A Derivative-Free Optimization Method With Application to Functions With Exploding and Vanishing Gradients

Said Al-Abri, Tony X. Lin, *Graduate Student Member, IEEE*, Molei Tao, and Fumin Zhang, *Senior Member, IEEE*

***Abstract*—In this letter, we propose a bio-inspired derivative-free optimization algorithm capable of minimizing objective functions with vanishing or exploding gradients. The proposed method searches for improvements by leveraging a PCA-based strategy similar to fish foraging. The strategy does not require explicit gradient computation or estimation and is shown in simulation to require few function evaluations. Additionally, our analysis proves that the proposed algorithm's search direction converges to the gradient direction everywhere outside of small neighborhoods around local minima. Applications to a data-driven LQR problem and noisy Rosenbrock optimization problem are demonstrated. Empirical results show the proposed method exhibits fast convergence and is able to find the LQR gains for any controllable system, including unstable systems, and is robust to noisy function evaluations.**

***Index Terms*—Derivative-free optimization, exploding and vanishing gradient, data-driven systems, linear quadratic regulator.**

I. Introduction

IN DERIVATIVE-FREE (or 0-th order) optimization problems, objective functions with unknown analytical forms (also known as black-box functions) are directly optimized without estimating gradients or higher-order components derivatives [1], [2]. They have been widely used for deep neural network fitting and for policy optimization in control and robotic systems [3], [4], [5], [6], [7]. These methods have recently received more attention in the optimization community as useful approaches when an objective function has a derivative that is prohibitively expensive to estimate [4]. In particular, these methods may be better suited than gradient estimation methods when the gradient is ill-defined, i.e., the gradient either explodes or vanishes [8].

These ill-defined gradients naturally occur in many common optimization problems. In deep neural networks, vanishing or exploding gradients may prevent proper backpropagation of the network weights [9], [10], while in reinforcement learning, exploding gradients may cause parameter estimation to fail. In the case of searching for optimal controller parameters, these exploding gradients are naturally evident when the control parameters induce an unstable system [11].

In this letter, we leverage a modified version of a decentralized bio-inspired source-seeking strategy known as Speeding-Up or Slowing-Down (SUSD) [12], [13] as a sample-efficient derivative-free solution for optimization in scenarios where the gradients may be ill-defined, i.e., vanish or grow to infinity. We validate our method through a data-driven Linear Quadratic Regulator (LQR) problem (in which exploding gradients occur if the feedback gains induce instability) and a noisy Rosenbrock optimization problem (whose global minimum is inside a long and narrow flat valley of a vanishingly small gradient).

Many existing derivative-free optimization methods rely on a direct search based only on function evaluations. The authors of [14] analyze the Nelder-Mead algorithm that refines the shape of a simplex to find a local minimum. However, the method suffers from a lack of convergence results as the existing convergence analysis depends on specific dimensionality assumptions of the optimization problem [14], [15]. In [8], random search methods are analyzed in which candidates are drawn randomly near the current estimate. While the analysis of these methods is robust, they suffer from slow convergence rates and high sampling rates. A consensus-based gradient-free optimization method is presented in [7] which is effective for high-dimensional problems. However, this method lacks stability analysis for choosing the optimal parameters.

In model-free optimal control, Extremum Seeking Control (ESC) is another derivative-free search method which also

Manuscript received March 17, 2020; revised May 18, 2020; accepted June 9, 2020. Date of publication June 24, 2020; date of current version July 9, 2020. The work of Said Al-Abri, Tony X. Lin, and Fumin Zhang were supported in part by Office of Naval Research under Grant N00014-19-1-2556 and Grant N00014-19-1-2266; in part by Air Force Office of Scientific Research under Grant FA9550-19-1-0283; in part by NSF Grant CNS-1828678, Grant S&AS-1849228, and Grant GCR-1934836; in part by NRL under Grant N00173-17-1-G001 and Grant N00173-19-P-1412; and in part by NOAA under Grant NA16NOS0120028. The work of Molei Tao was supported by NSF under Grant DMS-1847802 and Grant ECCS-1829821. Recommended by Senior Editor C. Prieur. *(Corresponding author: Fumin Zhang.)*

Said Al-Abri, Tony X. Lin, and Fumin Zhang are with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: saidalabri@gatech.edu; tlin339@gatech.edu; fumin@gatech.edu).

Molei Tao is with the School of Mathematics, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: mtao@gatech.edu).

Digital Object Identifier 10.1109/LCSYS.2020.3004747

relies only on function evaluations [16], [17], [18], [19], [20]. However, ESC implicitly estimates the gradient of the objective function at each iteration which depends on well- behaved gradients. Inspired by ESC, the authors in [21] use non-commutative maps to approximate a gradient descent step, which might be unreliable in circumstances where the gradient is not well-behaved. As another alternative, trust-region methods are designed to ensure each iteration improves the minimum estimate [22]. These methods iteratively compute approximations (usually quadratic models) of the objective in a neighborhood called the "trust region" of the current estimate. However, the computation of the local approximation is expensive and iterations may not yield improvements if the estimate of the trust region is insufficiently accurate.

In the data-driven LQR problem, in which system trajectories are used to minimize the finite horizon cost of a linear time-invariant system, various solutions have also been proposed. The authors of [11] provide analysis of policy gradient algorithms that identify the LQR gains when the system (A, B) is unknown and the authors of [23] identify the LQR gains using formulas that completely describe the input-state (or input-output) behavior of the linear system. The approach described in [23] in particular is very efficient in that the formulas obtained require only one data set system trajectory. In cases where the open-loop system is unstable or an initial stable policy for the closed-loop system is unknown, the gradient of the LQR cost function explodes causing the approaches detailed in [11] and [23] to fail. As such, computing an estimate of the LQR gains when the system is unstable and unknown is still a difficult problem.

The main contributions of this letter are therefore as follows: **i)** generalizing SUSD as a derivative-free optimization method for general functions defined in a Euclidean space of arbitrary dimensions, **ii)** proposing a novel exponential mapping of the objective function that allows for the application of the SUSD algorithm to a wide variety of optimization problems with ill-defined derivatives, i.e., with vanishing or exploding gradients, **iii)** deriving the SUSD optimization dynamics, and stability and robustness analysis under both linear and exponential objective function mappings, and **iv)** obtaining empirical results for solutions to the data-driven LQR problem when the system is inherently unstable and comparisons with the Natural Policy Gradient (NPG). In addition, we provide empirical evidence that our approach is robust to noisy evaluations of the function by optimizing a high dimensional classical nonconvex test function (Rosenbrock) where function evaluations are perturbed by Gaussian noise.

The main challenge this letter has solved is in optimizing black-box functions in which the gradient is ill-defined. Compared to our previous 2D robotic source seeking work in [12] and [13], this letter proposes an exponential objective function mapping and considers objective functions that are nonconvex and of arbitrary large dimensions. In addition, the search dynamics derivations and stability analysis are for arbitrary dimension and retain the higher-order derivatives of the function. From this, we are able to refine the convergence neighborhood around the desired equilibrium and obtain theoretical guarantees which are lacking in most existing derivative-free optimization methods. Code for the simulations can be found at https://github.com/tony-x-lin/susd$_s$earch.

II. Problem Formulation

Consider the optimization problem

$$\boldsymbol{x}^* = \arg\min_{\boldsymbol{x}} z(\boldsymbol{x}), \tag{1}$$

where $z : \mathbb{R}^d \to \mathbb{R}$ is a scalar objective function.

Assumption 1: The function is continuous and twice differentiable. However, the analytical forms of z, the gradient ∇z or the Hessian $\nabla^2 z$ are unknown.

As will be shown later, our algorithm only requires the function to be continuous, but our convergence proof requires the scalar objective function to be $\mathcal{C}^2$. A challenge we consider in this letter is that we consider the function z to possibly suffer from an exploding or vanishing gradient problem (EVGP). This problem occurs when the gradient $\nabla_{\boldsymbol{x}_t} z(\boldsymbol{x}_t)$ of the function $z(\boldsymbol{x}_t)$ at time t is very large for some $\boldsymbol{x}_t$ and very small for others [10]. That is $\nabla_{\boldsymbol{x}_t} z(\boldsymbol{x}_t) \approx 0$ or ∞. With such a problem, a gradient descent update $\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \lambda \nabla_{\boldsymbol{x}_t} z(\boldsymbol{x}_t)$ might be either too small to be provide progress or too large to be stable [10].

SUSD is a method developed in [12], [13] for multi-robot distributed source seeking in 2D smooth fields. Using SUSD, each agent relies only on its evaluation of the field function to modulate its speed and climb the gradient. This motivates us to use SUSD with virtual agents to solve the optimization problem (1). However, using the values of a function with an EVGP to govern the speeds of the virtual agents leads to either extremely slow performance as in the functions with vanishing gradients, or unstable performance as in the functions with exploding gradients. For example, the LQR cost function (26) is indeed infinite for an unstable policy K, and even for stable policy K it may assume very large values. Additionally, it is hard to design a termination policy when the minimum $z(\boldsymbol{x}^*)$ is extremely large and hence the virtual agents may not self terminate at the optimal.

III. The Speeding-Up and Slowing-Down (SUSD) Optimization Algorithm

Consider M virtual search agents, where each agent acts as a candidate solution $\boldsymbol{x}_i \in \mathbb{R}^d$. We need $M \geq d$, i.e., the number of search agents is at least equal to the dimension of the function. Define the covariance matrix $\boldsymbol{C} \in \mathbb{R}^{d \times d}$ as follows:

$$\boldsymbol{C} = \sum_{i=1}^{M} (\boldsymbol{x}_i - \boldsymbol{x}_c)(\boldsymbol{x}_i - \boldsymbol{x}_c)^\intercal, \tag{2}$$

where $\boldsymbol{x}_c = \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{x}_i$ is the center of the agents. Let $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d\}$ be the eigenvectors of the covariance matrix (2) associated with the eigenvalues $\{\lambda_1, \ldots, \lambda_d\}$, ordered from the smallest eigenvalue λ_1 to the largest eigenvalue λ_d. Observe that the eigenvectors of $\boldsymbol{C}$ produces the principal components of the spatial distribution of the agents.

Let the velocity of each agent be described by

$$\dot{\boldsymbol{x}}_i = f(\boldsymbol{x}_i)\boldsymbol{v}_1, \quad i = 1, \ldots, M, \tag{3}$$

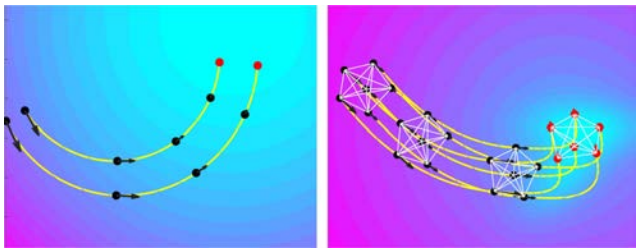

Fig. 1. Trajectories of 2-agent (left) and 6-agent (right) systems searching convex and nonconvex functions, respectively. All agents move along the SUSD direction ($\boldsymbol{v}_1$, illustrated by black arrows) with a speed proportional to the individual function values ($f(\boldsymbol{x}_i)$, illustrated by the length of the arrows). This inter-agent speed difference deforms the spatial shape which rotates the SUSD direction.

where $f : \mathbb{R} \to \mathbb{R}$ is a linear mapping defined by

$$f(\boldsymbol{x}_i) = \gamma z(\boldsymbol{x}_i) \quad i = 1, \ldots, M, \tag{4}$$

in which γ is a positive constant scalar value. Under (3), each agent speeds up or slows down along the direction $\boldsymbol{v}_1$ depending on its evaluation of the function $z(\boldsymbol{x}_i)$ at its current position $\boldsymbol{x}_i$. We call the $\boldsymbol{v}_1$ direction the SUSD direction. In Fig. 1, we demonstrate the approach for 2D scalar functions.

The Exponential Function Mapping: To handle the EVGP, we introduce the mapping

$$f(\boldsymbol{x}_i) = \gamma[1 - \exp(\bar{z} - z(\boldsymbol{x}_i))], \quad i = 1, \ldots, M, \tag{5}$$

where γ is a positive scalar value, and

$$\bar{z} = \bar{z}(t) = \min_i z(\boldsymbol{x}_i(t)) \tag{6}$$

is the instantaneous attained minimum function value among the agents. Note that $f(\boldsymbol{x}) \in [0, 1]$ where $f(\boldsymbol{x}) = 0$ if and only if $z(\boldsymbol{x}) = \bar{z}$, and $f(\boldsymbol{x}) = 1$ if and only if $z(\boldsymbol{x}) = \infty$.

The idea behind this mapping is that the SUSD direction rotates based on the inter-agent function differences. Observe that (5) preserves this inter-agent difference while it allows us to control the forward speed by the parameter γ even when $z(\boldsymbol{x}) = \infty$. Let $\tau = \{t | \bar{z} = z(\boldsymbol{x}_k)\}$ be the time interval where $\min_i z_i$ is attained by the k-th agent. Hence, in this interval, the k-th agent does not move while the remaining agents move toward the temporary target $\bar{z} = z(\boldsymbol{x}_k)$. However, and depending on γ, some agents overshoot the target where by equation (6) a new $\bar{z}$ is attained, and so on. This enforces $\bar{z}(t)$ to be decreasing and the algorithm will eventually reach close to the optimal minimum. We will show in Section V that even when the norm of the gradient explodes, using (3) with (5) allows our approach to converge to the gradient direction but with step sizes limited by γ.

A pseudocode description for the algorithm is given in Algorithm 1. Observe that we terminate the algorithm whenever there is no improvement in the attained minimum for the last W iterations.

Remark 1: Computing the SUSD direction at each iteration becomes a non-trivial operation as the number of dimensions increases. However, there exist computationally efficient methods for computing the PCA directions that can be used instead to ensure the proposed SUSD search method is inexpensive at each iteration [24], [25], [26].

Algorithm 1 SUSD Optimization Algorithm

Input: number of agents M, initials $\boldsymbol{x}_{i0}$, gain γ, termination parameters W, ϵ, and discretization constant η.
while $\bar{z}(t) - (1/W)\sum_{h=1}^{W} \bar{z}(t-h) < \epsilon$, **do**
  compute $\boldsymbol{C}(t)$ and $\boldsymbol{v}_1(t)$
  for $i = 1, \ldots, M$, **do**
    evaluate $z_i(t) = z_i(\boldsymbol{x}_i(t))$
  end for
  compute $\bar{z}(t) = \min_i z_i(t)$
  for $i = 1, \ldots, M$, **do**
    compute $f(\boldsymbol{x}_i(t)) = \gamma[1 - \exp(\bar{z}(t) - z_i(t))]$.
    update $\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \eta f(\boldsymbol{x}_i(t))\boldsymbol{v}_1(t)$.
  end for
end while
return $x^* = x_i$ such that $z_i(t) = \bar{z}(t)$.

IV. The Optimization Dynamics

In this section, we derive the dynamics of the SUSD PCA direction $\dot{\boldsymbol{v}}_1$, function value $\dot{z}$, and its gradient $\dot{\nabla} z$.

Lemma 1: Using the control law (3), the dynamics of the SUSD direction is described by

$$\dot{\boldsymbol{v}}_1 = \left(\sum_{k=2}^{d} \frac{1}{\lambda_1 - \lambda_k} \boldsymbol{v}_k \boldsymbol{v}_k^{\mathsf{T}}\right)\left(\sum_{i=1}^{M} (f_i - f_a)(\boldsymbol{x}_i - \boldsymbol{x}_c)\right), \tag{7}$$

where $\boldsymbol{v}_k$ is the k-th eigenvector of $\boldsymbol{C}$ and $f_a = \frac{1}{M}\sum_{i=1}^{M} f_i(z)$ is the average function value.

Proof: Recall that $\boldsymbol{x}_c = \frac{1}{M}\sum_{i=1}^{M} \boldsymbol{x}_i$. Hence, using (3) we obtain $\dot{\boldsymbol{x}}_c = f_a \boldsymbol{v}_1$, where $f_a = \frac{1}{M}\sum_{i=1}^{M} f_i(z)$. Taking the time derivative of (2) and using $\dot{\boldsymbol{x}}_i$ and $\dot{\boldsymbol{x}}_c$, we derive

$$\dot{\boldsymbol{C}} = \sum_{i=1}^{M} (f_i - f_a)[\boldsymbol{v}_1(\boldsymbol{x}_i - \boldsymbol{x}_c)^{\mathsf{T}} + (\boldsymbol{x}_i - \boldsymbol{x}_c)\boldsymbol{v}_1^{\mathsf{T}}]. \tag{8}$$

Moreover, by definition $\boldsymbol{C}\boldsymbol{v}_1 = \lambda_1 \boldsymbol{v}_1$. Taking the derivative, we obtain $\dot{\boldsymbol{C}}\boldsymbol{v}_1 + \boldsymbol{C}\dot{\boldsymbol{v}}_1 = \dot{\lambda}_1 \boldsymbol{v}_1 + \lambda_1 \dot{\boldsymbol{v}}_1$. Taking then inner product with the eigenvector $\boldsymbol{v}_k$, $k \neq 1$ on both sides, we obtain

$$\langle \boldsymbol{v}_k, \dot{\boldsymbol{C}}\boldsymbol{v}_1 \rangle + \langle \boldsymbol{v}_k, \boldsymbol{C}\dot{\boldsymbol{v}}_1 \rangle = \dot{\lambda}_1 \langle \boldsymbol{v}_k, \boldsymbol{v}_1 \rangle + \lambda_1 \langle \boldsymbol{v}_k, \dot{\boldsymbol{v}}_1 \rangle, \tag{9}$$

Since $\boldsymbol{C}$ is symmetric, then $\dot{\boldsymbol{C}}$ is also symmetric. This implies that $\langle \boldsymbol{v}_k, \boldsymbol{C}\dot{\boldsymbol{v}}_1 \rangle = \langle \boldsymbol{C}\boldsymbol{v}_k, \dot{\boldsymbol{v}}_1 \rangle = \lambda_k \langle \boldsymbol{v}_k, \dot{\boldsymbol{v}}_1 \rangle$. Using this along with the fact that $\langle \boldsymbol{v}_k, \boldsymbol{v}_1 \rangle = \langle \boldsymbol{v}_1, \boldsymbol{v}_k \rangle = 0$, we obtain from (9)

$$\langle \boldsymbol{v}_k, \dot{\boldsymbol{v}}_1 \rangle = -\frac{1}{\lambda_k - \lambda_1} \langle \boldsymbol{v}_k, \dot{\boldsymbol{C}}\boldsymbol{v}_1 \rangle \tag{10}$$

Since C is symmetric, one can always find a complete set of orthogonal eigenvectors $\{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_d\}$. Therefore, we may write

$$\dot{\boldsymbol{v}}_1 = \sum_{k=2}^{d} \langle \boldsymbol{v}_k, \dot{\boldsymbol{v}}_1 \rangle \boldsymbol{v}_k. \tag{11}$$

Substituting (8) in (10), and using (11), along with the fact that $\langle \boldsymbol{v}_k, \boldsymbol{v}_1 \rangle \langle \boldsymbol{x}_i - \boldsymbol{x}_c, \boldsymbol{v}_1 \rangle = 0$ yields the desired result. ■

Let $f_c = f(z(\boldsymbol{x}_c))$ and define the gradient $\nabla f = \nabla f(\boldsymbol{x}_c)$. Then we approximate $f_i = f(\boldsymbol{x}_i)$ using Taylor expansion with respect to the center $\boldsymbol{x}_c$, as follows

$$f_i - f_c = \langle \boldsymbol{x}_i - \boldsymbol{x}_c, \nabla f \rangle + \omega_i, \tag{12}$$

where $f_c = f(\boldsymbol{x}_c)$ and $\omega_i = \mathcal{O}(\|\boldsymbol{x}_i - \boldsymbol{x}_c\|)$ is the sum of remaining higher order components of the function.

Lemma 2: Using the control law (3), the dynamics of the SUSD direction is described by

$$\dot{\boldsymbol{v}}_1 = -\sum_{k=2}^{d} \frac{\lambda_k}{\lambda_k - \lambda_1} \boldsymbol{v}_k \boldsymbol{v}_k^\intercal \nabla f + \boldsymbol{\omega}, \tag{13}$$

$$\boldsymbol{\omega} = -\left(\sum_{k=2}^{d} \frac{1}{\lambda_k - \lambda_1} \boldsymbol{v}_k \boldsymbol{v}_k^\intercal\right)\left(\sum_{i=1}^{M} \omega_i (\boldsymbol{x}_i - \boldsymbol{x}_c)\right). \tag{14}$$

Proof: Let $\omega_a = \frac{1}{M}\sum_{i=1}^{M} \omega_i$ which implies from (12) that $f_a = f_c + \omega_a$. Using this, multiplying both sides of (12) by $(\boldsymbol{x}_i - \boldsymbol{x}_c)$, and summing over all i, we derive

$$\sum_{i=1}^{M} (f_i - f_a)(\boldsymbol{x}_i - \boldsymbol{x}_c) = \boldsymbol{C}\nabla f + \sum_{i=1}^{M} \omega_i (\boldsymbol{x}_i - \boldsymbol{x}_c). \tag{15}$$

Substituting (15) into (7), and using the fact that $\boldsymbol{v}_k^\intercal \boldsymbol{C} = \lambda_k \boldsymbol{v}_k^\intercal$, yields (13). ∎

Denote by $z_c = z(\boldsymbol{x}_c)$ the z function value at the center, $\boldsymbol{x}_c$. Also, let $\nabla z = \nabla z(\boldsymbol{x}_c)$ be the gradient of z function at the center, $\boldsymbol{x}_c$.

Lemma 3: Using the control law (3) along with the linear mapping (4) yields the search dynamics

$$\dot{z}_c = \gamma z_a \langle \nabla z, \boldsymbol{v}_1 \rangle, \quad z_a = \frac{1}{M}\sum_{i=1}^{M} z(\boldsymbol{x}_i(t)), \tag{16}$$

$$\dot{\boldsymbol{v}}_1 = -\gamma \sum_{k=2}^{d} \frac{\lambda_k}{\lambda_k - \lambda_1} \boldsymbol{v}_k \boldsymbol{v}_k^\intercal \nabla z + \boldsymbol{\omega}. \tag{17}$$

Proof: We have $\dot{z}_c = \dot{z}(\boldsymbol{x}_c) = \langle \nabla z(\boldsymbol{x}_c), \dot{\boldsymbol{x}}_c \rangle$. But $\dot{\boldsymbol{x}}_c = \frac{1}{M}\sum_{i=1}^{M} f(\boldsymbol{x}_i)\boldsymbol{v}_1 = \gamma z_a \boldsymbol{v}_1$, which proves (16). On the other hand, $f(\boldsymbol{x}_c) = \gamma z(\boldsymbol{x}_c)$ implies that $\nabla f(\boldsymbol{x}_c) = \frac{df}{dz}\nabla z(\boldsymbol{x}_c) = \gamma \nabla z(\boldsymbol{x}_c)$. Finally, substituting $\gamma \nabla z(\boldsymbol{x}_c)$ for $\nabla f(\boldsymbol{x}_c)$ in (13) completes the proof of (17). ∎

Lemma 4: Using the control law (3) along with exponential mapping (5) yields the search dynamics

$$\dot{z}_c = \frac{\gamma}{M}\sum_{i=1}^{M} [1 - \exp(\bar{z} - z_i)] \langle \nabla z, \boldsymbol{v}_1 \rangle, \tag{18}$$

$$\dot{\boldsymbol{v}}_1 = -\gamma \exp(\bar{z} - z_c) \sum_{k=2}^{d} \frac{\lambda_k}{\lambda_k - \lambda_1} \boldsymbol{v}_k \boldsymbol{v}_k^\intercal \nabla z + \boldsymbol{\omega}. \tag{19}$$

Proof: We first derive $\dot{\boldsymbol{x}}_c = \frac{1}{M}\sum_{i=1}^{M} f(\boldsymbol{x}_i)\boldsymbol{v}_1 = \frac{\gamma}{M}\sum_{i=1}^{M} [1 - \exp(\bar{z} - z(\boldsymbol{x}_i))]\boldsymbol{v}_1$. Substituting this in $\dot{z}(\boldsymbol{x}_c) = \langle \nabla z(\boldsymbol{x}_c), \dot{\boldsymbol{x}}_c \rangle$ proves (18). On the other hand, $\nabla f(\boldsymbol{x}_c) = \frac{df}{dz}\nabla z(\boldsymbol{x}_c) = \gamma \exp(\bar{z} - z(\boldsymbol{x}_c)\nabla z(\boldsymbol{x}_c)$. Therefore, substituting this for $\nabla f(\boldsymbol{x}_c)$ in (13) completes the proof of (19). ∎

Define $\boldsymbol{N} = \frac{\nabla z}{\|\nabla z\|}$ to be a unit-length vector in the direction of the gradient. Let $\boldsymbol{H} = \nabla^2 z(\boldsymbol{x}_c)$ be the Hessian matrix of the z function at the center, $\boldsymbol{x}_c$.

Lemma 5: Suppose the agents are moving according to the control law (3). Then, if we apply the linear mapping (4), the gradient at the center changes according to

$$\dot{\boldsymbol{N}} = \frac{\gamma z_a}{\|\nabla z\|}(\boldsymbol{I} - \boldsymbol{N}\boldsymbol{N}^\intercal)\boldsymbol{H}\boldsymbol{v}_1, \quad z_a = \frac{1}{M}\sum_{i=1}^{M} z(\boldsymbol{x}_i). \tag{20}$$

If instead, we apply the exponential mapping (5). Then gradient at the center changes according to

$$\dot{\boldsymbol{N}} = \frac{\gamma/M}{\|\nabla z\|}\sum_{i=1}^{M} [1 - \exp(\bar{z} - z_i)](\boldsymbol{I} - \boldsymbol{N}\boldsymbol{N}^\intercal)\boldsymbol{H}\boldsymbol{v}_1. \tag{21}$$

Proof: We first write $\dot{\boldsymbol{N}} = \frac{d}{dt}(\frac{1}{\|\nabla z\|})\nabla z + \frac{1}{\|\nabla z\|}\frac{d}{dt}(\nabla z)$. But $\frac{d}{dt}(\nabla z) = \boldsymbol{H}\dot{\boldsymbol{x}}_c$, where $\boldsymbol{H}$ is the Hessian matrix. On the other hand, $\frac{d}{dt}(\frac{1}{\|\nabla z\|}) = -\|\nabla z\|^{-3}(\nabla z)^\intercal \frac{d}{dt}(\nabla z)$. Consequently, we obtain $\dot{\boldsymbol{N}} = \frac{1}{\|\nabla z\|}(\boldsymbol{I} - \boldsymbol{v}_1\boldsymbol{v}_1^\intercal)\boldsymbol{H}\dot{\boldsymbol{x}}_c$. Substituting $\dot{\boldsymbol{x}}_c = \gamma z_a \boldsymbol{v}_1$ for the linear mapping yields the desired result (20). Similarly, substituting $\dot{\boldsymbol{x}}_c = \frac{\gamma}{M}\sum_{i=1}^{M}[1 - \exp(\bar{z} - z(\boldsymbol{x}_i))]\boldsymbol{v}_1$ for the exponential mapping yields the desired result (21). ∎

V. The Convergence Analysis

We now analyze the convergence of the SUSD direction $\boldsymbol{v}_1$ to the negative direction of the gradient $-\boldsymbol{N}$. Define $\theta = 1 + \langle \boldsymbol{v}_1, \boldsymbol{N} \rangle$, where $\theta = 0$ if and only if $\boldsymbol{v}_1 = -\boldsymbol{N}$. Then, using (17) and (19), we obtain

$$\dot{\theta} = \gamma\sigma\|\nabla z\| \sum_{k=2}^{d} \frac{\lambda_k}{\lambda_1 - \lambda_k} \langle \boldsymbol{N}, \boldsymbol{v}_k \rangle^2 + \delta = h(t, \theta, \delta), \tag{22}$$

where either $\sigma = 1$ for the linear mapping or $\sigma = \exp(\bar{z} - z_c)$ for the exponential mapping. Additionally, $\delta = \langle \boldsymbol{N}, \boldsymbol{\omega} \rangle + \langle \dot{\boldsymbol{N}}, \boldsymbol{v}_1 \rangle$ depends on the higher order components of the function. We will view δ as an external disturbance to the state θ due to the effect of the nonlinearity of the function which cannot be controlled by the swarm. Since $\sum_{k=2}^{d} \lambda_k/(\lambda_1 - \lambda_k)\langle \boldsymbol{N}, \boldsymbol{v}_k \rangle^2 = 0$ if and only if $\langle \boldsymbol{N}, \boldsymbol{v}_k \rangle = 0$, $\forall k \neq 1$, then $h(t, \theta, 0) = 0$ if and only if $\boldsymbol{v}_1 = \pm\boldsymbol{N}$, i.e., $\theta = 0$ or $\theta = 2$. Note that when $\boldsymbol{v}_1 = \pm\boldsymbol{N}$, then using (20) and (21), $\langle \dot{\boldsymbol{N}}, \boldsymbol{v}_1 \rangle = \pm\langle \dot{\boldsymbol{N}}, \boldsymbol{N} \rangle = 0$. Similarly, when $\boldsymbol{v}_1 = \pm\boldsymbol{N}$, then using (14), $\langle \boldsymbol{N}, \boldsymbol{\omega} \rangle = 0$. This implies that the perturbation δ vanishes at the equilibrium points $\theta = 0$ or $\theta = 2$.

Let $\epsilon \in (0, 1)$ and $\bar{\omega} = \sum_{i=1}^{M} \omega_i (\boldsymbol{x}_i - \boldsymbol{x}_c)$. Define

$$\mu = \frac{(d-1)}{2\lambda_1\sigma\epsilon}\frac{\lambda_d - \lambda_1}{\lambda_2 - \lambda_1} \cdot \left(\|\bar{\omega}\| + \sqrt{\|\bar{\omega}\|^2 + 4\frac{\lambda_1\sigma\epsilon\zeta\|\boldsymbol{H}\|(\lambda_2 - \lambda_1)^2}{(\lambda_d - \lambda_1)(d-1)^2}}\right), \tag{23}$$

where either $\zeta = \frac{1}{M}\sum_{i=1}^{M} z_i$ for the linear mapping, or $\zeta = \frac{1}{M}\sum_{i=1}^{M}[1 - \exp(\bar{z} - z_i)]$ for the exponential mapping. Recall that d is the dimension of the function and λ_d and λ_1 are the largest and smallest eigenvalues, respectively. Additionally, $\boldsymbol{H}$ is the Hessian matrix of the function.

Theorem 1: Consider (22) and suppose $\|\nabla z(\boldsymbol{x}_c)\| > \mu$ where μ is as defined by (23). Then the equilibrium $\theta = 0$ of the unforced system $\dot{\theta} = h(t, \theta, 0)$ is asymptotically stable in which whenever $\theta(0) \in [0, 2)$, then $\theta(t) \to 0$ as $t \to \infty$. Furthermore, for an input disturbance satisfying $|\delta| < \gamma\sigma\epsilon\frac{\lambda_1}{\lambda_d - \lambda_1}\mu$ for some $\epsilon \in (0, 1)$, the equilibrium $\theta = 0$ of forced system $h(t, \theta, \delta)$ is locally input-to-state stable.

Proof: Define $D = [0, 2)$. Let $V : D \to \mathbb{R}$ be a Lyapunov candidate give by $V = \theta/(2 - \theta)$, where $V = 0$ if and only if $\theta = 0$ and $V \to \infty$ as $\theta \to 2$. Then, when $\delta = 0$, we obtain

$$\dot{V} \leq -2\gamma\sigma\|\nabla z\|\frac{\lambda_1}{\lambda_d - \lambda_1} V, \tag{24}$$

Since $\dot{V} = 0$ if and only if $\theta = 0$, then the origin of the unforced system $h(t, \theta, 0)$ is asymptotically stable. Additionally, since $\dot{V} \to -\infty$ as $\theta \to 2$, and the fact that $V \to \infty$ whenever $\theta \to 2$ and $\|\nabla z\| > \mu > 0$, implies that $\mathbf{D}$ is a forward invariant set, and thus $\theta \in [0, 2)$ for all t. For the forced system $h(t, \theta, \delta)$, we obtain

$$\dot{V} \leq -2(1-\epsilon)\gamma\sigma\|\nabla z\|V, \quad \forall |\theta| > \rho(|\delta|), \tag{25}$$

where $\rho(|\delta|) = 1 - \sqrt{1 - \frac{\lambda_d - \lambda_1}{\lambda_1}\frac{|\delta|}{\epsilon\gamma\sigma\|\nabla z\|}}$ is a class $\mathcal{K}$ function. Since it is assumed that $|\delta| < \gamma\sigma\epsilon\frac{\lambda_1}{\lambda_d - \lambda_1}\mu$, then the set $\theta \in [0, \rho(|\delta|))$ is not empty. Let $\alpha_1(|\theta|) = \alpha_2(|\theta|) = \frac{|\theta|}{2-|\theta|}$ which are class $\mathcal{K}$ functions that satisfy $\alpha_1(|\theta|) \leq V(\theta) \leq \alpha_2(|\theta|)$. Therefore, using *Definition 3.3* of local input-to-state stability in [27], and according to [28, Th. 4.19], the origin of the forced system $h(t, \theta, \delta)$ is locally input-to-state stable. We used the local ISS definition of [27] since the state θ and disturbance δ are defined in local domains, i.e., $\theta \in [0, 2)$ and $|\delta| < \gamma\sigma\epsilon\frac{\lambda_1}{\lambda_d - \lambda_1}\mu$. ■

From (16) and (18), we see that z_c is decreasing until entering the set $\|\nabla z\| < \mu$. For the linear mapping, once $\|\nabla z\| < \mu$, z_c oscillates especially when z^* is large. However, for the exponential mapping, although z_c might increase when $\|\nabla z\| < \mu$, the agent with $z_i = \bar{z}$ becomes the anchor of the swarm where the other agents oscillate around it until another agent attains a smaller $\bar{z}$.

Remark 2: The bound μ in (23) describes a region of attraction for the origin (22). That is, the SUSD direction, $\boldsymbol{v}_1$, is attracted to the negative gradient, $-\boldsymbol{N}$, whenever $\|\nabla z\|$ dominates $\|\omega\|$. Interestingly, (23) suggests that μ decreases when the eigenvalues are close to each other, i.e., when the agents are radially distributed. Note that in (23), $\zeta \in [0, 1]$ for the exponential mapping, while $\zeta \in [0, \infty)$ for the linear mapping. This implies that when the minimum $z(\boldsymbol{x}^*) > 1$, then the region of attraction of the exponential mapping is larger than that of the linear mapping.

Remark 3: The ISS result in Theorem 1 may not hold for an arbitrary discretization constant η when using the discrete SUSD control law $\boldsymbol{x}_i(t+1) = \boldsymbol{x}_i(t) + \eta f(\boldsymbol{x}_i(t))\boldsymbol{v}_1(t)$. By making η small enough we can derive similar ISS result using techniques from [29], which we leave for future work.

VI. Simulations

The proposed method is applied first to a data-driven LQR problem where the LQR gains are estimated from only system trajectories, and then to a classical nonconvex Rosenbrock optimization problem, where the function may also be perturbed by various levels of noise.

A. Data-Driven LQR

Consider the linear system $\boldsymbol{\xi}_{t+1} = \boldsymbol{A}\boldsymbol{\xi}_t + \boldsymbol{B}\boldsymbol{u}_t$, where $\boldsymbol{\xi} \in \mathbb{R}^n$ is the state vector, $\boldsymbol{u} \in \mathbb{R}^m$ is the input vector, and ($\boldsymbol{A} \in \mathbb{R}^{n\times n}$, $\boldsymbol{B} \in \mathbb{R}^{n\times m}$) are the system matrices. Consider the feedback control law $\boldsymbol{u}_t = -\boldsymbol{K}\boldsymbol{\xi}_t$. Our objective is to find $\boldsymbol{K}^*$ that optimizes the LQR cost

$$z(\boldsymbol{K}) = \boldsymbol{\xi}_T^{\mathsf{T}}\boldsymbol{Q}\boldsymbol{\xi}_T + \sum_{t=0}^{T-1}\boldsymbol{\xi}_t^{\mathsf{T}}\left(\boldsymbol{Q} + \boldsymbol{K}^{\mathsf{T}}\boldsymbol{R}\boldsymbol{K}\right)\boldsymbol{\xi}_t, \tag{26}$$

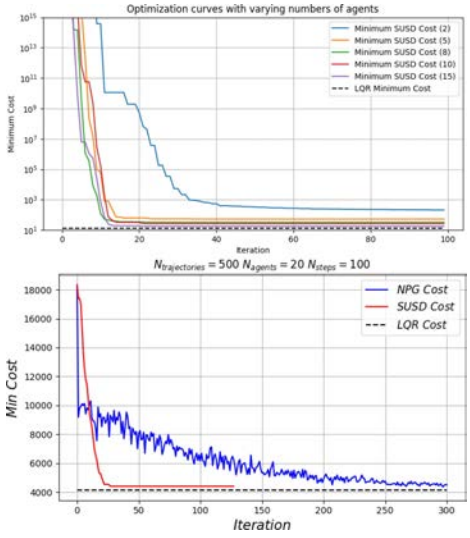

Fig. 2. The optimization performance of the SUSD method over each iteration (note y-axis is on a log-scale) with varying numbers of agents (top) and compared with the NPG (bottom).

using only state-input trajectories of the system $\{(\boldsymbol{\xi}_t, \boldsymbol{u}_t)\}_{t=0}^{\mathsf{T}}$ where $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ and $\boldsymbol{R} \in \mathbb{R}^{m\times m}$ are positive definite matrices. In general, the LQR cost (26) is nonconvex [11]. To apply Algorithm 1, we first reshape $\boldsymbol{K}$ into a vector $\boldsymbol{k} \in \mathbb{R}^{mn}$, i.e., $\boldsymbol{k} = [(1^{st}\text{ column of }\boldsymbol{k})^{\mathsf{T}}, \ldots, (m^{th}\text{ column of }\boldsymbol{k})^{\mathsf{T}}]^{\mathsf{T}}$. Then we randomly generate policies $\boldsymbol{k}_{i,0}$ where $i = 1, \ldots, M$ and $M \geq mn$. Then each agent applies the SUSD control law (3) where $\boldsymbol{x}_i$ is replaced by $\boldsymbol{k}_i$. We consider an unstable system where the NPG search method is unable to find the LQR gains due to an exploding gradient, and a high-dimensional asymptotically stable system ($K \in \mathbb{R}^{3\times 4}$, i.e., $\boldsymbol{k} \in \mathbb{R}^{12}$) in order to compare with the NPG search algorithm (as in [11]). The parameters for both systems are given by

$$A_1 = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}, A_2 = \begin{bmatrix} -2.5 & 1.2 & 4.3 & 0.1 \\ 0.97 & -10.3 & 0.4 & -6.1 \\ -9.2 & 1.1 & -4.9 & 0.3 \\ 1.1 & 0.9 & -3.4 & -0.9 \end{bmatrix}$$

and

$$B_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, B_2 = \begin{bmatrix} 1.1 & 0.4 & -0.2 \\ -3.2 & 1.4 & 0 \\ -0.8 & 0.1 & 3.0 \\ -1.1 & -0.9 & 5.2 \end{bmatrix}.$$

Fig. 2, top, demonstrates the performance when each agent's initial estimate starts near $K_0 = \begin{bmatrix} 0.1 & 0.1 \end{bmatrix}$ and induces an unstable closed-loop system. Note that SUSD minimizes the cost of the current best estimate of K with significant improvements occurring as the number of agents increases. The SUSD estimated LQR gains with 8 agents ($K_{SUSD} = \begin{bmatrix} -6.3724 & -3.6834 \end{bmatrix}$) aligns closely with the true LQR gains ($K_{LQR} = \begin{bmatrix} -6.4641 & -3.7321 \end{bmatrix}$). While the high sampled cost of an unstable policy may lead to numerical issues (such as a floating point overflow), the matrices Q and R may be scaled to ensure large costs are able to be stored in memory.

In Fig. 2 bottom, while NPG requires 500 trajectory samples per iteration, SUSD achieves higher accuracy with only

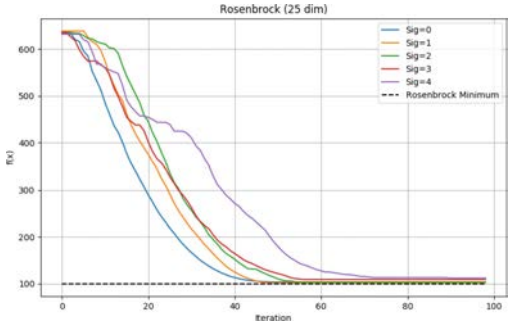

Fig. 3. Optimization performance using the Rosenbrock function.

20 trajectories per iteration. The NPG's step size was hand-tuned to 0.8 to achieve the fastest stable step size while the SUSD gain γ was hand-tuned to 0.1 to achieve the fastest convergence rate.

B. Rosenbrock Optimization

The Rosenbrock function is a nonconvex test function where the global minimum is inside a long and narrow flat valley of a vanishingly small gradient. For $\boldsymbol{x} \in \mathbb{R}^{25}$,

$$z(\boldsymbol{x}) = 100 + \sum_{i=1}^{24} \Big[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \Big]. \quad (27)$$

where the minimum is $z(\boldsymbol{x}^*) = 100$. Fig. 3 shows that our proposed approach achieves fast convergence and high accuracy with $f(\boldsymbol{x}_{SUSD}) = 101.83$. In addition, when the evaluation is perturbed by noise drawn from i.i.d. $\mathcal{N}(0, \Sigma)$ with various Σ values, we are still able to achieve $f(\boldsymbol{x}_{SUSD}) = 112.15$ when $\Sigma = 4I_{25}$. The cost shown in Fig. 3 is the true function evaluation of the current best estimate of $\boldsymbol{x}^*$ without noise.

VII. Conclusion

In this letter, we proposed a derivative-free optimization solver that leveraged a novel exponential mapping to handle optimization problems with EVGP. The proposed method is supported by a convergence analysis that shows our approach is able to converge to the gradient direction. Our approach is further supported by a data-driven LQR problem, in which our method is able to approximate the LQR gains for unstable systems, and the optimization of the Rosenbrock function, in which our method is able to find good approximations of the solution, even while under noisy perturbations. In future work, we will explore alternative function mappings that may extend our approach to other optimization problems that may be difficult to solve using conventional optimization techniques.

References

[1] A. R Conn, K. Scheinberg, and P. L. Toint, "Recent progress in unconstrained nonlinear optimization without derivatives," *Math. Program.*, vol. 79, nos. 1–3, p. 397, 1997.

[2] J. Larson, M. Menickelly, and S. M. Wild, "Derivative-free optimization methods," *Acta Numerica*, vol. 28, pp. 287–404, May 2019.

[3] S. Bhasin, M. Johnson, and W. E. Dixon, "A model-free robust policy iteration algorithm for optimal control of nonlinear systems," in *Proc. 49th IEEE Conf. Decis. Control (CDC)*, Atlanta, GA, USA, 2010, pp. 3060–3065.

[4] D. Malik, A. Pananjady, K. Bhatia, K. Khamaru, P. L. Bartlett, and M. J. Wainwright, "Derivative-free methods for policy optimization: Guarantees for linear quadratic systems," *J. Mach. Learn. Res.*, vol. 21, no. 21, pp. 1–51, 2020.

[5] Y. Abbasi-Yadkori, N. Lazic, and C. Szepesvari, "Model-free linear quadratic control via reduction to expert prediction," in *Proc. Mach. Learn. Res.*, vol. 89, pp. 3108–3117, Apr. 2019.

[6] S. Bhasin, R. Kamalapurkar, M. Johnson, K. G. Vamvoudakis, F. L. Lewis, and W. E. Dixon, "A novel actor–critic–identifier architecture for approximate optimal control of uncertain nonlinear systems," *Automatica*, vol. 49, no. 1, pp. 82–92, 2013.

[7] J. A. Carrillo, S. Jin, L. Li, and Y. Zhu, "A consensus-based global optimization method for high dimensional machine learning problems," 2019. [Online]. Available: arXiv:1909.09249.

[8] Y. Nesterov and V. Spokoiny, "Random gradient-free minimization of convex functions," *Found. Comput. Math.*, vol. 17, no. 2, pp. 527–566, 2017.

[9] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[10] B. Hanin. "Which neural net architectures give rise to exploding and vanishing gradients?" in *Advances in Neural Information Processing Systems*. Red Hook, NY, USA: Curran Assoc., Inc., 2018, pp. 582–591.

[11] M. Fazel, R. Ge, S. Kakade, and M. Mesbahi, "Global convergence of policy gradient methods for the linear quadratic regulator," in *Proc. 35th Int. Conf. Mach. Learn.*, vol. 80, 2018, pp. 1466–1475.

[12] W. Wu and F. Zhang, "A speeding-up and slowing-down strategy for distributed source seeking with robustness analysis," *IEEE Trans. Control Netw. Syst.*, vol. 3, no. 3, pp. 231–240, Sep. 2016.

[13] S. Al-Abri, S. Maxon, and F. Zhang, "Integrating a PCA learning algorithm with the SUSD strategy for a collective source seeking behavior," in *Proc. IEEE Annu. Amer. Control Conf. (ACC)*, Milwaukee, WI, USA, 2018, pp. 2479–2484.

[14] J. C. Lagarias, J. A. Reeds, M. H. Wright, and P. E. Wright, "Convergence properties of the Nelder–Mead simplex method in low dimensions," *SIAM J. Optim.*, vol. 9, no. 1, pp. 112–147, 1998.

[15] K. I. McKinnon, "Convergence of the Nelder–Mead simplex method to a nonstationary point," *SIAM J. Optim.*, vol. 9, no. 1, pp. 148–158, 1998.

[16] K. B. Ariyur and M. Krstic, *Real-Time Optimization by Extremum-Seeking Control*. Hoboken, NJ. USA: Wiley, 2003.

[17] C. Zhang and R. Ordóñez, *Extremum-Seeking Control and Applications: A Numerical Optimization-Based Approach*. London, U.K.: Springer, 2011.

[18] Y. Zhang, M. Rotea, and N. Gans, "Sensors searching for interesting things: Extremum seeking control on entropy maps," in *Proc. 50th IEEE Conf. Decis. Control Eur. Control Conf.*, Orlando, FL, USA, 2011, pp. 4985–4991.

[19] S. Z. Khong, Y. Tan, C. Manzie, and D. Nešić, "Multi-agent source seeking via discrete-time extremum seeking control," *Automatica*, vol. 50, no. 9, pp. 2312–2320, 2014.

[20] B. Calli, W. Caarls, P. Jonker, and M. Wisse, "Comparison of extremum seeking control algorithms for robotic applications," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Vilamoura, Portugal, 2012, pp. 3195–3202.

[21] J. Feiling, A. Zeller, and C. Ebenbauer, "Derivative-free optimization algorithms based on non-commutative maps," *IEEE Control Syst. Lett.*, vol. 2, no. 4, pp. 743–748, Oct. 2018.

[22] A. R. Conn, N. I. M. Gould, and P. L. Toint, *Trust Region Methods*, vol. 1. Philadelphia, PA, USA: Soc. Ind. Appl. Math., 2000.

[23] C. De Persis and P. Tesi, "Formulas for data-driven control: Stabilization, optimality, and robustness," *IEEE Trans. Autom. Control*, vol. 65, no. 3, pp. 909–924, Mar. 2020.

[24] E. Oja, "Simplified neuron model as a principal component analyzer," *J. Math. Biol.*, vol. 15, no. 3, pp. 267–273, 1982.

[25] A. Sharma and K. K. Paliwal, "Fast principal component analysis using fixed-point algorithm," *Pattern Recognit. Lett.*, vol. 28, no. 10, pp. 1151–1155, 2007.

[26] M. Tao and T. Ohsawa, "Variational optimization on lie groups, with examples of leading (generalized) eigenvalue problems," in *Proc. Int. Conf. Artif. Intell. Stat. (AISTATS)*, 2020, pp. 4269–4280.

[27] S. Dashkovskiy, D. V. Efimov, and E. D. Sontag, "Input to state stability and allied system properties," *Autom. Remote Control*, vol. 72, no. 8, p. 1579, 2011.

[28] H. K. Khalil. *Nonlinear Systems*. Upper Saddle River, NJ, USA: Prentice-Hall, 2002.

[29] Z.-P. Jiang and Y. Wang, "Input-to-state stability for discrete-time nonlinear systems," *Automatica*, vol. 37, no. 6, pp. 857–869, 2001.