A Recurrent Neural Network for Real-Time Computation of Semidefinite Programming

Danchi Jiang, Jun Wang
Department of Mechanical and Automation Engineering
Chinese University of Hong Kong
Shatin, N. T., Hong Kong
dcjjiang@mae.cuhk.edu.hk  jwang@mae.cuhk.edu.hk

Abstract
This paper proposes a novel recurrent neural network for the real-time computation of semidefinite programming. This network is developed to minimize the duality gap between the admissible points of the primal problem and the corresponding dual problem. By appropriately defining an auxiliary cost function, a modified gradient dynamical system can be obtained which ensures an exponential convergence of the duality gap. Then, two subsystems are developed to avoid the difficulties involving matrix inverse and determinant, so that the resulted dynamical system can be easily realized using an analog recurrent neural network. The architecture of the resulting neural network is also discussed. The operating characteristics and performance of the proposed approach are demonstrated by means of simulation results. The approach reported in this paper not only gives an promising way for real-time computation of semidefinite programming, but also offers several new insights for its numerical computation.

1. Introduction

Consider the semidefinite programming problem defined as:

\[ \begin{array}{c}
\min_{x} J_p(x) = C^T x, \\
\text{s.t. } F(x) := F_0 + F_1 x_1 + F_2 x_2 + \ldots + F_n x_n \succeq 0 \quad (2)
\end{array} \]

where \( x = (x_1, x_2, \ldots, x_n)^T \in \mathbb{R}^n, \quad x = (C_1, C_2, \ldots, C_n)^T, \quad F_i \in \mathcal{S}(m) \subseteq \mathbb{R}^{m \times m}, \mathcal{S}(m) \) denotes the set of all \( m \times m \) symmetric matrices, and \( F_i \), \( i = 0, 1, 2, \ldots, n \) are linearly independent. The constraint (2) means that \( F(x) \) is positive semidefinite.

This problem can be widely applied to control design, combinatorial optimization, and other areas. For example, see [1, 3, 4, 9]. There are many numerical algorithms reported to solve this problem and the literature is still quickly growing. See [4, 6, 8, 9, 10] and references cited therein. This paper will propose a recurrent neural network for real-time computation of it. Recurrent neural networks have been successfully applied to the real-time computation of many optimization problems. See [11, 12] for some examples. One of the main ideas is to build a recurrent neural network such that the state of the neural network represents the decision vector of the optimization problem and the dynamics of the neural network reflects certain gradient descent property of the corresponding cost function. If the dynamics is realized using an analog circuit, the steady state can be reached quickly. In addition to this intuition, the effectiveness of the neural network developed in this paper is also established by theoretical results and demonstrated by simulation results.

1.1 Duality Gap

The dual problem of the problem defined by (1) and (2) can be defined [8, 9] as:

\[ \begin{array}{c}
\max_{Z} J_d(Z) = -\text{tr}(F_0 Z) \\
\text{subject to } \text{tr}(F_i Z) = C_i, \quad i = 1, 2, \ldots, n; \\
Z \succeq 0, \quad Z \in \mathcal{S}(m).
\end{array} \]

To simplify the subsequent analysis, we need the following two assumptions: A-1 The cost function \( J_p(x) \) is bounded below on the admissible set defined by \( \{x \in \mathbb{R}^n : F(x) \succeq 0\} \). A-2 Both the primal problem defined by (1), (2) and the corresponding dual problem defined by (3), (4), (5) are strictly feasible; i.e., there exist a vector \( x \) and a positive definite matrix \( Z \) such that \( F(x) \succ 0, \quad \text{tr}(F_i Z) = C_i, \quad i = 1, 2, \ldots, m \).

The first assumption is concerned with the consistency of the problem. In the case that this assumption
is not satisfied, the minimum of the cost is negative infinity. The second assumption is not a restriction on a semidefinite programming problem either. In fact, many techniques have been reported that can convert a semidefinite programming problem into a primal and dual strictly admissible one. For example, one can obtain a primal and dual strictly admissible semidefinite programming problem by the “big-M” method reported in [2]. It is worth to mention that by using the “big-M” method, one can also obtain an admissible solution to the original semidefinite problem if there exists one. It is shown in [8] that the following duality theorem holds:

**Theorem 1 (Duality Theorem) [8, page 109]** If the assumptions A-1 and A-2 are satisfied, then, an admissible pair \((x^*, Z^*)\) of the primal and dual problems defined respectively by (1), (2), and (3)/(4), (5) respectively is optimal if and only if \(\text{tr}(F(x)Z) = 0\).

\(\text{tr}(F(x)Z)\) is termed as a duality gap. The optimal solution can be computed by minimizing the duality gap since it is always non-negative at any admissible point. Indeed, this idea is used in many studies such as those reported in [8,10]. One can justify this approach by the following analysis.

Note that the dual problem defined by (3)/(4), and (5) can also be converted into the following form:

\[
\begin{align*}
\min & \quad \bar{J}_d = D^T z \\
\text{s.t.} & \quad G(z) := G_0 + G_1 z_1 + \ldots + G_l z_l \geq 0,
\end{align*}
\]

where the following relations hold:

\[
\begin{align*}
J_d &= -J_d - \text{tr}(F_0 G_0), \\
D &= \left(\text{tr}(F_0 G_1), \text{tr}(F_0 G_2), \ldots, \text{tr}(F_0 G_l)\right)^T, \\
\text{tr}(F_k G_0) &= C_i, i = 1, 2, \ldots, n, \\
\text{tr}(F_k G_j) &= 0, \quad i = 1, 2, \ldots, n, \quad j = 1, 2, \ldots, l.
\end{align*}
\]

The duality gap, denoted as \(\text{Gap}\) in the rest of this paper, now becomes:

\(\text{Gap} := \text{tr}(F(x)G(z)) = \text{tr}(F_0 G_0) + C^T x + D^T z\).

Therefore, the primal and dual problems defined respectively by (1), (2) and (3)/(4), (5) can be solved simultaneously by solving the following problem:

\[
\text{minimise} \quad J_c(x, z) = \text{Gap} \quad \text{subject to} \quad F(x) \geq 0, \quad G(x) \geq 0
\]

This problem is larger than primal problem and its dual problem in terms of the dimension of the corresponding decision vector. However, it is known that its optimal cost value is zero, which provides an advantage for the development of appropriate optimization algorithms and the corresponding theoretical analysis.

### 1.2 Central Trajectory

Consider the minimization of the following cost function:

\[ J_\beta(x, z) = \beta \text{Gap} - \log \det(F(x)G(z)), \quad (12) \]

where \(\beta \geq 0\). This cost function is strictly convex. As \(\beta\), varies, the optimal solution of the unconstrained problem (12) defines a curve. This curve is called a central trajectory. Note that, as \(\beta\) tends to \(+\infty\), the central trajectory converges to the optimal solution pair of the primal and dual problems defined by (1), (2) and (3), (4), (5), respectively.

Based on the analysis of this problem, we obtain the following two properties concerning the central trajectory.

**Lemma 1** Consider the combined problem defined by (10) and (11).

1. An admissible pair \((x, z)\) is on the central trajectory if and only if

\[
F(x)G(z) = \frac{\text{Gap}}{m} I.
\]

(13)

2. For any admissible pair \((x, z)\)

\[
m^m \det(F(x)G(z)) \leq [\text{Gap}]^m.
\]

(14)

Proof. Omitted for brevity.

### 2 A Modified Gradient System

#### 2.1 An Auxiliary Cost Function

Introduce an auxiliary cost function as follows:

\[
\Phi(x, z) = \frac{[\text{Gap}]^{m+\nu}}{\det(F(x)G(z))^{m^m}}
\]

(15)

for \(x \in \mathbb{R}^n, z \in \mathbb{R}^l\), such that \(F(x) > 0, G(z) > 0\), \(\nu > 0\) is an adjustable parameter. This auxiliary cost function has the following interesting properties:

**Lemma 2** Consider the auxiliary cost function defined by (15).

1. \(\Phi(x, z) \geq 0\).
2. \(\Phi(x, z) \geq [\text{Gap}]^\nu\).
3. If there is a sequence \((x^k, z^k)\) such that

\[
\Phi(x^k, z^k) \to 0,
\]

then this sequence converges to the optimal solution set of the semidefinite programming problem.
Proof. Clearly, by the positive definite property of $F(x), G(z)$ and (14) in Lemma 1 one can obtain (1) and (2). The last result is a direct conclusion of (2).

Based on these results in Lemma 2, we can search for an optimal solution by minimizing this auxiliary cost function. Note that any optimal solution to the semidefinite programming problem is a global one because the cost is convex. The gradient $\nabla \Phi(x, z)$ can be calculated by the techniques in [7] as

$$\Phi(x, z)\{\rho \begin{pmatrix} C \\ D \end{pmatrix} - \begin{pmatrix} \text{tr}(F(x))^{-1}F_1 \\ \text{tr}(F(x))^{-1}F_2 \\ \vdots \\ \text{tr}(G(z))^{-1}G_1 \\ \text{tr}(G(z))^{-1}G_2 \\ \vdots \\ \text{tr}(G(z))^{-1}G_1 \end{pmatrix} \} \right)$$

where

$$\rho := \frac{m + \nu}{\text{Gap}}.$$ 

2.2 A Neural Network Model

The minimal point of the auxiliary function (15) might be a local one since the function $\Phi(x, z)$ is not a convex function. Furthermore, in the case that the global optimal solution can be obtained, the corresponding algorithm converges only linearly or quadratically. In order to obtain the global optimal solution efficiently, we propose a modified gradient system to compute the semidefinite programming problem as follows:

$$\begin{pmatrix} \dot{x} \\ \dot{z} \end{pmatrix} := -\gamma \frac{\nabla \Phi}{||\nabla \Phi||^2} \Phi^{1+\alpha},$$

for $F(x) > 0, G(z) > 0,$

where $||.||$ is the Frobenius norm (i.e., $||\nabla \Phi||^2 = [\nabla \Phi]^T \nabla \Phi$), and $\alpha \geq 0, \gamma > 0$ are any parameters that can be chosen freely. Later in this paper one can see that $\alpha = 0$ is preferred.

In fact, the auxiliary cost function $\Phi$ is an exponential function of the primal-dual potential reduction cost function addressed in [8, 10]. It is known [10, Theorem 3] that for any strictly feasible point $(x, x)$, $\nabla \Phi \neq 0$. Therefore, the definition of (17) is justified.

We extend the definition of (17) to more general cases as follows:

$$\begin{pmatrix} \dot{x} \\ \dot{z} \end{pmatrix} := -\gamma \frac{\Psi(\text{Gap})}{||\nabla \Phi||^2} \Phi^{1+\alpha}, \tag{18}$$

where $\Psi$ is defined as

$$\Psi(s) = \begin{cases} 1 & \text{for } s > 0, \\ 0 & \text{for } s \leq 0. \end{cases} \tag{19}$$

In fact, we consider the function $\Psi$ is a switch. More specifically, when Gap = 0, we let the right hand side of (18) to be zero without computing $\nabla \Phi$, which might be not well defined.

Within any interval such that Gap $> 0$ the time derivative of $\Phi(t)$ can be calculated as

$$\dot{\Phi} = \frac{\partial \Phi}{\partial (x^T, z^T)} \left( \begin{array}{c} \dot{x} \\ \dot{z} \end{array} \right) = -\gamma \Phi^{1+\alpha}. \tag{20}$$

Hence the solution to the differential equation (20), denoted as $\Phi(t)$ for brevity, is solved as

$$\Phi(t) = \begin{cases} \Phi(0)e^{-\gamma t} & \text{if } \alpha = 0, \\ \left\{ \left[ \Phi(0) \right]^{-\alpha} + \alpha \gamma t \right\}^{\frac{1}{\alpha}} & \text{if } \alpha > 0. \end{cases} \tag{21}$$

Therefore, by applying Lemma 2 we know that the solution of (18) exists for all time $t \in [0, +\infty)$. Hence the following convergence property of the model (18) holds:

**Theorem 2 (Convergence Property)** Suppose $(x(0), z(0))$ is an arbitrary initial point such that $F(x(0)) > 0, G(z(0)) > 0.$

1. There exists a unique solution for the equation (18) in the time interval $[0, +\infty)$. 

2. Along the trajectory of (18) starting from $(x(0), z(0))$,

$$F(x(t)) > 0, \ G(z(t)) > 0,$$ 

and the cost function $\Phi(t)$ strictly decreases until an optimal solution to the combined problem given by (10) and (11) is achieved; i.e., one of the inequalities in (22) violated and the duality gap becomes zero.

3. If $\alpha = 0$, along the trajectory there holds:

$$\text{tr}(F(x(t))G(x(t))) \leq \left[ \Phi(t) \right]^\frac{1}{2} = [\Phi(0)]^\frac{1}{2} e^{-\frac{\gamma}{2} t};$$

i.e., the duality gap converges to zero exponentially. If $\alpha > 0$, then

$$\text{tr}(F(x(t))G(x(t))) \leq [m^m \Phi(t)]^\frac{1}{\bar{\alpha}} = [\alpha \gamma]^{1-\bar{\alpha}},$$

where

$$\bar{\alpha} = \frac{\alpha}{\nu}, \quad \bar{\gamma} = [\alpha \gamma]^{-\bar{\alpha}}.$$

**Proof.** (1) Only two cases can happen:

Case 1, Gap $> 0$ along the trajectory of (18). Then, $(x, z)$ remains strictly feasible, the solution to (18) exists in the interval $[0, +\infty)$ and $\Phi$ satisfies (20) all the time.

1642
Case 2. \textbf{Gap} \ > 0 will be violated at some time. Then, there is a time interval \([0, b)\) such that \(\text{Gap} \ > 0\) for \(t \in [0, b)\) and \(\text{tr}(F(x(t))G(z(t))) = 0\). Then, \(x(t) = x(b), z(t) = x(b)\) for \(t > b\). (1) is established.

(2) and (3) can be clearly seen from the argument for (1) and the formula of \(\Phi\) given by (21).

3 Realization of the Neural Network Model

Note that the modified gradient flow (18) is, in fact, not ready yet for realization as an analog neural network because of the difficulties in the computation of the determinant and inverse of \(F(x)\) and \(G(x)\) contained in the gradient and \(\Phi\). In this section, we will discuss how to cope with those difficulties and present a neural network for the computation of semidefinite programming.

3.1 Computation of the Inverse of an Time-Varying Matrix

Instead of computing the inverse \((F(x)G(z))^{-1}\) or \((F(x)Y^{-1})_w,\) we construct a recurrent neural network to compute \([\text{Gap}]^{m+v}[F(x)G(z)]^{-1}\). The reason lies in the following result:

\textbf{Lemma 3} Along the trajectory of (18) starting from any strictly admissible pair \((x_0, z_0)\),

\begin{equation}
\lim_{t \to b^-} [\text{Gap}]^{m+v}[F(x)G(z)]^{-1} = 0, \tag{23}
\end{equation}

where \(b\) is \(+\infty\) or a positive real number corresponding to Case 1 or Case 2, respectively, in the proof of (1) of Theorem 2.

\textbf{Proof} Let the eigenvalues of \([F(x(t))G(z(t))])\) be denoted as \(\lambda_1(t) \leq \lambda_2(t) \leq \ldots \leq \lambda_m(t)\). Since

\[F(x)G(z) = [F(x)]^\frac{1}{2}([F(x)]^\frac{1}{2}G(z)[F(x)]^\frac{1}{2})[F(x)]^{-\frac{1}{2}},\]

all eigenvalues of \(F(x)G(z)\) are simple and positive. Then, from (14) we know that

\[\text{Gap}^{m+v} \lambda_1 = [\text{Gap}]^{m+v} \frac{\prod_{i=2}^{m} \lambda_i}{\det(F(x)G(z))} = m^{m} \Phi(x, z) \prod_{i=2}^{m} \lambda_i \leq m^{m} \Phi(x, z)[\text{Gap}^{m-1} \to 0, \text{as } t \to b^-].\]

Since \([\text{Gap}]^{m+v} \lambda_1\) is the largest eigenvalue of the matrix \([\text{Gap}]^{m+v}[F(x)G(z)]^{-1}\), the proof is complete. \(\blacksquare\)

Let the matrix \([\text{Gap}]^{m+v}[F(x)G(z)]^{-1}\) be denoted as \(W(t)\). Then, its derivative with respect to time is calculated as:

\begin{equation}
\dot{W}(t) = \frac{m + v}{\text{tr}(F(x(t))G(z(t)))} (C^T \dot{x} + I^T \dot{z}) W(t) - \text{Gap}^{-(m+v)} \sum_{i=1}^{n} W(t) F_i G(z(t)) W(t) \dot{x}_i - \text{Gap}^{-(m+v)} \sum_{j=1}^{n} W(t) G_j W(t) \dot{z}_j, \tag{24}\end{equation}

where \(\dot{x}, \dot{z}\) are given by (18). The time-varying matrix \(W(t)\) can also be determined by using the dynamical system (24) and the initial condition:

\begin{equation}
W(0) = [\text{tr}(F(x_0)G(z_0))]^{m+v}(F(x_0)G(z_0))^{-1}. \tag{25}\end{equation}

Furthermore, we can also compute \(\dot{x}\) and \(\dot{z}\) by the following formula which is equivalent to (18):

\begin{equation}
\begin{pmatrix} \dot{x} \\ \dot{z} \end{pmatrix} = -\gamma \Psi(\text{Gap}) \text{Gap}^{m+v} V(t) \left( \frac{\Phi(t)}{||V(t)||^\alpha} \right)^n, \tag{26}\end{equation}

\begin{equation}
\Phi(t) = -\gamma \Psi(\Phi(t))^{1+\alpha}, \quad \Phi(0) = \Phi(x_0, z_0), \tag{27}\end{equation}

where

\begin{equation}
\begin{pmatrix} \dot{x} \\ \dot{z} \end{pmatrix} = -\gamma \Psi(\text{Gap}) \text{Gap}^{m+v} V(t) \left( \frac{\Phi(t)}{||V(t)||^\alpha} \right)^n - \frac{\text{tr}(F(z(t))W(t)F_1)}{\text{tr}(W(t)F(x(t))G_1)} \end{pmatrix}. \tag{28}\end{equation}

3.2 Neural Network Architecture

Based on the previous discussion, we can build a neural network associated with the dynamical system (24), (26), and (27). For this recurrent neural network, the computation of the determinant \(\det(F(x(t))G(z(t)))\) and the matrix inverses \((F(x))^{-1}, G(z)^{-1}\) are avoided by computing \(\Phi\) and \(W\) using their dynamical systems. See Figure 1 for a simplified block diagram. The corresponding electronic circuits of those sub-networks can be easily obtained because the computation is only a combination of addition, subtraction and multiplication of \(x, z\) and a switch circuit realizing the binary (hard-limiting) function. It is worthy noting that the initial condition for \(x, z\) and \(W, \Phi\) are not independent. More specifically,

\begin{equation}
\Phi_0 = \frac{(\text{tr}(F(x_0)G(z_0)))^{m+v}}{\det(F(x_0)G(z_0))} W_0, \tag{29}\end{equation}

where

\[W_0 = \frac{(\text{tr}(F(x_0)G(z_0)))^{m+v}(F(x_0))^{-1}}{\det(F(x_0)G(z_0))} - \text{Gap}^{-(m+v)} \sum_{j=1}^{n} W(t) G_j W(t) \dot{z}_j, \]
4 Simulation Results

In this section, simulation results are given to demonstrate the efficiency of the proposed approach. First, we generate a problem randomly. For the illustration purpose, we choose the dimension of the positive definite matrices as 3 and the number of the primal decision variables as 4. Then we randomly generate an admissible primal semidefinite problem and use the “big-M” method to obtain a dual admissible problem.

In order to demonstrate the convergence property of the proposed neural network, we solve the trajectory of the modified gradient system (18) numerically. The simulation result is plotted in Figures 2. One can see that the duality gap rises first then converges to zero exponentially. The increasing is the effect of the determinant factor in the auxiliary cost function (15). Simulation also shows that the primal and dual variables converge to their steady states quickly. In the auxiliary cost function $\Phi_1, \nu$ is a weighting factor that adjustable. It is possible that we can keep the duality gap to be positive all the time by adjusting this parameter. Our simulation experiments suggest that this can happen.

As mentioned before, $W(t)$ is used instead of $(F(x)G(x))^{-1}$ to avoid the computation of the inverse of a matrix that is close to singular. To demonstrate the effect of this idea, we have also computed the trajectory of the modified gradient system (18) directly. It is much slower to obtain a solution with respect to the same precision required in the computation described above, as shown in Figures 3. This is due to the error generated in the computation of inverses of near-singular matrices.

In the analysis of our approach, we assume that the initial condition of $W(t)$ and $\Phi(t)$ are precisely the results of formulas (29). In practice, we can only get an approximate value of it. To demonstrate the effectiveness of our approach, we have done a simulation with one percent error in the initial condition. The simulation results show in the Figures 4 that the computation error of the initial condition of $W$ and $\Phi$ can be neglected. It would be an interesting topic for further investigation concerning the theoretical aspects of this phenomenon.

5 Concluding Remarks

We have proposed a recurrent neural network approach to semidefinite programming. We first constructed an auxiliary cost function reflecting both the minimization of the duality gap and the satisfaction of the linear matrix inequality constraint. Then, we proposed a modified gradient dynamical system based on this auxiliary cost function. Following any trajectory of this system starting from any strictly primal-dual feasible point, the duality gap converges to zero exponentially. Then, two subsystems are introduced so that the resulted dynamical system can be easily re-
Perturbation on the duality gap along the trajectory

Figure 4. Duality gap converges to zero exponentially along the trajectory of the dynamical system defined by (24), (26) and (27) where initial condition of \(W\) and \(\Phi\) have one percent perturbation.

alized using an analog recurrent neural network. The effectiveness and efficiency of the resulted neural system are demonstrated by means of simulation results.

In the development of the two subsystems of \(W\) and \(\Phi\) to avoid the computation of matrix inverses and determinant, the initial states are determined by the initial primal and dual decision variable of the original semidefinite programming problem. We assume that their initial states are precisely computed. The effect of the computation error of the initial condition has been illustrated unharmful by simulation results.

In the process of developing our approach, we can also obtain some new insights into the numerical computation of semidefinite programming. First, since in the solution process using a primal-dual approach the trace of the matrix \(F(x)G(z)\) will converge to zero, either \(F(x)\) or \(G(z)\) will tend to a singular matrix. This implies that the computation of the inverse of the near-singular matrix will become ill-conditioned. There has been no report yet to discuss how this may affect the computation of a semidefinite programming problem and how to handle it. In the light of this paper, we can see this issue can be avoided by using \(W\) instead of \((F(x))^{-1}\) and \((G(z))^{-1}\). Another insight that we can obtain in the light of our proposed approach is that, for the inner algorithm of a path-following or potential-reduction method, the Newton’s step is computed based on the expensive computation of the matrix inverses \((F(x))^{-1}\) and \((G(z))^{-1}\), the discretization of the dynamical system of \(x(t), x(t), W(t), \Phi(t)\) in this paper clearly suggests an iterative procedure. It would be a very interesting issue to analyze the property of the corresponding algorithm.

References


