at point $\bar{x}$, we can use Euler approximation to find the slope of the tangent line to V(x) that touches the point $\bar{x}$. Since we do not have the closed-form expression for V(x), exactly finding the tangent that touches $\bar{x}$ would be infeasible. Instead, the tangent line can be approximately constructed by fitting a line around a small neighborhood of the point $\bar{x}$. The fitting is done using linear regression and is illustrate in *Demo4*, which uses the CCLF V(x) found in *Demo3*.



In the above figure, the estimated tangent hyperplane at the point (x1,x2)=(0.6,-0.3) has the equation of:

$$0.94336x_1 - 0.19809x_2 = 0.0010087$$

Therefore, at the point (x1,x2)=(0.6,-0.3), we can estimate:

$$\frac{dV}{dx} = [0.94336 \quad -0.19809]$$

For the sake of convenience, finding dV/dx and implementing the CCLF-controller will be separately implemented in the Matlab functions dVdzCal() and u_CLF() respectively. In section 5, the two functions would be used in all of the examples.

5. Simulation Examples

In this section, examples of synthesizing a CCLF controller for multi-input systems and systems whose eigenvalues are on both the right and left half-planes will be illustrated. Moreover, a direct comparison between CCLF controller and LQR controller will be considered to demonstrate the strength of CCLF controller in maximizing the system's region of attraction under the input constraints.

5.1 Multi-Input Systems

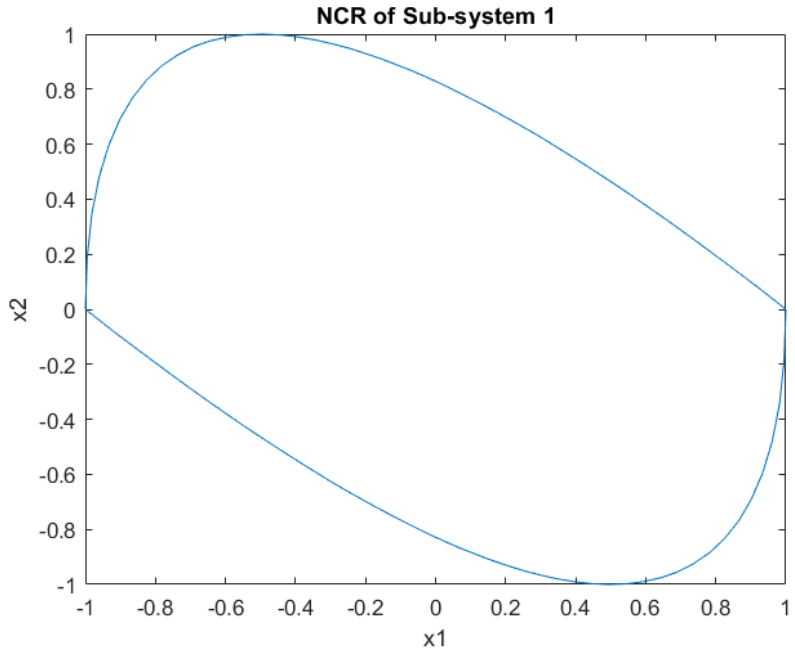
As discussed in Section 3, the overall NCR of a multi-input LIT system is the Minkowski sum of the NCR's contributed by each input. This section provides an example of how to synthesize a CCLF-controller for a 2-dimensional system that has 2 control inputs. The Matlab code for this example is in *Demo5*, and the system of interest is the following:

$$\dot{x} = \begin{bmatrix} 0 & -0.5 \\ 1 & 1.5 \end{bmatrix} x + \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

The first single-input subsystem would then have the form of:

$$\dot{x} = \begin{bmatrix} 0 & -0.5 \\ 1 & 1.5 \end{bmatrix} x + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u_1$$

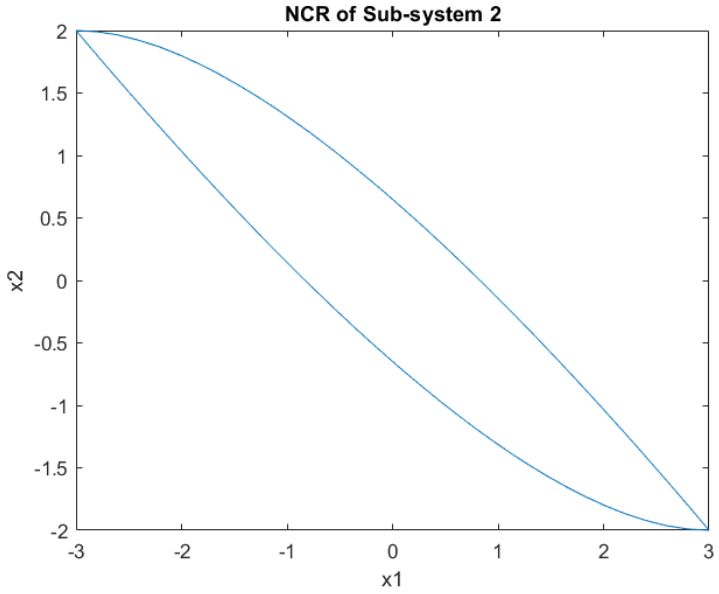
, and its corresponding NCR is:



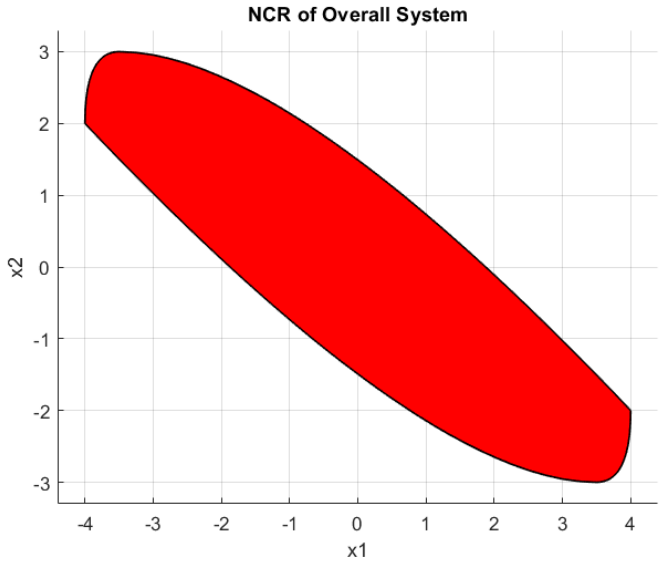
The second single-input system is:

$$\dot{x} = \begin{bmatrix} 0 & -0.5 \\ 1 & 1.5 \end{bmatrix} x + \begin{bmatrix} 1 \\ 0 \end{bmatrix} u_2$$

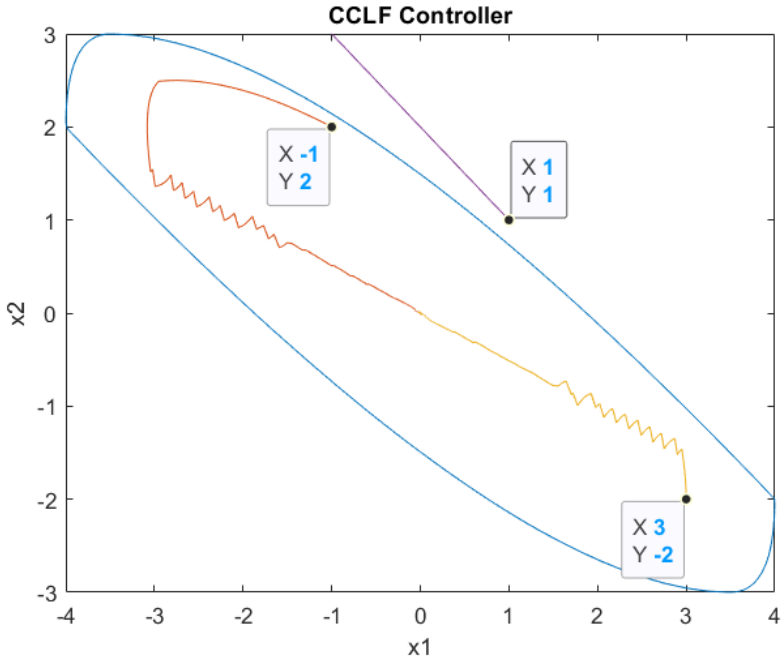
, and its corresponding NCR is:



The NCR of the original system is the Minkowski sum of the NCRs of subsystems 1&2. The Minkowski can be easily calculated using the multi-parametric toolbox [4].



With the calculated overall NCR, a CCLF-controller can be synthesized to stabilize the system at the origin. In *Demo5*, the synthesized CCLF-controller would try the drive the system toward the origin from 3 different initial states, and the results are as follow:



Notice that since the initial states [x1,x2] = {[-1,2];[3,-2]} are inside the system's NCR, their trajectories can thus be driven to the origin under the input constraints. On the other hand, since the initial state [x1,x2] = [1,1] is outside of the system's NCR, the control inputs are unable to drive the system to the origin.

5.2 Systems with Stable and Anti-stable Eigenvalues

Recall from Section 2.2 that if a system is anti-stable (eigenvalues on the right-half plane), then its NCR would be well defined. If a system is stable (eigenvalues on the right-left plane), then its NCR would span the entire state space, and thus we would not know the specific geometry of the NCR. As a result, if a 3-dimensional system contains eigenvalues on both the right- and left-half planes then the system's NCR would be an infinitely long tube in which the tube's infinite length is due to the stable subsystem and the tube's cross-section is defined by the NCR of the anti-stable subsystem. This concept is illustrated in *Demo6* which considers the following system:

$$\dot{x} = \begin{bmatrix} 3 & 1 & 1 \\ 0 & -5 & 0 \\ 0 & 2 & 3 \end{bmatrix} x + \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} u$$

The eigenvalues of this system are 3 and -5 which are on both the right and left-half planes. To decompose the original system into its corresponding stable and anti-stable subsystems, we use the Jordan decomposition which decomposes the matrix A into the followings:

$$A = MJM^{-1} \; \forall \; J = \begin{bmatrix} 3 & 1 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & -5 \end{bmatrix} \; \& \; M = \begin{bmatrix} 1 & 0 & -0.0906 \\ 0 & 0 & 0.9662 \\ 0 & 1 & -0.2415 \end{bmatrix}$$

Then, we consider the state transformation $z = M^{-1}x$, which results in the following new dynamics:

$$\dot{z} = M^{-1}\dot{x} = M^{-1}(Ax + Bu) = M^{-1}(AMz + Bu) = M^{-1}AMz + M^{-1}Bu$$
$$\dot{z} = Jz + \overline{B}u \quad \forall \, \overline{B} = M^{-1}B$$

Anti-stable subsystem

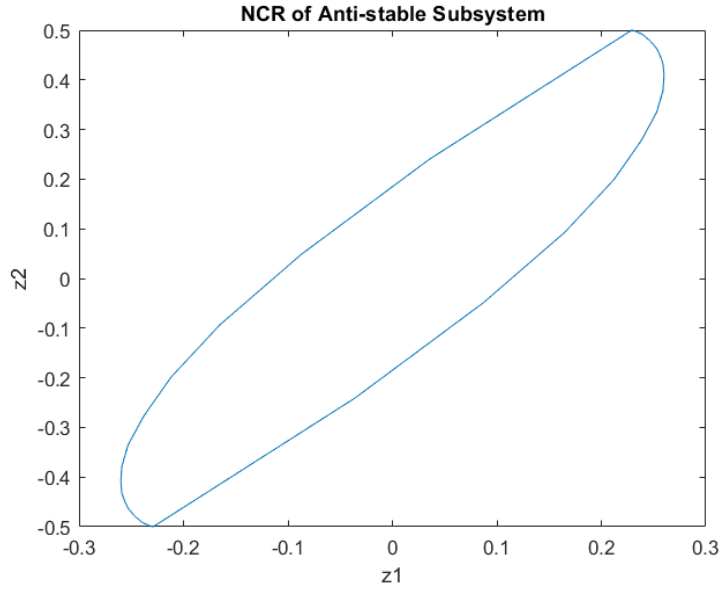
$$\dot{z} = \begin{bmatrix} 3 & 1 & 0 \\ 0 & 3 & 0 \\ 0 & 0 & -5 \end{bmatrix} z + \begin{bmatrix} 1.1875 \\ 1.5 \\ 2.0701 \end{bmatrix} u$$

Stable subsystem

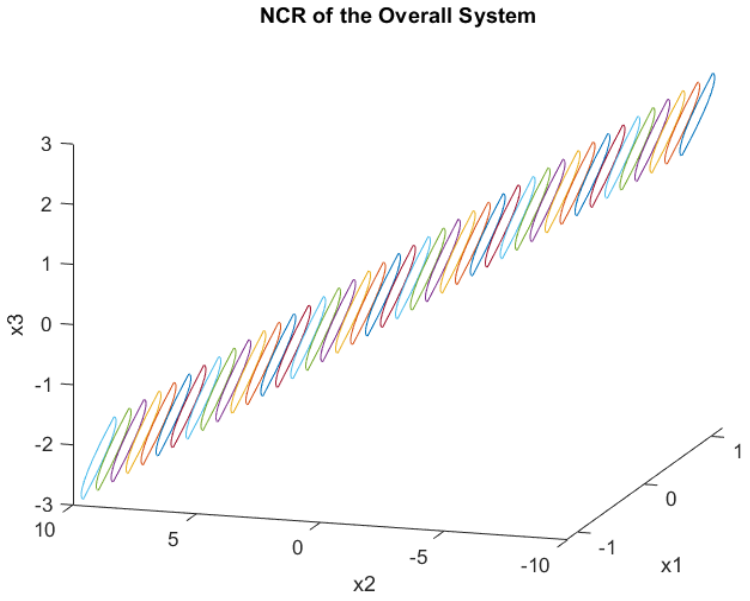
Notice that z1 and z2 are states of the anti-stable subsystem while z3 belongs to stable subsystem's state. As a result, the dynamics of the anti-stable subsystem is:

$$\dot{z} = \begin{bmatrix} 3 & 1 \\ 0 & 3 \end{bmatrix} z + \begin{bmatrix} 1.1875 \\ 1.5 \end{bmatrix} u$$

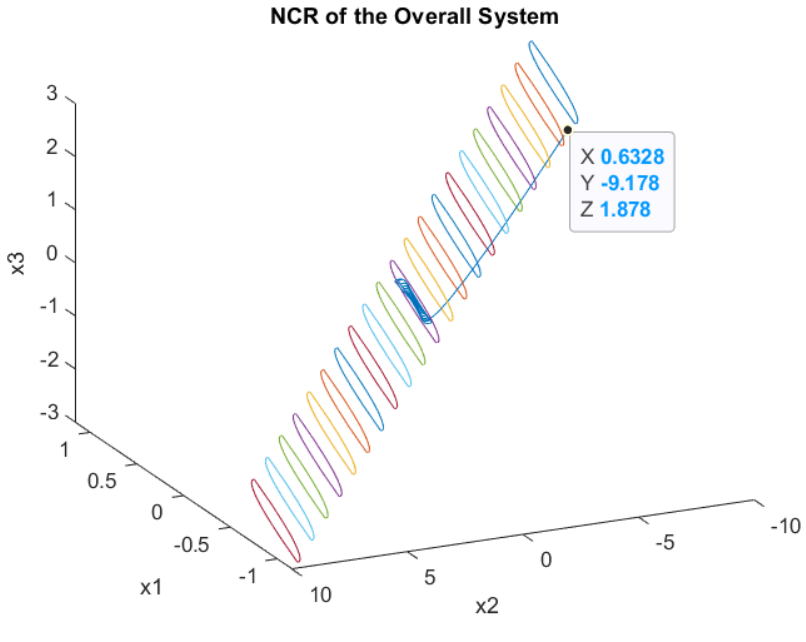
, and its correspond NCR is:



Since z3 can span the entire state space of the stable subsystem, the NCR tube in z-dynamics would have the z3-axis as its axis of symmetry, and the cross-section of the tube is exactly the same as the NCR of the anti-stable subsystem. The NCR tube in z-dynamics can be converted into the NCR tube in x-dynamics using the transformation $x = Mz$, and the resulting transformation is shown below:

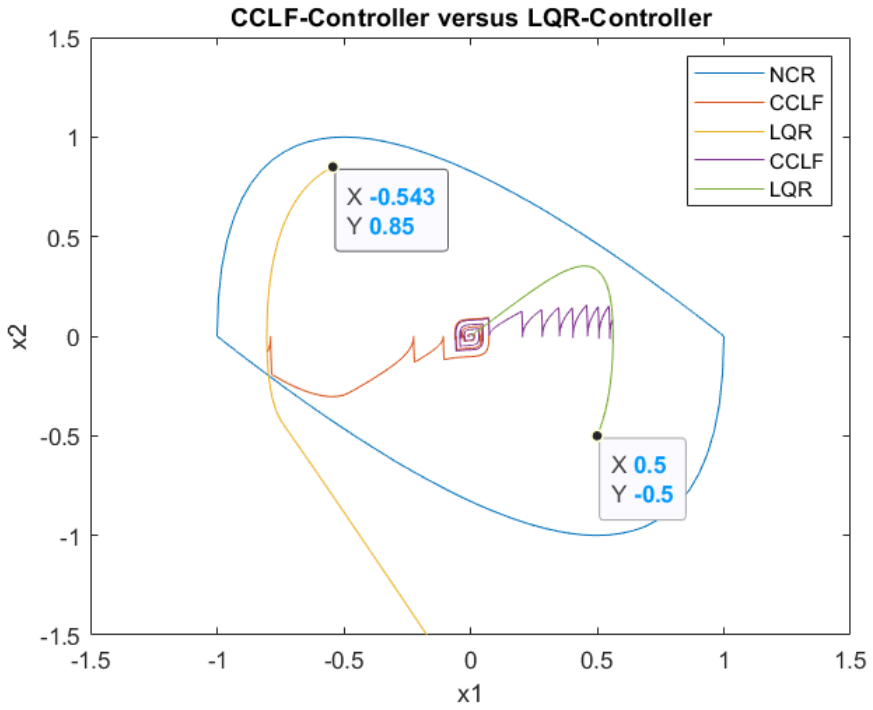


With the calculated overall NCR, a CCLF-controller can be synthesized to stabilize the system at the origin. In *Demo6*, since the initial state x = [0.6328,-9.178,1.878] is within the system's NCR tube, the control input is thus able to drive the system's trajectory to the origin as shown in the following figure:



5.3 CCLF Controller versus LQR Controller

This section provides a comparison between the region of attraction of a CCLF-controller and that of an LQR-controller. Intuitively, we would want to design a control law such that, given the input constraints, the region of attraction of that control law should be approximately equal to the system's NCR which is technically the best any control law can do. *Demo7* uses the same LTI system as that of *Demo1* and also synthesizes a CCLF-controller whose performance is then compared to an LQR-controller. The resulting comparison is as follow:



Unlike the LQR-controller, the CCLF-controller is not continuous so that the trajectories driven by the CCLF-controller are much fuzzier than those driven by the LQR-controller. Notice that if a trajectory starts well within the system's NCR, as in the case of the initial state [x1,x2]=[0.5025,-0.5], then both CCLF and LQR-controllers are able to drive the system to the origin. However, if a trajectory starts near the boundary of the system's NCR, as in the case of the initial state [x1,x2]=[-0.543,0.85], then only the CCLF-controller is able to stabilize the system to the origin while the LQR-controller diverges the system's trajectories to infinity. The goal of *Demo7* is to demonstrate that the capability of a CCLF-controller in maximizing the system's region of attraction under the input constraints. Even though the trajectories driven by a CCLF-controller might be fuzzy, this problem can be alleviated by integrating the system's CCLF into a model predictive controller (MPC) [3].

References

[1] Hu, Tingshu., and Zongli Lin. *Control Systems with Actuator Saturation: Analysis and Design.* Boston: Birkhäuser, 2001.

[2] M. Mahmood and P. Mhaskar, "On Constructing Constrained Control Lyapunov Functions for Linear Systems," in IEEE Transactions on Automatic Control, vol. 56, no. 5, pp. 1136-1140, May 2011, doi: 10.1109/TAC.2011.2114470.

[3] Battista, Angela, and Prashant Mhaskar. 2021. "Definition and Utilization of the Null Controllable Region for Model Predictive Control of Multi-Input Linear Systems" *Mathematics* 9, no. 10: 1110. https://doi.org/10.3390/math9101110

[4] M. Herceg, M. Kvasnica, C.N. Jones, and M. Morari. Multi-Parametric Toolbox 3.0. In *Proc. of the European Control Conference*, pages 502–510, Zurich, Switzerland, July 17–19 2013.