Neurocomputing 72 (2009) 1131–1137

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Applications of the general projection neural network in solving extended linear-quadratic programming problems with linear constraints ☆

Xiaolin Hu

Tsinghua National Lab for Information Science and Technology, State Key Lab of Intelligent Technology and Systems, and Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China

ARTICLE INFO

Article history:
Received 13 October 2007
Received in revised form
31 January 2008
Accepted 25 February 2008
Communicated by T. Heskes
Available online 20 March 2008

Keywords:
Extended linear-quadratic programming
Extended linear programming
Generalized linear variational inequality
General projection neural network
Global asymptotic stability

ABSTRACT

Extended linear-quadratic programming (ELQP) is an extension of the conventional linear programming and quadratic programming, which arises in many dynamic and stochastic optimization problems. Existing neural network approaches are limited to solve ELQP problems with bound constraints only. In the paper, I consider solving the ELQP problems with general polyhedral sets by using recurrent neural networks. An existing neural network in the literature, called general projection neural network (GPNN) is investigated for this purpose. In addition, based on different types of constraints, different approaches are utilized to lower the dimensions of the designed GPNNs and consequently reduce their structural complexities. All designed GPNNs are stable in the Lyapunov sense and globally convergent to the solutions of the ELQP problems under mild conditions. Numerical simulations are provided to validate the results.

1. Introduction

The notion of extended linear-quadratic programming (ELQP) was introduced by Rockafellar and Wets in the 1980s in setting up various large-scale models in dynamic and stochastic programming and multistage optimization [20,19]. A general ELQP problem takes the form

$$\min_{x\in X} f(x) = p^{\mathrm{T}}x + \tfrac{1}{2}x^{\mathrm{T}}Px + \rho_{XQ}(-q - D^{\mathrm{T}}x), \tag{1}$$

where

$$\rho_{XQ}(w) = \max_{y\in Y}\{w^{\mathrm{T}}y - \tfrac{1}{2}y^{\mathrm{T}}Qy\},$$

$x \in \Re^n$, $p \in \Re^n$, $y \in \Re^m$, $q \in \Re^m$, $D \in \Re^{n\times m}$, $P = P^{\mathrm{T}} \in \Re^{n\times n}$, $Q = Q^{\mathrm{T}} \in \Re^{m\times m}$, and $X \subseteq \Re^n$, $Y \subseteq \Re^m$ are nonempty convex sets. The matrices P and Q are both positive semi-definite. The ELQP problem includes many classical problems as special cases. For example, by setting different parameters in (1) to zeros, one easily obtains the usual linear programming problems or quadratic programming problems. ELQP also includes the extended linear programming (ELP) as a special case, which itself has many applications in practice [2]. ELQP differs from the usual linear programming, quadratic programming in that its objective function is only piecewise linear-quadratic in general [20].

To solve the ELQP problems, many numerical algorithms have been proposed (e.g., see [7,18–21]). Neural network approaches for solving ELQP problems have also been investigated and good results have been obtained, e.g. [23,5,4]. The primary aim of developing neural networks for solving this type of problems, as well as for solving other optimization related problems (see [24–28,10–14,3,8,6,17,16] and references therein), is not to invent numerical algorithms and then compete with existing ones on conventional computers, though sometimes such an algorithm can be obtained as a byproduct (e.g. [14]). Instead, it is to develop some devices that can imitate our brains to do complex computing in noisy and uncertain circumstances, which would be definitely useful in the evolving artificial intelligence community. Such devices, we call *artificial brains*, could be analog circuits. For specific computing, these analog circuits will be much faster than numerical algorithms that have to be implemented on conventional digital equipments [9,22].

To the best of the author's knowledge, the earliest attempt for designing neural networks for solving ELQP problems (1) refers to [23] by Tao and Fang, where the constraint sets X and Y in (1) are assumed to be of box type. Actually, if this assumption holds, it is shown in [5] that the condition of the saddle point of the problem can be formulated as the following linear variational inequality (LVI): to find a vector $x^* \in \Omega$ such that

$$(Mx^* + a)^{\mathrm{T}}(x - x^*) \geqslant 0, \quad \forall x \in \Omega, \tag{2}$$

☆ The work was supported by the National Natural Science Foundation of China under Grant 60621062 and by the National Key Foundation R&D Project under Grants 2003CB317007, 2004CB318108 and 2007CB311003.

E-mail address: xiaolin.hu@gmail.com

0925-2312/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2008.02.016

where M is a matrix, a is a vector, and Ω is a box set. In view of this fact, the neural network models for solving LVIs such as in [8,25,11] can all be applied to solve (1). In particular, the projection neural network (PNN) for solving LVIs proposed in [25] is studied in [5] for solving ELQPs. Another model is presented in [4] with improved convergence property.

However, the neural networks presented in [23,5,4] deal with ELQP problems with box type X and Y only, thus have great limitations in practice. It is possible to formulate the ELQP problem with general polyhedral constraint sets into LVI in the form of (2) (e.g., using similar techniques to what will be presented in the paper), then the neural networks in [23,5,4] can be used to solve the problem. However, the dimensions of the resulting neural networks will be very high. In this paper, I consider to solve the ELQP problem with general polyhedral X and Y, which are defined by a set of inequality and equality constraints, for instance,

$$X = \{x \in \Re^n | x \in \Omega_x, Ax \in \Omega_s, Cx = c\},$$
$$Y = \{y \in \Re^m | y \in \Omega_y, By \in \Omega_w, Ey = e\}, \quad (3)$$

where $A \in \Re^{h_1 \times n}$, $B \in \Re^{h_2 \times n}$, $C \in \Re^{r_1 \times m}$, $E \in \Re^{r_2 \times m}$, $c \in \Re^{r_1}$, $e \in \Re^{r_2}$, and $\Omega_x \subseteq \Re^n$, $\Omega_y \subseteq \Re^m$, $\Omega_s \subseteq \Re^{h_1}$, $\Omega_w \subseteq \Re^{h_2}$ are box sets. For convenience, in (3), $x \in \Omega_x$, $y \in \Omega_y$ are termed bound constraints, $Ax \in \Omega_s$, $By \in \Omega_w$ are termed inequality constraints, and $Cx = c$, $Ey = e$ are termed equality constraints. Throughout the paper it is assumed that neither X nor Y is empty.

The idea of my approach is as follows. First, by utilizing some techniques in the optimization context, I formulate the optimality conditions of the ELQP problems into the so-called generalized linear variational inequalities (GLVIs): to find a vector x^* such that $Nx^* + b \in \Omega$ and

$$(Mx^* + a)^{\mathrm{T}}(x - Nx^* - b) \geqslant 0, \quad \forall x \in \Omega, \quad (4)$$

where M, N are matrices, a, b are vectors and Ω is a closed convex set. Then, the general projection neural network (GPNN) presented in [3,26] is applied to solve the GLVIs, and consequently, the original ELQP problems.

The network complexity is an important issue in studying neural networks. Based on different types of constraints, I will devote myself to design lower dimensional neural networks for solving ELQP problems with different combinations of constraints.

2. ELQP problems with bound constraints and inequality constraints

Let us first consider the ELQP problem with inequality constraints and bound constraints only, that is,

$$X = \{x \in \Re^n | x \in \Omega_x, Ax \in \Omega_s\},$$
$$Y = \{y \in \Re^m | y \in \Omega_y, By \in \Omega_w\}, \quad (5)$$

where the notations are the same as in (3). Note that the equality constraints in (3) can be rewritten as inequality constraints, so the neural network that will be designed in this section is applicable to the ELQP problem with X, Y defined in (3), too; see Remark 3 for a discussion in the end of this section. Denote the box sets $\Omega_s = \{s \in \Re^{h_1} | \underline{s} \leqslant s \leqslant \overline{s}\}, \Omega_w = \{w \in \Re^{h_2} | \underline{w} \leqslant w \leqslant \overline{w}\}$. The following theorem establishes a necessary and sufficient optimality condition for problem (1) with X and Y defined by (5).

Theorem 1. *Let $x^* \in \Omega_x$ and $y^* \in \Omega_y$. Then (x^*, y^*) is a solution to problem (1) with X and Y defined by (5) if and only if there exists $s^* \in \Re^{h_1}$, $w^* \in \Re^{h_2}$ such that $Ax^* \in \Omega_s$, $By^* \in \Omega_w$, and*

$$\begin{cases} (Px^* - Dy^* - A^{\mathrm{T}}s^* + p)^{\mathrm{T}}(x - x^*) \geqslant 0, & \forall x \in \Omega_x, \\ (D^{\mathrm{T}}x^* + Qy^* - B^{\mathrm{T}}w^* + q)^{\mathrm{T}}(y - y^*) \geqslant 0, & \forall y \in \Omega_y, \\ (s^*)^{\mathrm{T}}(s - Ax^*) \geqslant 0, & \forall s \in \Omega_s, \\ (w^*)^{\mathrm{T}}(w - By^*) \geqslant 0, & \forall w \in \Omega_w. \end{cases} \quad (6)$$

Proof. In view of the fact that the general constraints in problem (1) can be expressed as

$$Ax - \alpha = 0, \quad By - \beta = 0, \ \alpha \in \Omega_s, \ \beta \in \Omega_w,$$

define the Lagrangian function to problem (1) on $\Omega_x \times \Omega_y \times \Omega_s \times \Omega_w \times \Re^{h_1} \times \Re^{h_2}$ as follows:

$$L(x, y, \alpha, \beta, s, w) = g(x, y) + s^{\mathrm{T}}(\alpha - Ax) - w^{\mathrm{T}}(\beta - By),$$

where

$$g(x, y) = \tfrac{1}{2}x^{\mathrm{T}}Px - \tfrac{1}{2}y^{\mathrm{T}}Qy - x^{\mathrm{T}}Dy + p^{\mathrm{T}}x - q^{\mathrm{T}}y.$$

According to the well-known saddle point theorem [1], (x^*, y^*) is a solution to (1) if and only if there exists $\alpha^* \in \Omega_s$, $\beta^* \in \Omega_w$, $s^* \in \Re^{h_1}$, $w^* \in \Re^{h_2}$ such that

$$\begin{aligned} L(x^*, y, \alpha^*, \beta, s, w^*) &\leqslant L(x^*, y^*, \alpha^*, \beta^*, s^*, w^*) \\ &\leqslant L(x, y^*, \alpha, \beta^*, s^*, w), \\ &\quad \forall x \in \Omega_x, \ y \in \Omega_y, \ \alpha \in \Omega_s, \\ &\quad \beta \in \Omega_w, \ s \in \Re^{h_1}, \ w \in \Re^{h_2}. \end{aligned} \quad (7)$$

The left inequality in (7) implies

$$\begin{aligned} &\tfrac{1}{2}y^{\mathrm{T}}Qy + q^{\mathrm{T}}y + (x^*)^{\mathrm{T}}Dy - s^{\mathrm{T}}(\alpha^* - Ax^*) + (w^*)^{\mathrm{T}}(\beta - By) \\ &\geqslant \tfrac{1}{2}(y^*)^{\mathrm{T}}Qy^* + q^{\mathrm{T}}y^* + (x^*)^{\mathrm{T}}Dy^* - (s^*)^{\mathrm{T}}(\alpha^* - Ax^*) \\ &\quad + (w^*)^{\mathrm{T}}(\beta^* - By^*), \quad \forall y \in \Omega_y, \ \beta \in \Omega_w, \ s \in \Re^{h_1}. \end{aligned}$$

Define a function

$$\begin{aligned} \phi(y, \beta, s) \triangleq\ & \tfrac{1}{2}y^{\mathrm{T}}Qy + q^{\mathrm{T}}y + (x^*)^{\mathrm{T}}Dy \\ & - s^{\mathrm{T}}(\alpha^* - Ax^*) + (w^*)^{\mathrm{T}}(\beta - By), \end{aligned}$$

which is obviously convex in y, linear in s and β. Note

$$\phi(y^*, \beta^*, s^*) = \min \phi(y, \beta, s), \quad \forall y \in \Omega_y, \ \beta \in \Omega_w, \ s \in \Re^{h_1},$$

which is equivalent to the following equations [15]:

$$\begin{cases} (Qy^* + q + D^{\mathrm{T}}x^* - B^{\mathrm{T}}w^*)^{\mathrm{T}}(y - y^*) \geqslant 0, & \forall y \in \Omega_y, \\ (w^*)^{\mathrm{T}}(\beta - \beta^*) \geqslant 0, & \forall \beta \in \Omega_w, \\ \alpha^* - Ax^* = 0. \end{cases} \quad (8)$$

The right inequality in (7) implies

$$\begin{aligned} &\tfrac{1}{2}(x^*)^{\mathrm{T}}Px^* + p^{\mathrm{T}}x^* - (x^*)^{\mathrm{T}}Dy^* + (s^*)^{\mathrm{T}}(\alpha^* - Ax^*) - (w^*)^{\mathrm{T}}(\beta^* - By^*) \\ &\leqslant \tfrac{1}{2}x^{\mathrm{T}}Px + p^{\mathrm{T}}x - x^{\mathrm{T}}Dy^* + (s^*)^{\mathrm{T}}(\alpha - Ax) - w^{\mathrm{T}}(\beta^* - By^*), \\ &\quad \forall x \in \Omega_x, \ \alpha \in \Omega_s, \ w \in \Re^{h_2}. \end{aligned}$$

Define a function

$$\begin{aligned} \psi(x, \alpha, w) \triangleq\ & \tfrac{1}{2}x^{\mathrm{T}}Px + p^{\mathrm{T}}x - x^{\mathrm{T}}Dy^* \\ & + (s^*)^{\mathrm{T}}(\alpha - Ax) - w^{\mathrm{T}}(\beta^* - By^*), \end{aligned}$$

which is obviously convex in x, linear in α and w. Note

$$\psi(x^*, \alpha^*, w^*) = \min \psi(x, \alpha, w), \quad \forall x \in \Omega_x, \ \alpha \in \Omega_s, \ w \in \Re^{h_2},$$

which is equivalent to [15]

$$\begin{cases} (Px^* + p - Dy^* - A^{\mathrm{T}}s^*)^{\mathrm{T}}(x - x^*) \geqslant 0, & \forall x \in \Omega_x, \\ (s^*)^{\mathrm{T}}(\alpha - \alpha^*) \geqslant 0, & \forall \alpha \in \Omega_s, \\ \beta^* - By^* = 0. \end{cases} \quad (9)$$

By combining (8) and (9), and replacing α and β with notations s and w, respectively, we obtain the equivalent formulation of (7) as (6), which completes the proof. □

Note that (6) can be put into the following compact form:

$$(M_1u^* + a_1)^{\mathrm{T}}(u - N_1u^*) \geqslant 0, \quad \forall u \in U_1, \tag{10}$$

where $u = (x^{\mathrm{T}}, y^{\mathrm{T}}, s^{\mathrm{T}}, w^{\mathrm{T}})^{\mathrm{T}}$ and

$$M_1 = \begin{pmatrix} P & -D & -A^{\mathrm{T}} & 0 \\ D^{\mathrm{T}} & Q & 0 & -B^{\mathrm{T}} \\ 0 & 0 & I & 0 \\ 0 & 0 & 0 & I \end{pmatrix}, \quad a_1 = \begin{pmatrix} p \\ q \\ 0 \\ 0 \end{pmatrix},$$

$$N_1 = \begin{pmatrix} I & 0 & 0 & 0 \\ 0 & I & 0 & 0 \\ A & 0 & 0 & 0 \\ 0 & B & 0 & 0 \end{pmatrix}, \quad U_1 = \Omega_x \times \Omega_y \times \Omega_s \times \Omega_w,$$

with I being the identity matrix of suitable dimension. Then solving the ELQP problem (1) is equivalent to solving this GLVI problem. Therefore the following neural network can be used to solve the problem [3,26]:

- state equation:

$$\frac{\mathrm{d}u}{\mathrm{d}t} = \lambda(N_1 + M_1)^{\mathrm{T}}(\mathscr{P}_{U_1}((N_1 - M_1)u - a_1) - N_1u); \tag{11a}$$

- output equation:

$$v = H_1u, \tag{11b}$$

where $\lambda > 0$, $H_1 = (I, 0) \in \Re^{(n+m)\times(n+m+h_1+h_2)}$ and $\mathscr{P}_{\Omega}(\cdot)$ is a projection operator which projects a point onto a closed convex set Ω. If Ω is defined as $\{\xi \in \Re^n | \underline{\xi}_i \leqslant \xi_i \leqslant \overline{\xi}_i, i = 1, \ldots, n\}$, then $P_{\Omega}(\xi) = (P_{\Omega}(\xi_1), \ldots, P_{\Omega}(\xi_n))^{\mathrm{T}}$ with

$$P_{\Omega}(\xi_i) = \begin{cases} \underline{\xi}_i, & \xi_i < \underline{\xi}_i, \\ \xi_i, & \underline{\xi}_i \leqslant \xi_i \leqslant \overline{\xi}_i, \\ \overline{\xi}_i, & \xi_i > \overline{\xi}_i. \end{cases}$$

This neural network is termed GPNN in [26] in order to emphasize that it is actually an extension of the PNN presented in [25,11].

Theorem 2. *Consider solving the ELQP problem* (1) *with X, Y defined in* (5) *by the GPNN* (11). *The state of the neural network is stable in the sense of Lyapunov and globally convergent to an equilibrium point, and the output is globally convergent to a solution of the problem.*

Proof. Consider the equivalent formulation (10) of the ELQP problem. Simple calculation yields

$$M_1^{\mathrm{T}}N_1 = \begin{pmatrix} P & D & 0 & 0 \\ -D^{\mathrm{T}} & Q & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix},$$

which is skew-symmetric and positive semi-definite as both P and Q are positive semi-definite. Then the theorem follows directly from Corollary 4 in [26]. □

Remark 1. Certainly, the GPNN (11) can be modified to solve the ELQP problem with box constraints only. It can be done by simply deleting the terms A, B, s, w, Ω_s, Ω_w in the model. The resulting neural network is

$$\frac{\mathrm{d}}{\mathrm{d}t}\begin{pmatrix} x \\ y \end{pmatrix} = \lambda\begin{pmatrix} I + P & -D \\ D^{\mathrm{T}} & I + Q \end{pmatrix}^{\mathrm{T}} \times \begin{pmatrix} \mathscr{P}_{\Omega_x}(x - (Px - Dy + p)) - x \\ \mathscr{P}_{\Omega_y}(y - (Qy + D^{\mathrm{T}}x + q)) - y \end{pmatrix}, \tag{12}$$

which turns out to be the model proposed in [23].

Remark 2. Also, the GPNN (11) can be applied to solve the ELQP problem with inequality constraints only. But such a neural network would be somewhat complicated which leaves spaces for improving. See Section 3.

Remark 3. Note that the equality constraints $Cx = c$, $Ey = e$ can be written as $c \leqslant Cx \leqslant c$, $e \leqslant Ey \leqslant e$. That is, the equality constraints can be unified into inequality constraints. Therefore, if equality constraints are present, one can also adopt the neural network (11) to solve the problem, e.g., the ELQP problem with X and Y defined exactly as in (3). However, in Section 4 another GPNN model will be presented which handles the equality constraints in a more efficient way.

3. ELQP with inequality constraints only

Now consider the ELQP problem with inequality constraints only, i.e.,

$$X = \{x \in \Re^n | Ax \in \Omega_s\}, \quad Y = \{y \in \Re^m | By \in \Omega_w\}. \tag{13}$$

For solving the problem, a variant of the GPNN (11) can be used with Ω_x, Ω_y replaced by $\Re^n, \Re^m$, respectively, which is of $n + m + h_1 + h_2$ dimensions. In the following I show that under some stronger conditions the problem can be solved by using a lower dimensional GPNN.

Firstly it is observed that the optimality conditions in (6) become

$$\begin{cases} Px^* - Dy^* - A^{\mathrm{T}}s^* + p = 0, \\ D^{\mathrm{T}}x^* + Qy^* - B^{\mathrm{T}}w^* + q = 0, \\ (s^*)^{\mathrm{T}}(s - Ax^*) \geqslant 0, \quad \forall s \in \Omega_s, \\ (w^*)^{\mathrm{T}}(w - By^*) \geqslant 0, \quad \forall w \in \Omega_w. \end{cases}$$

These equations can be rewritten as

$$\begin{pmatrix} P & -D \\ D^{\mathrm{T}} & Q \end{pmatrix}\begin{pmatrix} x^* \\ y^* \end{pmatrix} = \begin{pmatrix} A^{\mathrm{T}} & 0 \\ 0 & B^{\mathrm{T}} \end{pmatrix}\begin{pmatrix} s^* \\ w^* \end{pmatrix} - \begin{pmatrix} p \\ q \end{pmatrix}$$

and

$$\begin{pmatrix} s^* \\ w^* \end{pmatrix}^{\mathrm{T}}\left[\begin{pmatrix} s \\ w \end{pmatrix} - \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix}\begin{pmatrix} x^* \\ y^* \end{pmatrix}\right] \geqslant 0, \quad \forall s \in \Omega_s, \ w \in \Omega_w.$$

If P and Q are both positive definite, it is easily seen that the matrix $\left(\begin{smallmatrix} P & D \\ -D^{\mathrm{T}} & Q \end{smallmatrix}\right)$ is also positive definite and invertible. Hence we have

$$\begin{pmatrix} x^* \\ y^* \end{pmatrix} = \begin{pmatrix} P & -D \\ D^{\mathrm{T}} & Q \end{pmatrix}^{-1}\begin{pmatrix} A^{\mathrm{T}} & 0 \\ 0 & B^{\mathrm{T}} \end{pmatrix} \times \begin{pmatrix} s^* \\ w^* \end{pmatrix} - \begin{pmatrix} P & -D \\ D^{\mathrm{T}} & Q \end{pmatrix}^{-1}\begin{pmatrix} p \\ q \end{pmatrix} \tag{14}$$

and

$$(u^*)^{\mathrm{T}}(u - N_2u^* - b_2) \geqslant 0, \quad \forall u \in U_2, \tag{15}$$

where $u=(s^{\mathrm{T}}, w^{\mathrm{T}})^{\mathrm{T}}$ and

$$N_2=\begin{pmatrix}A & 0\\ 0 & B\end{pmatrix}\begin{pmatrix}P & -D\\ D^{\mathrm{T}} & Q\end{pmatrix}^{-1}\begin{pmatrix}A^{\mathrm{T}} & 0\\ 0 & B^{\mathrm{T}}\end{pmatrix},$$
$$b_2=\begin{pmatrix}-A & 0\\ 0 & -B\end{pmatrix}\begin{pmatrix}P & -D\\ D^{\mathrm{T}} & Q\end{pmatrix}^{-1}\begin{pmatrix}p\\ q\end{pmatrix},\quad U_2=\Omega_s\times\Omega_w.$$

Based on this equivalent formulation of the ELQP problem concerned in this section, the following GPNN is proposed to solve it:

- state equation:

$$\frac{\mathrm{d}u}{\mathrm{d}t}=\lambda(N_2+I)^{\mathrm{T}}(\mathscr{P}_{U_2}((N_2-I)u+b_2)-N_2u-b_2); \tag{16a}$$

- output equation:

$$v=\begin{pmatrix}P & -D\\ D^{\mathrm{T}} & Q\end{pmatrix}^{-1}\begin{pmatrix}A^{\mathrm{T}} & 0\\ 0 & B^{\mathrm{T}}\end{pmatrix}u-\begin{pmatrix}P & -D\\ D^{\mathrm{T}} & Q\end{pmatrix}^{-1}\begin{pmatrix}p\\ q\end{pmatrix}, \tag{16b}$$

where $u=(s^{\mathrm{T}}, w^{\mathrm{T}})^{\mathrm{T}}$, $v=(x^{\mathrm{T}}, y^{\mathrm{T}})^{\mathrm{T}}$ and $\lambda>0$. Clearly, this neural network is of h_1+h_2 dimensions. The convergence results are summarized below.

Theorem 3. *Consider solving the ELQP problem* (1) *with X, Y defined in* (13) *by the GPNN* (16). *Suppose that both P and Q are positive definite, then the state of the neural network is stable in the sense of Lyapunov and globally convergent to an equilibrium point, and the output is globally convergent to a solution of the problem. If furthermore* $\mathrm{rank}(A)=h_1$ *and* $\mathrm{rank}(B)=h_2$, *then both the state and output are globally exponentially convergent.*

Proof. The theorem follows from Corollary 4 in [26] and the following facts: (i) if P and Q are positive definite, then N_2 is positive semi-definite; (ii) if furthermore $\mathrm{rank}(A)=h_1$ and $\mathrm{rank}(B)=h_2$ then N_2 is positive definite. □

4. ELQP with inequality constraints and equality constraints

Consider the ELQP problems with both inequality and equality constraints, but without bound constraints, i.e.,

$$X=\{x\in\Re^n|Ax\in\Omega_s, Cx=c\},$$
$$Y=\{y\in\Re^m|By\in\Omega_w, Ey=e\}. \tag{17}$$

As stated in Remark 3, if we view the equality constraints as inequality constraints, a variant of GPNN (11), which would be of $n+m+h_1+h_2+r_1+r_2$ dimensions, can be used to solve the problem. Moreover, because the bound constraints $x\in\Omega_x$, $y\in\Omega_y$ are absent in the problem, a variant of GPNN (16), which is of $h_1+h_2+r_1+r_2$ dimensions, can be also used to solve the problem. In what follows an even lower dimensional GPNN will be designed to solve this particular problem.

First of all, by using the well-known saddle point theorem (just as we do in Theorem 1), it is not difficult to obtain the following results.

Theorem 4. *Let* $x^*\in\Omega_x$ *and* $y^*\in\Omega_y$. *Then* (x^*,y^*) *is a solution to problem* (1) *with X and Y defined by* (17) *if and only if there exists* $s^*\in\Re^{h_1}$, $w^*\in\Re^{h_2}$, $\rho^*\in\Re^{r_1}$, $\eta^*\in\Re^{r_2}$ *such that* $Ax^*\in\Omega_s$, $By^*\in\Omega_w$, *and*

$$\begin{cases}Px^*-Dy^*-A^{\mathrm{T}}s^*-C^{\mathrm{T}}\rho^*+p=0,\\ D^{\mathrm{T}}x^*+Qy^*-B^{\mathrm{T}}w^*-E^{\mathrm{T}}\eta^*+q=0,\\ Cx^*=c,\\ Ey^*=e,\\ (s^*)^{\mathrm{T}}(s-Ax^*)\geqslant 0,\quad \forall s\in\Omega_s,\\ (w^*)^{\mathrm{T}}(w-By^*)\geqslant 0,\quad \forall w\in\Omega_w.\end{cases} \tag{18}$$

Suppose that both P and Q are positive definite, $\mathrm{rank}(C)=r_1$ and $\mathrm{rank}(E)=r_2$. Let

$$G=\begin{pmatrix}P & -D\\ D^{\mathrm{T}} & Q\end{pmatrix}^{-1},\quad \hat{A}=\begin{pmatrix}A & 0\\ 0 & B\end{pmatrix},\quad \hat{C}=\begin{pmatrix}C & 0\\ 0 & E\end{pmatrix},$$
$$\hat{p}=\begin{pmatrix}p\\ q\end{pmatrix},\quad \hat{c}=\begin{pmatrix}c\\ e\end{pmatrix},\quad \theta=\begin{pmatrix}x\\ y\end{pmatrix},\quad u=\begin{pmatrix}s\\ w\end{pmatrix},$$
$$\xi=\begin{pmatrix}\rho\\ \eta\end{pmatrix}$$

and $U_3=\Omega_s\times\Omega_w$. Then (6) can be written as

$$\begin{cases}G^{-1}\theta^*-\hat{A}^{\mathrm{T}}u^*-\hat{C}^{\mathrm{T}}\xi^*+\hat{p}=0,\\ \hat{C}\theta^*=\hat{c},\\ (u^*)^{\mathrm{T}}(u-\hat{A}\theta^*)\geqslant 0,\quad \forall u\in U_3.\end{cases} \tag{19}$$

From the first equation in (19) we have

$$\theta^*=G\hat{A}^{\mathrm{T}}u^*+G\hat{C}^{\mathrm{T}}\xi^*-G\hat{p}.$$

Substituting θ^* into the second equation in (19) yields

$$\xi^*=-(\hat{C}G\hat{C}^{\mathrm{T}})^{-1}\hat{C}G\hat{A}^{\mathrm{T}}u^*+(\hat{C}G\hat{C}^{\mathrm{T}})^{-1}\hat{C}G\hat{p}+(\hat{C}G\hat{C}^{\mathrm{T}})^{-1}\hat{c}.$$

Substituting ξ^* into the expression of θ^* obtained above yields

$$\theta^*=G\hat{A}^{\mathrm{T}}u^*-G\hat{C}^{\mathrm{T}}(\hat{C}G\hat{C}^{\mathrm{T}})^{-1}\hat{C}G\hat{A}^{\mathrm{T}}u^*+G\hat{C}^{\mathrm{T}}(\hat{C}G\hat{C}^{\mathrm{T}})^{-1}\hat{C}G\hat{p}+G\hat{C}^{\mathrm{T}}(\hat{C}G\hat{C}^{\mathrm{T}})^{-1}\hat{c}-G\hat{p}.$$

Substituting this θ^* into the third equation in (19) gives a GLVI with respect to the variable u. Therefore, the following GPNN can be designed to solve the ELQP problem:

- state equation:

$$\frac{\mathrm{d}u}{\mathrm{d}t}=\lambda(N_3+I)^{\mathrm{T}}(\mathscr{P}_{U_3}((N_3-I)u+b_3)-N_3u-b_3); \tag{20a}$$

- output equation:

$$v=D_3u+d_3, \tag{20b}$$

where $\lambda>0$ and

$$D_3=G\hat{A}^{\mathrm{T}}-G\hat{C}^{\mathrm{T}}(\hat{C}G\hat{C}^{\mathrm{T}})^{-1}\hat{C}G\hat{A}^{\mathrm{T}},$$

$$d_3=G\hat{C}^{\mathrm{T}}(\hat{C}G\hat{C}^{\mathrm{T}})^{-1}(\hat{C}G\hat{p}+\hat{c})-G\hat{p},$$

$$N_3=\hat{A}D_3=\hat{A}W\hat{A}^{\mathrm{T}},\quad W=G-G\hat{C}^{\mathrm{T}}(\hat{C}G\hat{C}^{\mathrm{T}})^{-1}\hat{C}G,$$

$$b_3=\hat{A}G\hat{C}^{\mathrm{T}}(\hat{C}G\hat{C}^{\mathrm{T}})^{-1}(\hat{C}G\hat{p}+\hat{c})-\hat{A}G\hat{p}.$$

Clearly, this GPNN is of h_1+h_2 dimensions. The following results can be easily obtained.

Theorem 5. *Consider solving the ELQP problem* (1) *with X and Y defined in* (17) *by using the GPNN* (20). *If* $\mathrm{rank}(C)=r_1$, $\mathrm{rank}(E)=r_2$, $W=G-G\hat{C}^{\mathrm{T}}(\hat{C}G\hat{C}^{\mathrm{T}})^{-1}\hat{C}G$ *is positive semi-definite, then the state of the neural network is stable in the sense of Lyapunov and globally convergent to an equilibrium point, and the output is globally convergent to a solution of the problem.*

A critical condition in above theorem is how to ensure the positive semi-definiteness of W. See the following lemma from [14].

Lemma 1. *Let $H^T = H \in \Re^{n\times n}$, $P \in \Re^{m\times n}$. If $\mathrm{Rank}(P) = m$ and H is positive definite, then the matrix $H - HP^T(PHP^T)^{-1}PH$ is positive semi-definite.*

The above lemma implies that if $\mathrm{rank}(C) = r_1$, $\mathrm{rank}(E) = r_2$, $D = 0$, and P, Q are positive definite, then the conditions in Theorem 5 are met. But we are actually not very much interested in the case $D = 0$. This condition is imposed to ensure the symmetry of G and in turn the positive semi-definiteness of W via Lemma 1. However, as pointed out in Remark 4 of [14], when H is asymmetric while other conditions in Lemma 1 are valid, the conclusion of the lemma may still hold according to a variety of numerical tests. Nevertheless, to the best of the author's knowledge, this is still a hypothesis as a rigorous proof for that is unavailable. If this hypothesis holds, then the conditions in Theorem 5 can be reduced to: $\mathrm{rank}(C) = r_1$, $\mathrm{rank}(E) = r_2$, and P, Q are positive definite. A simulation example in Section 6 shows that the GPNN (20) correctly converges to a solution of the problem when $D \neq 0$.

5. Extended linear programming

Consider a special case of ELQP problem, the ELP problem [2], which is obtained by setting $p = q = P = Q = 0$, $D = -I, m = n$ in (1), i.e.,

$$\min_x \left\{ \max_y y^T x \right\} \quad \text{subject to } x \in X,\ y \in Y, \tag{21}$$

where X and Y are nonempty polyhedral sets. The ELP problem represents a class of linear programming problems in which the decision vector (denoted by y in (21)) varies in a set. Such a situation may be encountered in many real world applications where the standard linear programming are employed. However, many effective methods for solving linear programming problems such as Simplex method and Karmarkar's method cannot be used to solve the ELP problems.

In [24], Xia studied the ELP problem (21) with

$$X = \{x \in \Re^n | Ax \in \Omega_s\}, \quad Y = \Omega_y,$$

where the notations are defined as same as in (3), and proposed a recurrent neural network for solving it. In [7], He studied the ELP problem (21) with

$$X = \{x \in \Re^n | x \in \Omega_x, Ax \in \Omega_s\}, \quad Y = \Omega_y,$$

where the notations are defined as same as in (3), and formulated it into an GLVI problem in the form of (4). Based on this formulation, a GPNN was presented in [3] to solve the problem. In these studies [24,7,3], the decision vector y in the ELP problem is allowed to vary within a box set. If the GPNN designed in Section 2 is adopted, then both the variable x and the decision vector y are allowed to vary within general polyhedral sets. This property can be regarded as a significant advantage of the designed GPNN over existing neural networks for solving ELP problems. The stability and convergence results of the corresponding GPNN can be stated similarly as in Theorem 2. Interested readers may refer to [12] for details as well as other related results.

6. Numerical simulations

In this section I use a numerical example to illustrate the results. The simulations are conducted in MATLAB.

Consider an ELQP problem (1) with

$$P = \begin{pmatrix} 5 & 0 & -1 \\ 0 & 3 & 0 \\ -1 & 0 & 1 \end{pmatrix}, \quad D = \begin{pmatrix} 2 & 0 & 0 & 2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -4 & 0 \end{pmatrix},$$

$$Q = \begin{pmatrix} 2 & -3 & 0 & -1 \\ -3 & 10 & 0 & 0 \\ 0 & 0 & 2 & 3 \\ -1 & 0 & 3 & 10 \end{pmatrix}, \quad p = \begin{pmatrix} 4 \\ 8 \\ -8 \end{pmatrix}, \quad q = \begin{pmatrix} 5 \\ 5 \\ -5 \\ -5 \end{pmatrix}.$$

Let us first consider the case where X, Y are defined in (5). Let

$$A = \begin{pmatrix} 1 & 1 & 0 \\ 2 & 0 & -1 \\ -5 & 5 & -1 \end{pmatrix}, \quad B = \begin{pmatrix} 1 & 4 & 0 & 0 \\ 2 & 1 & 0 & 5 \\ 5 & 0 & 0 & -1 \\ 0 & 0 & 5 & -5 \end{pmatrix},$$

$$\Omega_x = \{x \in \Re^3 | x \geqslant 0\},$$
$$\Omega_y = \{y \in \Re^4 | y \geqslant 0\},$$
$$\Omega_s = \{s \in \Re^3 | -10 \leqslant s \leqslant 10\},$$
$$\Omega_w = \{w \in \Re^4 | -5 \leqslant w \leqslant 5\}.$$

The GPNN (11) is used to solve the problem. Since both P and Q are positive definite, according to Theorem 2, the neural network should be globally convergent to its equilibrium points. The numerical simulations validate this prediction. Actually, all simulations with different initial points in $\Re^{14}$ converge to a solution of the ELQP problem $((x^*)^T, (y^*)^T)^T = (-0.0000, -0.0000, 1.5296, -0.0000, -0.0000, 1.6177, 0.6177)^T$. Fig. 1 displays the output trajectories of the neural network with 10 different initial points when $\lambda = 1$.

Next we eliminate the bound constraints in the above problem. Then the constraint sets X and Y are defined by inequality constraints only; see (13). The GPNN (16) is used to solve the problem. All simulations converge to the equilibrium point $u^* = (-0.0000, 0.0000, 0.1026, 0.0000, -0.0000, 0.4962, -0.8713)^T$. Fig. 2 displays the state trajectories of the neural network with 10 different initial points when $\lambda = 1$. The corresponding solution of the ELQP problem is calculated as $((x^*)^T, (y^*)^T)^T = (-0.8596,$

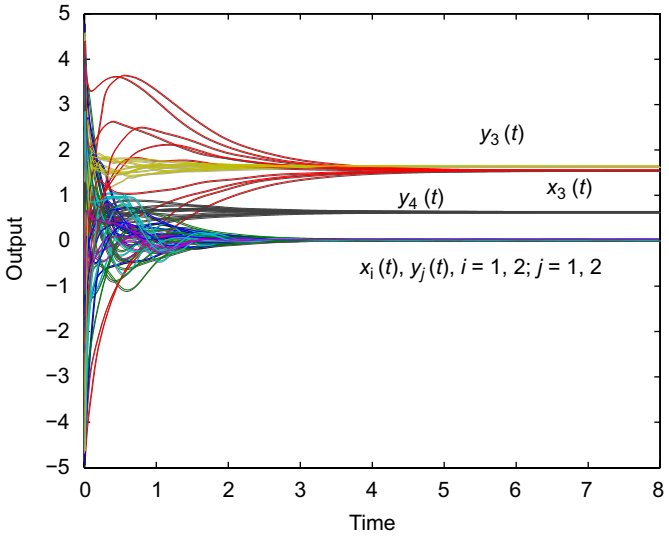

Fig. 1. Output trajectories of the GPNN (11) with 10 random initial points.

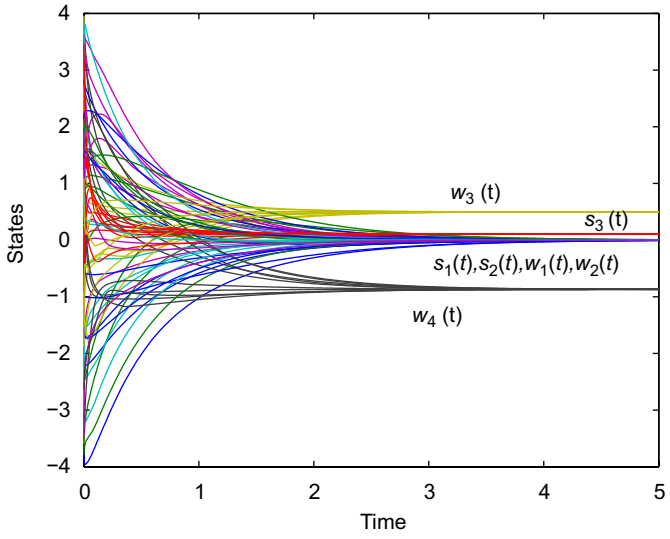

Fig. 2. State trajectories of the GPNN (16) with 10 random initial points.

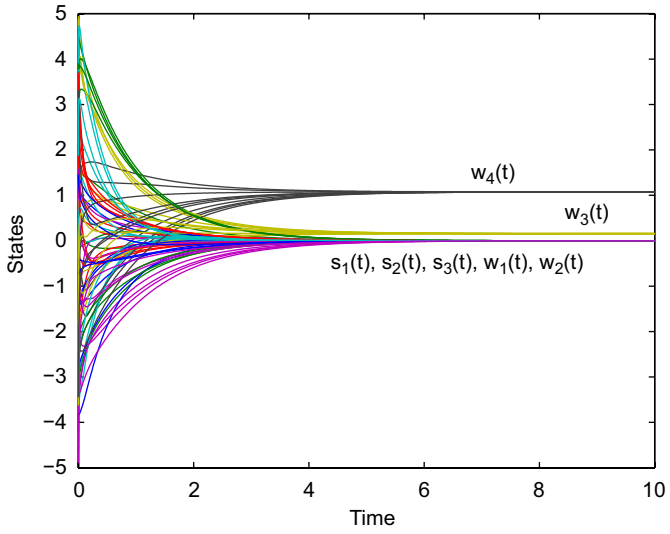

Fig. 3. State trajectories of the GPNN (20) with 10 random initial points.

$-2.6632, 0.9817, -0.8972, -0.5028, 1.5140, 0.5140)^{\mathrm{T}}$. Moreover, when the GPNN (11) is used to solve the problem, same solution is obtained.

Finally, we set X and Y as in (17) where A, B, Ω_s, Ω_w remain the same as above and

$$C = (1, 5, -6), \quad c = 2, \quad E = (-2, 0, 8, 3), \quad e = -7.$$

It can be readily checked that all conditions in Theorem 5 are valid. Then GPNN (20) can be used to solve the problem. All simulations converge to the equilibrium point $u^* = (-0.0001, 0.0002, 0.0001, 0.0000, -0.0002, 0.1561, 1.0753)^{\mathrm{T}}$. Fig. 3 displays the state trajectories of the neural network with 10 different initial points when $\lambda = 1$. The corresponding solution of the ELQP problem is calculated as $((x^*)^{\mathrm{T}}, (y^*)^{\mathrm{T}})^{\mathrm{T}} = (-0.9008, 0.3060, -0.2285, -1.0189, -0.8363, -1.0943, -0.0943)^{\mathrm{T}}$. Moreover, when the GPNN (11) is used to solve the problem, same solution is obtained.

7. Concluding remarks

In this paper I consider solving the extended linear-quadratic programming (ELQP) problem with general linear constraints by using the general projection neural network (GPNN), an existing recurrent neural network model in the literature. I have shown that with different combinations of bound constraints, equality constraints and inequality constraints, the ELQP problem can be always transformed into the so-called generalized linear variational inequality with box type constraints only. Thus the GPNN can be adopted for solving the corresponding problem. Moreover, in order to reduce the network complexity, much effort has been devoted to reduce their dimensions based on specific types of constraints. All of these designed GPNNs are globally convergent to the solutions of the corresponding ELQP problems under mild conditions. Finally, a numerical example was discussed to illustrate the performances of the designed GPNNs.

References

[1] M.S. Bazaraa, H.D. Sherali, C.M. Shetty, Nonlinear Programming: Theory and Algorithms, Wiley, New York, 1993.
[2] G.B. Dantzig, Linear Programming and Extensions, Princeton University Press, Princeton, 1980.
[3] X.B. Gao, A neural network for a class of extended linear variational inequalities, Chin. J. Electron. 10 (2001) 471–475.
[4] X.B. Gao, L.Z. Liao, A novel neural network for a class of convex quadratic minimax problems, Neural Comput. 18 (2006) 1818–1846.
[5] X.B. Gao, L.Z. Liao, W. Xue, A neural network for a class of convex quadratic minimax problems with constraints, IEEE Trans. Neural Networks 15 (2004) 622–628.
[6] F.M. Ham, I. Kostanic, Principles of Neurocomputing for Science and Engineering, McGraw-Hill, New York, 2001.
[7] B. He, Solution and applications of a class of general linear variational inequalities, Sci. China Ser. A 39 (1996) 397–404.
[8] B. He, H. Yang, A neural-network model for monotone linear asymmetric variational inequalities, IEEE Trans. Neural Networks 11 (2000) 3–16.
[9] J.J. Hopfield, D.W. Tank, Computing with neural circuits: a model, Science 233 (1986) 625–633.
[10] S.Q. Hu, D.R. Liu, On the global output convergence of a class of recurrent neural networks with time-varying inputs, Neural Networks 18 (2005) 171–178.
[11] X. Hu, J. Wang, Solving pseudomonotone variational inequalities and pseudoconvex optimization problems using the projection neural network, IEEE Trans. Neural Networks 17 (2006) 1487–1499.
[12] X. Hu, J. Wang, Solving extended linear programming problems using a class of recurrent neural networks, in: Proceedings of the ICONIP '06, Part II, Lecture Notes in Computer Science, vol. 4233, Springer, Berlin, Heidelberg, 2006, pp. 994–1003.
[13] X. Hu, J. Wang, Design of general projection neural networks for solving monotone linear variational inequalities and linear and quadratic optimization problems, IEEE Trans. Syst. Man Cybernet. Part B 37 (2007) 1414–1421.
[14] X. Hu, J. Wang, Solving generally constrained generalized linear variational inequalities using the general projection neural networks, IEEE Trans. Neural Networks 18 (2007) 1697–1708.
[15] D. Kinderlehrer, G. Stampcchia, An Introduction to Variational Inequalities and Their Applications, Academic, New York, 1980.
[16] Q. Liu, J. Cao, Y. Xia, A delayed neural network for solving linear projection equations and its analysis, IEEE Trans. Neural Networks 16 (2005) 834–843.
[17] R. Perfetti, E. Ricci, Analog neural network for support vector machine learning, IEEE Trans. Neural Networks 17 (2006) 1085–1091.
[18] L. Qi, R.S. Womersley, An SQP algorithm for extended linear-quadratic problems in stochastic programming, Ann. Oper. Res. 56 (1995) 251–285.
[19] R.T. Rockafellar, Computational schemes for large-scale problems in extended linear-quadratic programming, Math. Program. 48 (1990) 447–474.
[20] R.T. Rockafellar, R.J.-B. Wets, A Lagrangian finite-generation technique for solving linear-quadratic problems in stochastic programming, Math. Program. Stud. 28 (1986) 63–93.
[21] J. Sun, J. Zhu, G. Zhao, A predictor-corrector algorithm for a class of nonlinear saddle point problems, SIAM J. Control Optim. 35 (1997) 532–551.
[22] D.W. Tank, J.J. Hopfield, Simple neural optimization networks: an A/D converter, signal decision circuit, and a linear programming circuit, IEEE Trans. Circuits Syst. 33 (1986) 533–541.

[23] Q. Tao, T.J. Fang, The neural network model for solving minimax problems with constraints, Control Theory Appl. 17 (2000) 82–84 (in Chinese).
[24] Y. Xia, Neural network for solving extended linear programming problems, IEEE Trans. Neural Networks 8 (1997) 803–806.
[25] Y. Xia, H. Leung, J. Wang, A projection neural network and its application to constrained optimization problems, IEEE Trans. Circuits Syst. I 49 (2002) 447–458.
[26] Y. Xia, J. Wang, A general projection neural network for solving monotone variational inequalities and related optimization problems, IEEE Trans. Neural Networks 15 (2004) 318–328.
[27] Y. Yang, J. Cao, A delayed network method for solving convex optimization problems, Int. J. Neural Syst. 16 (2006) 295–303.
[28] Y. Yang, J. Cao, Solving quadratic programming problems by delayed projection neural network, IEEE Trans. Neural Networks 17 (2006) 1630–1634.

Xiaolin Hu received the B.E. and M.E. degrees in automotive engineering from Wuhan University of Technology, China, and the Ph.D. in Automation & Computer-Aided Engineering from The Chinese University of Hong Kong, Hong Kong, China, in 2001, 2004, 2007, respectively. He is now a Postdoctoral Fellow at Tsinghua National Lab for Information Science & Technology, State Key Lab of Intelligent Technology & Systems, and Department of Computer Science & Technology, Tsinghua University, Beijing 100084, China. His current research interests include artificial neural networks, evolutionary computation, computational neuroscience and multimedia processing.