System Identification and Control Using Quadratic Neural Networks

Luis Rodrigues[1] and Sidney Givigi[2]
[1]Department of Electrical and Computer Engineering, Concordia University, Canada
[2] School of Computing, Queens University, Canada

Abstract— **This paper addresses the analysis and design of quadratic neural networks, which have been recently introduced in the literature, and their applications to system identification and control of dynamical systems. These networks offer several advantages, the most important of which are the fact that the architecture is a by-product of the design and is not determined a-priori, their training can be done by solving a convex optimization problem so that the global optimum of the weights is achieved, and the input-output mapping can be expressed analytically by a quadratic form. It also appears from several examples that these networks work extremely well using only a small fraction of the training data. The results in the paper cast system identification, stability and control design as convex optimization problems, which can be solved efficiently with polynomial-time algorithms to a global optimum. Several examples will show the effectiveness of quadratic neural networks in applications.**

I. INTRODUCTION

The first mathematical model of an artificial neuron was proposed in 1943 by McCulloch and Pitts [1]. Since then, artificial neural networks have been an active area of research with applications such as image recognition, natural language processing, and signal processing to name a few. However, their widespread use did not yet reach the same proportion in safety-critical applications of control systems, such as autonomous vehicles, because of the lack of theoretical guarantees on safety, stability, and performance, which are of prime concern for such applications. In particular, formal results on Lyapunov stability of a system in feedback with a neural network controller are very scarce and only started appearing very recently in the literature [2]. Another current issue with most artificial neural networks is that before the training of the network can be performed one must decide on the network architecture. This is most often a difficult trial-and-error task that is heavily dependent on the application, although upper bounds on the number of neurons in feedforward networks needed to learn a given number of data were determined in reference [3]. Architectures in which the activation functions are different in different layers have been proposed for efficient approximation of high dimensional functions in [4]. The architecture redundancy for pixel level estimation was recently addressed in reference [5]. Once an architecture has been chosen, another important issue is that the training of the neural network weights does not usually guarantee that the global optimum value is achieved. An additional difficulty in the analysis of neural networks is the fact that the mapping between the input and output cannot be written as a concise analytical expression for a general input. This makes it difficult to estimate the robustness of the output of the neural network for small changes in the input, which is typically done by computing a Lipschitz continuity constant [6], [7]. Moreover, the best current results of training neural networks using deep learning [8] typically require an extremely large amount of data, so-called *big-data*. One of the first papers to address convex neural networks was reference [9]. More recently, the training of quadratic neural networks (QNN) was shown in reference [10] to be a convex optimization program with guarantees of achieving the global optimum value of the weights. Additionally, quadratic neural networks also offer advantages relative to many other issues mentioned in the previous paragraph. In particular, the input and output are related by a quadratic form and the architecture of the network is a by-product of the training itself. This comes at the expense of only having one hidden layer, although extensions to more hidden layers have been proposed in reference [10].

The main focus of this paper is to perform a theoretical analysis and to provide several simulation results for quadratic neural networks applied to regression, classification, system identification, and control problems. The paper is organized as follows. Section II will review quadratic neural networks and provide three results on their relationship with quadratic forms. System identification is addressed in section III, followed by control design in section IV. After section V on examples the paper ends with the conclusions.

II. QUADRATIC NEURAL NETWORKS

The quadratic neural network proposed in [10] is constrained to have a single hidden layer with $M = \sum_{k=1}^{p} M_k$ neurons. Each output is connected to M_k neurons in the hidden layer. The value of each network output is obtained by a linear combination of the outputs of the hidden layer neurons. Each output $k = 1, \ldots, p$, is

$$\hat{y}^k(x) = \hat{f}^k(x) = \sum_{j=1}^{M_k} \sigma\left(x^T w^{k,j}\right) \alpha_j^k \tag{1}$$

where the activation function is quadratic and is written as

$$\sigma(z) = az^2 + bz + c \tag{2}$$

where $a \neq 0, b, c$, are pre-defined constants that parameterize the quadratic activation function. The weights w connect the input $x \in I\!R^n$ to each neuron in the hidden layer. The desired (label) outputs will be denoted by y and the actual outputs of the network will be denoted by $\hat{y}$. Following [10] it will be assumed that the weights $w^{k,j}$ are normalized to have unit norm. Using a convex loss function $l(\cdot)$, the original (primal) non-convex training problem for a quadratic network where all hidden neurons are connected to all outputs (using all weights w^j instead of $w^{k,j}$ for each output $\hat{y}^k$) is [10]

$$\begin{aligned} &\min_{w^j,\alpha_j} l(\hat{y}-y)+\beta\textstyle\sum_{i=1}^{M}\|\alpha_i\|_1 \\ &\text{s.t. } \hat{y}^k=\textstyle\sum_{j=1}^{M}\sigma\left(x^T w^j\right)\alpha_j^k, \\ &\quad\|w^j\|_2=1,\ k=1,\ldots,p,\ j=1,\ldots,M, \end{aligned} \tag{3}$$

for fixed $a\neq 0, b, c$, and a fixed regularization coefficient $\beta\geq 0$. The following result from reference [10] recasts the training as an equivalent convex optimization problem.

Lemma 1: [10] Given fixed $a\neq 0, b, c$, and a fixed regularization coefficient $\beta\geq 0$, the solution of the convex problem that is dual to (3) and is formulated as

$$\begin{aligned} &\min l(\hat{y}-y)+\beta\textstyle\sum_{k=1}^{p}\left(Z_+^{k,4}+Z_-^{k,4}\right) \\ &\text{s.t.} \\ &\hat{y}_i^k=\bar{x}_i^T\begin{bmatrix} a\left(Z_+^{k,1}-Z_-^{k,1}\right) & \frac{b}{2}\left(Z_+^{k,2}-Z_-^{k,2}\right) \\ \frac{b}{2}\left(Z_+^{k,2}-Z_-^{k,2}\right)^T & c\mathbf{Trace}\left(Z_+^{k,1}-Z_-^{k,1}\right)\end{bmatrix}\bar{x}_i \\ &Z_+^{k,4}=\mathbf{Trace}\left(Z_+^{k,1}\right), Z_-^{k,4}=\mathbf{Trace}\left(Z_-^{k,1}\right), \\ &Z_+^k=\begin{bmatrix} Z_+^{k,1} & Z_+^{k,2} \\ \left(Z_+^{k,2}\right)^T & Z_+^{k,4}\end{bmatrix},\ Z_-^k=\begin{bmatrix} Z_-^{k,1} & Z_-^{k,2} \\ \left(Z_-^{k,2}\right)^T & Z_-^{k,4}\end{bmatrix}, \\ &Z_+^k\geq 0,\ Z_-^k\geq 0,\ \bar{x}_i^T=[x_i^T\quad 1],\ k=1,\ldots,p,\ i=1,\ldots N, \end{aligned} \tag{4}$$

where $l(\cdot)$ is a convex loss function, provides a global optimal solution for the parameters $Z_+^k, Z_-^k\in I\!R^{(n+1)\times(n+1)}$, for $k=1,\ldots,p$, when $M\geq M_*$ with

$$M_*=\sum_{k=1}^{p}\left[rank\left(Z_+^{k*}\right)+rank\left(Z_-^{k*}\right)\right], \tag{5}$$

where Z_+^{k*} and Z_-^{k*} for $k=1,\ldots,p$, are the solution of the optimization problem (4) given N input data vectors $x_i\in I\!R^n$ with corresponding labels $y_i\in I\!R^p$. Moreover, the optimal value of the solutions of problems (3) and (4) are the same and therefore the duality gap is zero. □

Instead of describing the neural network by its weights, we use the quadratic form (6), which is equivalent to the quadratic form in (4) and to the expression (1) (see [10])

$$\hat{y}^k=\hat{f}^k(x)=\bar{x}^T\begin{bmatrix} aZ_1^k & \frac{b}{2}Z_2^k \\ \frac{b}{2}\left(Z_2^k\right)^T & cZ_4^k\end{bmatrix}\bar{x}=\bar{x}\bar{Z}^k\bar{x} \tag{6}$$

where $\bar{x}=[x^T\quad 1]^T$.

Lemma 2: [11] Given a symmetric matrix P and a random vector x_* with mean μ and covariance matrix Σ,

$$E\left[x_*^T P x_*\right]=\mu^T P\mu+\mathbf{Trace}\left(P\Sigma\right). \tag{7}$$

Proof: See Appendix B of [11]. □

Theorem 1: Let $\bar{x}=[x^T\ 1]^T$ with $x\in I\!R^n$, and let $f^k(z)=z^T\bar{Z}^k z$ with $\bar{Z}^k=\left(\bar{Z}^k\right)^T\in I\!R^{(n+1)\times(n+1)}$. Given parameters $a\neq 0$, b, c, the quadratic form $f^k(z)$ evaluated at $z=\bar{x}$ represents an output of a quadratic neural network with activation function (2) if and only if $\bar{Z}_{n+1,n+1}^k=\frac{c}{a}\mathbf{Trace}\left(\bar{Z}_{1:n,1:n}^k\right)$, where $\bar{Z}_{n+1,n+1}^k$ is the $(n+1)$-th diagonal element of $\bar{Z}^k$, and $\bar{Z}_{1:n,1:n}^k\in I\!R^{n\times n}$ is the top left block submatrix of $\bar{Z}^k$.

Proof: The proof of the only if statement follows trivially from expression (6). To prove the if statement we assume that a quadratic form is given. Since we know a, b, c, we can then compute the values of $Z_+^{k,1}-Z_-^{k,1}$, $Z_+^{k,2}-Z_-^{k,2}$, and $Z_+^{k,4}-Z_-^{k,4}$ in expression (6). Additionally, we are assuming that $Z_+^{k,4}-Z_-^{k,4}=\mathbf{Trace}\left[Z_+^{k,1}-Z_-^{k,1}\right]$. What remains to prove is that there is no additional constraint relating $Z_+^{k,2}-Z_-^{k,2}$ with $Z_+^{k,4}-Z_-^{k,4}$ and $Z_+^{k,1}-Z_-^{k,1}$ for the quadratic form to belong to the feasible set of the optimization (4). By the Schur complement the inequality constraints from (4) are equivalent to

$$\begin{aligned} Z_+^{k,4}=\mathbf{Trace}\left(Z_+^{k,1}\right)\geq 0, \\ \left[1-\mathbf{Trace}\left(Z_+^{k,1}\right)\mathbf{Trace}^\dagger\left(Z_+^{k,1}\right)\right]\left(Z_+^{k,2}\right)^T=0, \\ Z_+^{k,1}-Z_+^{k,2}\mathbf{Trace}^\dagger\left(Z_+^{k,1}\right)\left(Z_+^{k,2}\right)^T\geq 0, \end{aligned}$$

where, defining $t_k^+=\mathbf{Trace}\left(Z_+^{k,1}\right)$,

$$\mathbf{Trace}^\dagger\left(Z_+^{k,1}\right)=\begin{cases} 0, & \text{if}\quad t_k^+=0, \\ \mathbf{Trace}^{-1}\left(Z_+^{k,1}\right), & \text{if}\quad t_k^+\neq 0, \end{cases}$$

is the Moore-Penrose pseudo-inverse of $\mathbf{Trace}\left(Z_+^{k,1}\right)$, with the same conditions applying for the case of $Z_-^{k,1}$, $Z_-^{k,2}$. Simple algebraic manipulations then lead to

$$\begin{aligned} Z_+^{k,2}\left(Z_+^{k,2}\right)^T &\leq Z_+^{k,1}\left[\mathbf{Trace}\left(Z_+^{k,1}\right)\right], \\ Z_-^{k,2}\left(Z_-^{k,2}\right)^T &\leq Z_-^{k,1}\left[\mathbf{Trace}\left(Z_-^{k,1}\right)\right]. \end{aligned}$$

Applying the trace operator to these constraints leads to

$$\begin{aligned} &\|Z_+^{k,2}-Z_-^{k,2}\|^2\leq\left(\|Z_+^{k,2}\|+\|Z_-^{k,2}\|\right)^2 \\ &\leq\left[\mathbf{Trace}(Z_+^{k,1})+\mathbf{Trace}(Z_-^{k,1})\right]^2=\left[Z_+^{k,4}+Z_-^{k,4}\right]^2. \end{aligned} \tag{8}$$

We thus observe that $\|Z_+^{k,2}-Z_-^{k,2}\|$ is constrained by $Z_+^{k,4}+Z_-^{k,4}$. Note however that although the value of $Z_+^{k,4}-Z_-^{k,4}$ is constrained to be fixed given a quadratic form written as (6) and parameters a, b, c, the value of $Z_+^{k,4}+Z_-^{k,4}$ is arbitrary. Therefore, from (8) we see that there is no constraint relating $Z_+^{k,1}-Z_-^{k,1}$ or $Z_+^{k,4}-Z_-^{k,4}$ with $Z_+^{k,2}-Z_-^{k,2}$. □

Theorem 2: Each output of a quadratic neural network satisfies the following Lipschitz inequality $\forall x_1,\ x_2\in I\!R^n$

$$\begin{aligned} |\hat{f}^k(x_1)-\hat{f}^k(x_2)| &\leq L_n\left(\bar{x}_1,\bar{x}_2,\bar{Z}^k\right)\|\bar{x}_1-\bar{x}_2\|_2, \end{aligned} \tag{9}$$

$$L_n\left(\bar{x}_1,\bar{x}_2,\bar{Z}^k\right)=\sqrt{n+1}\ \left|\lambda_{max}\left(\bar{Z}^k\right)\right|\left(\|\bar{x}_1\|_\infty+\|\bar{x}_2\|_\infty\right).$$

Proof: Using the expression (6) one can write

$$|\hat{f}^k(x_1)-\hat{f}^k(x_2)|=|\left(\bar{x}_1+\bar{x}_2\right)^T\bar{Z}^k\left(\bar{x}_1-\bar{x}_2\right)|.$$

Using the Cauchy-Schwartz and triangular inequalities,

$$|\hat{f}^k(x_1) - \hat{f}^k(x_2)| \leq (\|\bar{x}_1\|_2 + \|\bar{x}_2\|_2) \|\bar{Z}^k\|_2 \|\bar{x}_1 - \bar{x}_2\|_2. \quad (10)$$

The result then follows since $\|\bar{x}\|_2 \leq \sqrt{n+1}\ \|\bar{x}\|_\infty$, for all $x \in I\!R^n$ and $\|\bar{Z}^k\|_2 = \left|\lambda_{max}\left(\bar{Z}^k\right)\right|$. □

III. System Identification

It is assumed that a collection of input output data pairs $\{u(k), y(k)\}_{k=1}^N$ are measured with $N \gg 1$. Based on the data one can identify the parameters of an autoregressive model of the form

$$y(t+1) = f(y(t-n+1), \ldots, y(t), u(t)) \quad (11)$$

by training a quadratic neural network, where $y \in I\!R^p,\ u \in I\!R^m$, and $n \geq 1$. The value of $n-1$ gives the number of delays considered in the output. We define the training matrices as

$$\begin{aligned} X &= \begin{bmatrix} u^T(n) & y^T(1) & \ldots & y^T(n) \\ \vdots & \vdots & \vdots & \vdots \\ u^T(N-1) & y^T(N-n) & \ldots & y^T(N-1) \end{bmatrix}, \\ Y &= \begin{bmatrix} y^T(n+1) \\ \vdots \\ y^T(N) \end{bmatrix}, \end{aligned} \quad (12)$$

where $X \in I\!R^{(N-n)\times(m+pn)}$ and $Y \in I\!R^{(N-n)\times p}$. Each row of the matrix X is a neural network input sample and each row of the matrix Y is an output label of the training set of the quadratic neural network. It is assumed that $N \geq n + 0.5(pn+m+1)(pn+m+2)$ and that the collected data is rich enough in terms of persistent excitation [12]. After training the network, the input-output model is written as in equation (6) changing $\bar{x}(t)$ to $\bar{y}_u(t) = [u^T(t)\quad y^T(t-n+1)\quad \ldots \quad y^T(t)\quad 1]^T$. Defining state variables as $x_1(t) = y(t-n+1),\ \ldots,\ x_n(t) = y(t)$, a state vector $x(t) = [x_1^T(t), \ldots, x_n^T(t)]^T$, and noting that $\bar{y}_u(t) = [u^T(t)\ \ x^T(t)\ \ 1]^T = [u^T(t)\ \ \bar{x}^T(t)]^T$ yields

$$x(t+1) = Ax(t) + g(x(t), u(t)) =$$
$$\begin{bmatrix} 0_{p(n-1)\times p} & I_{p(n-1)\times p(n-1)} \\ 0_{p\times p} & 0_{p\times p(n-1)} \end{bmatrix} x(t) + \begin{bmatrix} 0_{p(n-1)\times 1} \\ \mathcal{Z}(x(t), u(t)) \end{bmatrix}, \quad (13)$$

$$\mathcal{Z}(x(t), u(t)) = \begin{bmatrix} y^1(t+1) \\ \vdots \\ y^p(t+1) \end{bmatrix} = \begin{bmatrix} \bar{y}_u^T(t)\bar{Z}^1 \\ \vdots \\ \bar{y}_u^T(t)\bar{Z}^p \end{bmatrix} \bar{y}_u(t),$$

$$\bar{Z}^i = \begin{bmatrix} \bar{Z}^i_{uu} & \bar{Z}^i_{ux} & \bar{Z}^i_u \\ \left(\bar{Z}^i_{ux}\right)^T & \bar{Z}^i_{xx} & \bar{Z}^i_x \\ \left(\bar{Z}^i_u\right)^T & \left(\bar{Z}^i_x\right)^T & \bar{Z}^i_{nn} \end{bmatrix},\ i = 1, \ldots, p. \quad (14)$$

Expanding the quadratic forms in $\mathcal{Z}(x(t), u(t))$, the system (13) can be rewritten as

$$\bar{x}(t+1) = \bar{A}(x(t))\bar{x}(t) + \bar{B}(x(t))u(t) + E(u(t))u(t), \quad (15)$$

where $x \in I\!R^{n_x}, n_x = pn, u \in I\!R^m$, $\bar{A}(x(t)) = \mathcal{A} + F(x(t))$,

$$\mathcal{A} = \begin{bmatrix} A & 0_{pn\times 1} \\ 0_{1\times pn} & 1 \end{bmatrix}, A = \begin{bmatrix} 0_{p(n-1)\times p} & I_{p(n-1)\times p(n-1)} \\ 0_{p\times p} & 0_{p\times p(n-1)} \end{bmatrix}$$

$$\bar{Z}^i_{\bar{x}\bar{x}} = \begin{bmatrix} \bar{Z}^i_{xx} & \bar{Z}^i_x \\ \left(\bar{Z}^i_x\right)^T & \bar{Z}^i_{nn} \end{bmatrix}, \bar{Z}^i_{u\bar{x}} = \begin{bmatrix} \bar{Z}^i_{ux} & \bar{Z}^i_u \end{bmatrix}$$

$$F(x(t)) = \begin{bmatrix} 0_{p(n-1)\times(pn+1)} \\ \bar{x}^T(t)\bar{Z}^1_{\bar{x}\bar{x}} \\ \vdots \\ \bar{x}^T(t)\bar{Z}^p_{\bar{x}\bar{x}} \\ 0_{1\times(pn+1)} \end{bmatrix}, E(u(t)) = \begin{bmatrix} 0_{p(n-1)\times m} \\ u^T(t)\bar{Z}^1_{uu} \\ \vdots \\ u^T(t)\bar{Z}^p_{uu} \\ 0_{1\times m} \end{bmatrix}$$

$$\bar{B}(x(t)) = \begin{bmatrix} B(x(t)) \\ 0_{1\times m} \end{bmatrix} = 2 \begin{bmatrix} 0_{p(n-1)\times m} \\ \bar{x}^T(t)\left[\bar{Z}^1_{u\bar{x}}\right]^T \\ \vdots \\ \bar{x}^T(t)\left[\bar{Z}^p_{u\bar{x}}\right]^T \\ 0_{1\times m} \end{bmatrix}$$

IV. Lyapunov Control

We assume a model of the form (15), which includes but is not limited to quadratic neural networks under the formulation (13). For a system or a neural network modelled by equation (15) we propose to design a controller

$$u(t) = K(x(t))(x(t) - x_\star) + u_\star = \bar{K}_{x_t}\bar{x}(t) \quad (16)$$

to stabilize the closed-loop system to a desired state x_*, where $u_\star$ is the input steady state value when $x(t) = x_\star$,

$$\bar{K}_{x_t} = [K_{x_t}\quad u_* - K_{x_t}x_*] \quad (17)$$

where $K_{x_t} = K(x(t))$. Replacing the control input (16) in (15) yields

$$\bar{x}(t+1) = A_{cl}(x(t), \bar{K}_{x_t})\bar{x}(t), \quad (18)$$

where

$$A_{cl}(x(t), \bar{K}_{x_t}) = \bar{A}(x(t)) + \bar{B}(x(t))\bar{K}_{x_t} + E(\bar{K}_{x_t}\bar{x}(t))\bar{K}_{x_t}. \quad (19)$$

We first design u_* based on the desired steady state response of the system and then design K_{x_t} using Lyapunov theory.

A. *Steady State Input*

When a system model is in the form (15) and the desired setpoint for the steady state x_* is given, one must solve

$$\bar{x}_* = \bar{A}(x_*)\bar{x}_* + \bar{B}(x_*)u_* + E(u_*)u_* \quad (20)$$

to determine the steady state value of the input u_*.

Assumption 1: The input $u_* = 0$ is a solution of (20) when $x_* = 0$.

For the case where the system model is obtained from an input-output system identification and is in the form (13) with $x(t) = [y^T(t-n+1)\quad \ldots \quad y^T(t)]^T$, we assume that a desired setpoint y_* for the output is given. The desired state setpoint will then be $x_* = \Gamma y_*$, where $\Gamma \in I\!R^{np\times p}$ is $\Gamma = \left[I_p^T \ldots I_P^T\right]^T$, where I_p is the identity matrix of order p. The steady state value of each output is $y^i_{ss} = y^i_*$, where

y_*^i is the i-th coordinate of the desired steady state output vector y_*. Therefore, from (13) one can write

$$y_*^i = \bar{y}_{u*}^T \bar{Z}^i \bar{y}_{u*},\ i = 1, \ldots, p, \tag{21}$$

$$\bar{y}_{u*}(t) = \begin{bmatrix} u_* \\ \Gamma y_* \\ 1 \end{bmatrix}. \tag{22}$$

Equations (21)–(22) must be solved for u_* given a desired setpoint y_* in order to find the steady state values of the control input. Notice from (21)–(22) that if $\bar{Z}_{nn}^i = 0$ (no constant offset terms) then $u_* = 0$ will be a solution of (21) when $y_* = 0$ and assumption 1 will be satisfied.

B. Lyapunov Controller Synthesis

After computing a solution of (21)–(22), when one exists, replacing the input (16) in equation (13) yields

$$y^i(t) = \bar{x}^T(t) \bar{Z}_{cl}^i(\bar{K}_{x_t}) \bar{x}(t),\ i = 1, \ldots, p, \tag{23}$$

where

$$\bar{Z}_{cl}^i(\bar{K}_{x_t}) = \begin{bmatrix} \bar{Z}_{x_{cl}x_{cl}}^i & \bar{Z}_{x_{cl}}^i \\ \left(\bar{Z}_{x_{cl}}^i\right)^T & \bar{Z}_{n_{cl}}^i \end{bmatrix},\ i = 1, \ldots, p, \tag{24}$$

with

$$\begin{aligned} Z_{x_{cl}x_{cl}}^i &= \bar{Z}_{xx}^i + K_{x_t}^T \bar{Z}_{ux}^i + \left[\bar{Z}_{ux}^i\right]^T K_{x_t} + K_{x_t}^T \bar{Z}_{uu}^i K_{x_t}, \\ \bar{Z}_{x_{cl}}^i &= \bar{Z}_x^i + \left(\bar{Z}_{uu}^i K_{x_t} + \bar{Z}_{ux}^i\right)^T (u_* - K_{x_t} x_*) + K_{x_t}^T \bar{Z}_u^i, \\ \bar{Z}_{n_{cl}}^i &= \bar{Z}_{nn}^i + 2\left(u_* - K_{x_t} x_*\right)^T Z_u^i + \delta u_*^T \bar{Z}_{uu}^i \delta u_*, \end{aligned} \tag{25}$$

where $\delta u_* = (u_* - K_{x_t} x_*)$. Therefore, the ouput is

$$y(t) = \begin{bmatrix} \bar{x}^T(t) \bar{Z}_{cl}^1(\bar{K}_{x_t}) \\ \vdots \\ \bar{x}^T(t) \bar{Z}_{cl}^p(\bar{K}_{x_t}) \end{bmatrix} \bar{x}(t). \tag{26}$$

From (13) and (26) the closed-loop state space model can be rewritten as in equation (18) where

$$A_{cl}(x(t), \bar{K}_{x_t}) = \begin{bmatrix} \bar{I} \\ \bar{x}^T(t) Z_{cl}^1(\bar{K}_{x_t}) \\ \vdots \\ \bar{x}^T(t) Z_{cl}^p(\bar{K}_{x_t}) \\ \bar{0}^T \end{bmatrix}, \tag{27}$$

$\bar{I} = \begin{bmatrix} 0_{p(n-1)\times p} & I_{p(n-1)\times p(n-1)} & 0_{p(n-1)\times 1} \end{bmatrix}$, and $\bar{0}^T = \begin{bmatrix} 0_{1\times pn} & 1 \end{bmatrix}$. Equation (27) can be rewritten in the form (19). Making the substitution $\bar{K}_{x_t} = 0$ in (27) yields the open-loop model. Stability of the closed-loop system for a given controller can be proved by finding a Lyapunov function. The next Theorem provides a Lyapunov-based design strategy that yields a provably stabilizing controller.

Theorem 3: Given a desired setpoint $x_* \in \mathcal{X} \subseteq I\!R^{n_x}$, if

$$\begin{aligned} A_{cl}^T(x(t), \bar{K}_{x_t}) \bar{P} A_{cl}(x(t), \bar{K}_{x_t}) - \bar{P} \leq 0, \\ \forall\ x(t) \neq x_*, x(t) \in \mathcal{X} \subseteq I\!R^{n_x}, \end{aligned} \tag{28}$$

is satisfied where A_{cl} is given by (19) [or by (27)] and $\bar{P}$ is defined as

$$\bar{P} = \begin{bmatrix} P & -Px_* \\ -x_*^T P & x_*^T P x_* \end{bmatrix} \tag{29}$$

where $P > 0$, then the controller $u(t) = \bar{K}_{x_t} \bar{x}(t)$ renders the closed-loop system (18) stable in the sense of Lyapunov inside the largest invariant set of the Lyapunov function (30) fully contained in $\mathcal{X}$. If the inequality (28) is strict, the closed-loop system is asymptotically stable inside the same invariant set. If $\mathcal{X} = I\!R^{n_x}$, the stability is global.

Proof: If x_* is the desired setpoint in steady state we define the candidate quadratic control Lyapunov function

$$V(x(t)) = (x(t) - x_*)^T P (x(t) - x_*) = \bar{x}^T(t) \bar{P} \bar{x}(t), \tag{30}$$

where $\bar{P}$ is defined in (29). For Lyapunov stability,

$$V(x_*) = 0, \tag{31}$$

$$V(x) > 0, \qquad \forall\ x \neq x_*, x \in \mathcal{X} \tag{32}$$

$$\Delta V =\leq 0, \qquad \forall\ x(t) \neq x_*, x(t) \in \mathcal{X}, \tag{33}$$

where $\Delta V = V(x(t+1)) - V(x(t))$. Notice from the definition of the Lyapunov function (30) that the condition (31) is guaranteed to be satisfied. Furthermore, the condition (32) is also satisfied because $P > 0$. Computing $V(x(t+1))$ using (18) yields

$$V(x(t+1)) = \bar{x}^T(t) A_{cl}^T(x(t), \bar{K}_{x_t}) \bar{P} A_{cl}(x(t), \bar{K}_{x_t}) \bar{x}(t). \tag{34}$$

Using (34) condition (33) is implied by (28). From standard Lyapunov theory the system is then stable inside the largest invariant set of the Lyapunov function (30) fully contained in $\mathcal{X}$. The proof of asymptotical stability follows by noting that the Lyapunov function is decreasing when the strict inequality is satisfied. Global stability follows from the fact that the Lyapunov function (30) is radially unbounded. □

Note that, according to Theorem 1, for a known vector x_* the Lyapunov function (30) is a quadratic neural network with activation parameters $a \neq 0, b, c,$ if and only if $x_*^T P x_* = \frac{c}{a} \textbf{Trace}(P)$.

Theorem 4: Consider the quadratic Lyapunov function (30) with $P > 0$. Given scalars c and $a \neq 0$, if the desired steady state x_* is a random vector with zero mean and covariance matrix $\Sigma = \frac{c}{a} I$, where I is the identity, then

$$E\left[x_*^T P x_*\right] = \frac{c}{a} \textbf{Trace}\,(P)\,. \tag{35}$$

In other words, the matrix $\bar{P}$ in (29) can represent "on average" a quadratic neural network if and only if x_* is a zero mean random vector with covariance matrix $\Sigma = \frac{c}{a} I$.

Proof: Use Theorem 1 and Lemma 2, $\mu = 0, \Sigma = \frac{c}{a} I$. □

Unfortunately, condition (28) is not convex. The following assumption enables control design as a convex optimization.

Assumption 2: $E(u(t)) = 0$ for all $u(t) \in I\!R^m$.

Under assumptions 1 and 2 the model (15) becomes

$$x(t+1) = A(x(t)) x(t) + B(x(t)) u(t), \tag{36}$$

for appropriate matrices $A \in I\!R^{n_x \times n_x}, B \in I\!R^{nx \times m}$. The following result can then be stated and proved.

Theorem 5: Assume that a system model is given in the form (36) where $x(t) \in \mathcal{X} \subseteq I\!R^{n_x}$. If the desired steady state setpoint is $x_* = 0$ and if for a given $\epsilon \in (0,1)$ there are $P = P^T$ and $L(x(t))$ satisfying

$$\begin{bmatrix} (1-\epsilon)P - \epsilon I & (*)^T \\ A(x(t))P + B(x(t))L(x(t)) & P \end{bmatrix} \geq 0 \qquad (37)$$

$\forall\ x(t) \in \mathcal{X}$, where $(*) = A(x(t))P + B(x(t))L(x(t))$, then the closed-loop system is asymptotically stable in the largest invariant set of the Lyapunov function $V(x) = x^T P^{-1} x$ fully contained inside $\mathcal{X}$. If $\mathcal{X} = I\!R^{n_x}$ then the stability is global. The control input is $u(t) = L(x(t))P^{-1}x(t)$.

Proof: Replacing $u(t) = K(x(t))x(t)$ in (36) yields

$$x(t+1) = A_{cl}(x(t), K(x(t))x(t),$$
$$A_{cl}(x(t), K(x(t))) = A(x(t)) + B(x(t))K(x(t)). \qquad (38)$$

Consider a candidate Lyapunov function of the form $V(x) = x^T P^{-1} x$. Note that if the condition (37) is satisfied then $P > 0$ and therefore it is invertible. It is clear that $V(0) = 0$ and $V(x) > 0$ for $x \neq 0$. Additionally, if (37) is satisfied then applying the Schur complement and using (38) it implies that $PA_{cl}^T(x(t), K(x(t)))P^{-1}A_{cl}(x(t), K(x(t)))P - (1-\epsilon)P \leq -\epsilon I < 0\ \forall\ x \in \mathcal{X}$, using the substitution $K = L(x(t))P^{-1}$. Multiplying on the left by $x^T(t)P^{-1}$ and on the right by $P^{-1}x(t)$ yields $\Delta V < -\epsilon x^T P^{-1} x,\ \forall x \in \mathcal{X}$, which guarantees asymptotic stability in the largest level set of $V(x)$ fully contained in $\mathcal{X}$. Global stability follows because the Lyapunov function is radially unbounded. □

C. Convex Formulation of Lyapunov Controller Synthesis

To formulate controller synthesis as a convex optimization problem we will need the framework of sum of squares polynomials. For $\mathbf{x} \in I\!R^n$, a multivariate polynomial $p(\mathbf{x})$ is a sum of squares (SOS) if there exist some polynomials $f_i(\mathbf{x}),\ i = 1, \ldots, M$, such that $p(\mathbf{x}) = \sum_{i=1}^{M} f_i^2(\mathbf{x})$. A polynomial $p(\mathbf{x})$ of degree $2d$ is a sum of squares if and only if there exists a positive semidefinite matrix Q and a vector $Z(\mathbf{x})$ containing monomials in $\mathbf{x}$ of degree less than d such that [13] $p(\mathbf{x}) = W(\mathbf{x})^T Q W(\mathbf{x})$ [13]. It should be noted that if $p(\mathbf{x})$ is a sum of squares then $p(\mathbf{x}) \geq 0$, but the converse is generally not true. For a convex formulation the inequality (37) will be relaxed into a sum of squares and $K(x(t))$ will be constrained to have polynomial entries.

Corollary 1: Assume that a system model is given in the form (36) where $A(x(t))$ and $B(x(t))$ have polynomial entries and $x(t) \in \mathcal{X} \subseteq I\!R^{n_x}$. If the desired steady state setpoint is $x_* = 0$ and if for a given $0 \leq \epsilon < 1$ there are $P > 0$ and $L(x(t))$ with polynomial entries satisfying

$$\begin{bmatrix} (1-\epsilon)P - \epsilon I & (*)^T \\ A^T(x(t))P + B(x(t))L(x(t)) & P \end{bmatrix} \text{is SOS}, \qquad (39)$$

$\forall\ x(t) \in \mathcal{X}$, where $(*) = PA(x(t)) + L^T(x(t))B^T(x(t))$, then the closed-loop system is stable in the largest invariant set of the Lyapunov function $V(x) = x^T P^{-1} x$ fully contained inside $\mathcal{X}$. If the condition (39) is satisfied for $\epsilon \in (0,1)$, then the closed-loop system is asymptotically stable inside the same invariant set. If $\mathcal{X} = I\!R^n$ the stability is global. The control input is given by $u(t) = L(x(t))P^{-1}x(t)$.

Proof: The proof follows the argument of the proof of Theorem 5 upon observing that satisfaction of the condition (39) implies that the inequality (37) is satisfied. □

Sometimes one is interested not only on stability but in some measure of performance of a stabilizing controller. To measure performance one can consider the quadratic cost

$$J(x,u) = \sum_{k=0}^{\infty} \left(x^T(k)Qx(k) + u^T(k)Ru(k) \right) \qquad (40)$$

for given matrices $Q = Q^T \geq 0$ and $R = R^T > 0$. The lower this cost is for a given controller the better is its performance.

Theorem 6: Assume that a system model is given in the form (36) where $A(x(t))$ and $B(x(t))$ have polynomial entries. If there exists $P = P^T$ and $W(x(t))$ defined as

$$\begin{bmatrix} A^T(x)PA(x) - P + Q & A^T(x)PB(x) \\ B^T(x)PA(x) & R + B^T(x)PB(x) \end{bmatrix} \qquad (41)$$

is SOS $\forall\ x(t) \in I\!R^{n_x}$, then any controller that asymptotically stabilizes the system to the origin yields a cost (40) that is bounded below by $V(x_0) = x_0^T P x_0$ when the system trajectories start from an initial condition $x_0 \in I\!R^{n_x}$.

Proof: The condition (41) implies that the matrix $W(x(t))$ is positive semidefinite for all $x(t) \in I\!R^{n_x}$. Left multiplying $W(x(t))$ by $[x^T(t)\ \ u^T(t)]$ and right multiplying by $[x^T(t)\ \ u^T(t)]^T$ and using (36) yields

$$x^T(t)Qx(t) + u^T(t)Ru(t) + V(x(t+1)) - V(x(t)) \geq 0, \qquad (42)$$

where $V(x(t)) = x^T(t)Px(t)$. Summing the terms on the left hand side of inequality (42) from zero to infinity yields

$$J(x,u) \geq V(x(0)) - V\left(\lim_{t\to\infty} x(t)\right) = V(x_0), \qquad (43)$$

provided $x(t) \to 0$ as $t \to \infty$, which is guaranteed when the closed-loop system is asymptotically stable to the origin. □

Remark 1: For some systems in the form (36) the optimal cost to go to the origin from a given state x is $V(x) = x^T P x$, where P is the matrix of maximum trace that satisfies condition (41). That is the case for example when the system is linear and the inequality (42) becomes an equality called the Bellman equation [14]. The optimal control is

$$u(t) = -\left(R + B^T(x)PB(x)\right)^{-1} B^T(x)PA(x)x \qquad (44)$$

A common heuristic is to find P with maximum trace that satisfies condition (41) and to compute the control (44).

V. Examples

Example 1: We perform system identification of a flexible robot arm nonlinear autoregressive model with $n = 1$. The data is available at the website of the Database for the Identification of Systems (DaISy) [15]. The applied input was a periodic sine sweep for which an output consisting of $N = 1024$ data points was collected. The training was done for a quadratic activation function with parameters $a = 0.0937, b = 0.5, c = 0.4688$, using a regularization coefficient of $\beta = 0.01$ and an infinity norm loss function

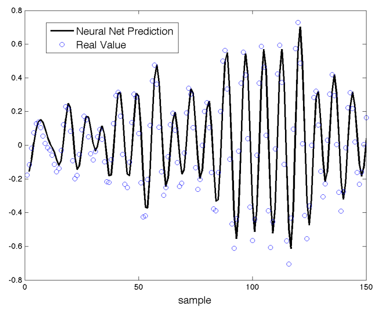

Fig. 1. System identification (black curve) and data points (blue circles).

$l(\cdot)$. Only the first 122 data points (12% of all data) were used to train the neural network. The training was solved with cvx [16] using MOSEK [17]. The result is $y(t+1) = \bar{y}_u^T(t)\bar{Z}\bar{y}_u(t)$ where $\bar{y}_u(t) = [u^T(t) \quad y^T(t) \quad 1]^T$,

$$\bar{Z} = \begin{bmatrix} -0.0989 & 0.5030 & -0.1682 \\ 0.5030 & 0.1599 & 1.8265 \\ -0.1682 & 1.8265 & 0.0610 \end{bmatrix}.$$

Fiigure 1 compares the model predictions with the data points.

Example 2: Consider a model of a quadrotor flying at a constant altitude in the positive X direction.

$$\begin{bmatrix} \dot{X} \\ \dot{V}_X \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} X \\ V_X \end{bmatrix} + \begin{bmatrix} 0 \\ g \end{bmatrix} u - \begin{bmatrix} 0 \\ d \end{bmatrix} V_X^2, \tag{45}$$

where X and V_X represent the position and the velocity of the quadrotor, respectively, $u = \tan(\theta)$ is the control input where θ is the pitch angle of the quadrotor (positive counter clockwise), g is the acceleration of gravity, and $d = \frac{\rho S C_D}{2m}$ is the term related to the drag force where ρ is the air density, S is a characteristic surface, m is the mass of the quadrotor, and C_D is the coefficient of drag. The control objective is to take the system to the origin from any initial position with a negative X coordinate. The model (45) is in continuous-time. It is discretized using a forward Euler approximation with sampling time $T = 0.102s$ yielding model (36) with

$$A(x(t)) = \begin{bmatrix} 1 & T \\ 0 & 1 - TdV_X(t) \end{bmatrix}, \quad B(x(t)) = \begin{bmatrix} 0 \\ Tg \end{bmatrix}.$$

The parameters of the system are $g = 9.8ms^{-2}, \rho = 1.097kgm^{-3}, S = 1m^2, C_D = 0.05, m = 1.2kg$, which yield $Td = 0.0023sm^{-1}$ and $Tg = 1ms^{-1}$. The matrix P with minimum condition number that satisfies condition (37) with $\epsilon = 0.1$ and $L(x)$ a first order polynomial is

$$P = \begin{bmatrix} 1.4589 & -1.6008 \\ -1.6008 & 2.6636 \end{bmatrix},$$

which represents a quadratic neural network with $ca^{-1} = (2.6636)(1.4589)^{-1} = 1.8258$. The controller is given by $u(t) = -1.1556X(t) - 1.1771V_X(t) + 0.0023V_X^2(t)$. Note that the controller cancels the term in $V_X^2(t)$ from the open-loop system and yields a linear closed-loop system with stable eigenvalues $\lambda_1 = 0.8895, \lambda_2 = -0.0666$. Finding the matrix P with maximum trace that satisfies condition (41) for $Q = I$ and $R = 2.2 \times 10^{-16}$ using SoSTools [13] yields

$$P = \begin{bmatrix} 11.3167 & 1.0523 \\ 1.0523 & 1.1073 \end{bmatrix},$$

which represents a quadratic neural network with $ca^{-1} = 0.0979$. Using equation (44) leads to the controller $u(t) = -0.9503X(t) - 1.097V_X(t) + 0.0023V_X^2(t)$. Note that this controller also cancels the term in $V_X^2(t)$ from the open-loop system and yields a linear closed-loop system with stable eigenvalues $\lambda_1 = 0.9031, \lambda_2 = -0.0001$. One can write

$$u(t) = \begin{bmatrix} X \\ V_X \\ 1 \end{bmatrix}^T \begin{bmatrix} 0 & 0 & -0.47515 \\ 0 & 0.0023 & -0.5485 \\ -0.47515 & -0.5485 & 0 \end{bmatrix} \begin{bmatrix} X \\ V_X \\ 1 \end{bmatrix}.$$

Thus the control can be implemented by a QNN with $c = 0$.

References

[1] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115–133, 1943.

[2] H. Dai, B. Landry, L. Yand, M. Pavone, and R. Tedrake, "Lyapunov-stable neural-network control," in *Proceedings of Robotics: Science and Systems 2021*, Held Virtually, July 12-16 2021.

[3] G.-B. Huang and H. Babri, "Upper bounds on the number of hidden neurons in feedforward networks with arbitrary bounded nonlinear activation functions," *IEEE Transactions on Neural Networks*, vol. 9, no. 1, pp. 224–229, 1998.

[4] P. Cheridito, A. Jentzen, and F. Rossmannek, "Efficient approximation of high-dimensional functions with neural networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2021.

[5] Y. Luo, J. Lü, X. Jiang, and B. Zhang, "Learning from architectural redundancy: Enhanced deep supervision in deep multipath encoder-decoder networks," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2021.

[6] M. Fazlyab, A. Robey, H. Hassani, M. Morari, and G. J. Pappas, "Efficient and accurate estimation of Lipschitz constants for deep neural networks," in *Proceedings of the 33rd International Conference on Neural Information Processing Systems (NIPS)*, December 2019, pp. 11 427–11 438.

[7] P. Pauli, A. Koch, J. Berberich, P. Kohler, and F. Allgöwer, "Training robust neural networks using Lipschitz bounds," *IEEE Control Systems Letters*, vol. 6, pp. 121–126, 2022.

[8] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[9] Y. Bengio, N. L. Roux, P. Vincent, O. Dellaleau, and P. Marcotte, "Convex neural networks," in *Advances in Neural Information Processing Systems*. MIT Press, 2005.

[10] B. Bartan and M. Pilanci, "Neural spectrahedra and semidefinite lifts: Global convex optimization of polynomial activation neural networks in fully polynomial-time," *arxiv.org*, 2021.

[11] D. A. Kendrick, *Stochastic Control for Economic Models*, 2nd ed. McGraw-Hill Inc., 2002.

[12] L. Ljung, *System Identification: Theory for the User*. Prentice Hall, 1999.

[13] S. Prajna, A. Papachristodoulou, P. Seiler, and P. A. Parrilo, *SOSTOOLS and Its Control Applications*. Springer-Verlag, 2005, pp. 273–292.

[14] F. L. Lewis, D. L. Vrabie, and V. L. Syrmos, *Optimal Control*, 3rd ed. John Wiley & Sons, Inc., 2012.

[15] B. D. Moor, P. D. Gersem, B. D. Schutter, and W. Favoreel, "Daisy: A database for identification of systems," *Journal A*, vol. 38, no. 3, pp. 4–5, 1997.

[16] M. C. Grant and S. Boyd, *The CVX User's Guide Release 2.1*, CVX Research, Inc., December 2017.

[17] *MOSEK Modeling Cookbook Release 3.2.3*, November 2021.