A Sublinear Algorithm for Barrier-Certificate-Based Data-Driven Model Validation of Dynamical Systems

Shuo Han, Ufuk Topcu, George J. Pappas

***Abstract*— The paper considers the problem of scaling the method of barrier certificates for data-driven validation of dynamical system models using a large number of collected trajectories. Construction of a barrier certificate requires solving a convex feasibility problem that consists of a set of affine constraints whose number grows with the size of the dataset. The time complexity of traditional methods such as the interior-point method depends at least linearly on the size of the dataset and can be expensive to use for large datasets. We show that sublinear time complexity can be achieved using the multiplicative weights method, which was originally proposed to compute an approximate feasible solution for affine constraints. After modifications, the multiplicative weights method is able to yield an exact solution to the convex feasibility problem and hence a valid barrier certificate. We also present numerical studies and show that the multiplicative weights method is favorable to traditional methods for large datasets.**

I. Introduction

The use of "big data" has become increasingly important in cyber-physical systems due to the advances in sensing and data collection technology [12]. On one hand, more data allow us to gain more insight into the system and hence make better decisions in system operation for improved performance. On the other hand, the presence of large datasets also brings computational challenges; this is often manifested in the size of the data-driven optimization problem, which grows with the size of the dataset [2].

In this paper, we consider the problem of model validation for dynamical systems [10]. Suppose we are given a model in the form of a parameterized class of dynamical systems and some information on the trajectories of the actual system. The goal is to check whether knowledge about the trajectories is consistent with the model, i.e., there exists a system within the parameterized class that can explain the behaviors of the trajectories. In previous work on model validation, knowledge about the trajectories is often expressed in the form of set constraints, such as sets that contain the initial/terminal states of the trajectory. In contrast, we focus on a data-driven variant of the problem, where we assume that we have access to a dataset of trajectories collected from the actual physical system.

Past work, in particular the pioneering work by Prajna [10], has shown that the system model validation problem can be solved using the technique of barrier certificates. Strictly speaking, the technique of barrier certificates is used to invalidate the system model; if a barrier certificate can be constructed, then the class of system models is invalidated by knowledge about the trajectories, i.e., there is no model within the class that is able to explain the behaviors of the trajectories. When the technique of barrier certificates is applied to solve the data-driven variant of model validation, it can be shown that one needs to solve a feasibility problem that involves a single convex constraint (usually linear matrix inequality) and a set of affine constraints whose number depends on the number of trajectories in the dataset.

When the dataset is relatively small, the convex feasibility problem of finding a barrier certificate is typically solved using the interior-point method for fast convergence and good numerical precision. However, the time complexity of the interior-point method grows at least linearly with the number of constraints [7], hence also linearly with the size of the dataset. For large datasets, the interior-point method can be expensive to use in practice, which motivates the need for a numerical solution method that achieves sublinear time complexity.

A number of sublinear algorithms have been developed in the area of constrained convex optimization (which generalizes convex feasibility problems). The idea of sketching has been applied to regression problems, where the algorithm works by projecting the problem data into a lower dimensional space [14]. There has also been work that uses sampling-based techniques to avoid handling all the data, such as the work of de Farias and Van Roy [4] with application in approximate dynamic programming and the work of Calafiore and Campi [3] on solving uncertain convex programs. In this paper, we explore the application of the multiplicative weights algorithm [1], one of whose applications is solving affine feasibility problems with a large number of constraints.

Contributions: We show that the data-driven model validation problem can be solved by searching for a feasible solution to a convex constraint and a number of affine constraints generated by the trajectory data. This feasibility problem can be solved by the multiplicative weights method, whose (worst-case) time complexity grows only logarithmically with the size of the dataset. Although the multiplicative weights algorithm was originally proposed to obtain an approximate solution to affine feasibility problems, we show that, by implementing the algorithm properly, an exact solution and hence a valid barrier certificate can be obtained. In addition, through simulations, we show that the actual running time of the multiplicative weights algorithm is

The authors are with the Department of Electrical and Systems Engineering, University of Pennsylvania, Philadelphia, PA 19104. {hanshuo,utopcu,pappasg}@seas.upenn.edu. This work was supported in part by the NSF (CNS-1239224) and TerraSwarm, one of six centers of STARnet, a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

very insensitive to size of the dataset. In the regime of large datasets, the multiplicative weights algorithm is significantly faster than the interior-point method.

Paper organization: We introduce the problem of data-driven model validation in Section II. This has a similar description as the one used by Prajna [10]; the difference is that our formulation explicitly incorporates available trajectory data. We then briefly review the technique of barrier certificates in Section III. We show that finding a barrier certificate requires solving a convex feasibility problem with a number of affine constraints generated by trajectory data. Section IV presents the main result of the paper, where we show that the multiplicative weights algorithm can be used to solve the feasibility problem in sublinear time (in terms of the size of dataset). In the end, we present in Section V a numerical example that shows the multiplicative weights method is favorable to conventional direct solution methods in the regime of large datasets.

II. Problem statement

A. Notation

Denote the ℓ_2-norm of any $x \in \mathbb{R}^n$ by $\|x\|$. The vector consisting of all ones is written as $\mathbf{1}$. The symbol $\succeq$ is used to represent element-wise inequality: for any $x, y \in \mathbb{R}^n$, we have $x \succeq y$ if and only if $x_i \geq y_i$ for all $1 \leq i \leq n$. All the sets are represented by uppercase calligraphic letters.

B. Problem statement: System model validation

The problem setting we consider in this paper is similar to the model validation problem presented in the work of Prajna [10]. The model under consideration is a class of continuous-time nonlinear dynamical systems described by

$$\dot{x}(t) = f(x(t), p, t), \tag{1}$$

where $x(t)$ is the system state, t is the time, and p is some fixed parameter that appears in the dynamics. Suppose that we do not know the exact value of p. Instead, we are presented with a dataset consisting of m trajectories collected from the actual system for $t \in [0, T]$, where each trajectory starts with initial state $x(0) = x_0^{(i)}$ and ends at terminal state $x(T) = x_T^{(i)}$ $(i = 1, 2, \ldots, m)$. In this paper, we assume for simplicity that we are only able to record the initial and terminal states, but not any intermediate points along the trajectory $x(t)$. The case where intermediate points are also recorded can be handled in a similar way using the notion of extended vector fields [10]. For simplicity, we do not consider imperfections such as measurement noise in the trajectory data.

In addition to the initial and terminal states, we assume that we have prior knowledge about the system trajectories: there exists a set $\mathcal{X}$ such that

$$x(t) \in \mathcal{X}, \quad \forall t \in [0, T]. \tag{2}$$

The set $\mathcal{X}$ represents our prior knowledge about the system and does not rely on the dataset of trajectories. In the absence of prior knowledge, for example, the set $\mathcal{X}$ becomes the entire state space of the system. Given all the trajectory data $\{(x_0^{(i)}, x_T^{(i)})\}_{i=1}^m$ and the prior knowledge $\mathcal{X}$, we would like to check whether the system dynamics is consistent with the trajectory data, i.e., whether any system within the parameterized family (1) is able to generate the trajectory data without violating the prior knowledge $\mathcal{X}$.

Problem 1 (Data-driven model validataion)**.** Given a set of initial and terminal states $\{(x_0^{(i)}, x_T^{(i)})\}_{i=1}^m$, prior knowledge $\mathcal{X}$ as described by (2), and a set $\mathcal{P}$ of possible parameters. Verify whether there exists $p \in \mathcal{P}$ such that

$$x_T^{(i)} = \phi(x_0^{(i)}, p, T) \quad \text{and} \quad \phi(x_0^{(i)}, p, t) \in \mathcal{X}$$

for all $t \in [0, T]$ and $i \in \{1, 2, \ldots, m\}$, where $\phi(\cdot, p, T)$ is the flow of the parameterized dynamics (1) at time T.

In Problem 1, if such $p \in \mathcal{P}$ exists, we say that the model is *validated* by the trajectory data $\{(x_0^{(i)}, x_T^{(i)})\}_{i=1}^m$. Otherwise, we say that the model is *invalidated*. Later, we shall see that it is often computationally more tractable to solve the invalidation problem.

The formulation presented in Problem 1 is slightly different than the ones used in existing model validation literature [10], [11], [8], which often assumes that knowledge about the initial and terminal states is given as sets that contain the system states. Our formulation incorporates the available data into model validation problem explicitly without any additional processing of the trajectory data (such as computing the set that contains the states). As a consequence, the complexity of the model validation problem depends on the size of the dataset and motivates the use of algorithms that are scalable with the problem size.

III. Model validation using barrier certificates

In this section, we review the method of barrier certificates proposed by Prajna [10] for system model validation. A barrier certificate, in this context, is a function that witnesses that the available data cannot be explained by any system model within the parameterized family. For computational reasons, we focus on the case where the dynamics can be expressed in polynomials. In this case, construction of a barrier certificate requires solving a feasibility problem in which the constraints can be divided into two parts. The first part is a polynomial positivity constraint that can be relaxed into a convex constraint; the second part consists of a collection of affine constraints whose number is the same as the number of trajectories in the dataset.

A. Model validation using barrier certificates

There are certain cases where the model validation problem adopts simple solutions and does not require using the method of barrier certificates. One such case is when the set $\mathcal{P}$ of parameters is a singleton set, i.e., $\mathcal{P} = \{p_0\}$, for which the system validation problem can be solved by simulations. Specifically, one can run m simulations of the system for $t \in [0, T]$ using the dynamics (1) with $p = p_0$, where each simulation starts with the initial state $x_0^{(i)}$ given in the dataset. If any of the terminal states at $t = T$ does not coincide with the corresponding $x_T^{(i)}$ in the dataset,

then the model is invalidated. However, the simulation-based validation method does not apply for a general set $\mathcal{P}$; in this case, one needs to run simulations for all $p \in \mathcal{P}$, which becomes intractable since the number of simulations is infinite.

To circumvent this difficulty, Prajna [10] proposed to use *barrier certificates* as a way for system model validation without performing simulations or computing the flow of the system. The existence of a barrier certificate provides a *direct proof* that the system model is *invalidated* by the data.

Proposition 2 (Barrier certificate [10])**.** *Suppose that the dataset of initial and terminal states is given as* $\{(x_0^{(i)}, x_T^{(i)})\}_{i=1}^m$*, and the prior knowledge about the system trajectories as described by (2) is given by* $\mathcal{X}$*. If there exists a function* $B(\cdot, p, \cdot)$ *that is differentiable for all* $p \in \mathcal{P}$ *and satisfies*

$$B(x_T^{(i)}, p, T) - B(x_0^{(i)}, p, 0) > 0, \quad \forall p \in \mathcal{P},\ i = 1, 2, \ldots, m, \tag{3}$$

$$\frac{\partial B}{\partial x}(x, p, t) f(x, p, t) + \frac{\partial B}{\partial t}(x, p, t) \leq 0, \quad \forall x \in \mathcal{X},\ p \in \mathcal{P},\ t \in [0, T], \tag{4}$$

then the system model (1) is invalidated by $\{(x_0^{(i)}, x_T^{(i)})\}_{i=1}^m$ *and* $\mathcal{X}$*.*

Any function B that satisfies conditions (3) and (4) is called a *barrier certificate* (for model validation) for the system model (1), trajectory dataset $\{(x_0^{(i)}, x_T^{(i)})\}_{i=1}^m$, and prior knowledge $\mathcal{X}$ about the system trajectories. The existence of a barrier certificate is a sufficient condition for the model to be invalidated by the given trajectory data.

B. Construction of barrier certificates

It is generally difficult to apply Proposition 2 directly in order to find a barrier certificate, since it is intractable to search over all possible differentiable functions for a barrier certificate. However, in certain cases, there are numerical methods for constructing a valid barrier certificate. In the next, we will focus on system dynamics $f(x, p, t)$ that are polynomial in (x, p, t). In this case, instead of searching for a barrier certificate over the space of all functions, we can construct the barrier certificate B using polynomials with bounded degree [10]. Moreover, we will only search over barrier certificates that do not depend on p, so that we can write B as

$$B(x, t) = \sum_{k=1}^{K} c_k B_k(x, t), \tag{5}$$

where $\{B_k\}_{k=1}^K$ are monomials in (x, t) and $c = (c_1, c_2, \ldots, c_K)$ are the coefficients. Substituting (5) into (3) and (4), we obtain

$$\sum_{k=1}^{K} c_k \left[B_k(x_T^{(i)}, T) - B_k(x_0^{(i)}, 0) \right] > 0, \quad i = 1, 2, \ldots, m \tag{6}$$

$$\sum_{k=1}^{K} c_k \left[\frac{\partial B_k}{\partial x}(x, t) f(x, p, t) + \frac{\partial B_k}{\partial t}(x, t) \right] \leq 0, \quad \forall x \in \mathcal{X},\ p \in \mathcal{P},\ t \in [0, T]. \tag{7}$$

Under the representation (5), the search of a barrier certificate B becomes equivalent to finding a feasible c that satisfies constraints (6) and (7). When the monomial basis functions B_k are chosen, condition (6) consists of m linear constraints in c. On the other hand, condition (7) is a negative polynomial constraint, i.e., the coefficients c must maintain negativity of the polynomial over a given set. In general, finding a feasible solution to constraint (7) is difficult. However, one can relax (7) into linear matrix inequality constraints in c using sums-of-squares techniques. Relaxation of (7) into linear matrix inequality constraints is beyond the scope of this paper, and readers can refer to [10] and the work of Parrilo [9] for more details.

To summarize, for systems whose dynamics are characterized by polynomials, the search of barrier certificates amounts to finding feasible solutions for m affine constraints (6) and a single negative polynomial constraint (7) that can be relaxed into a linear matrix inequality constraint. When the number of trajectories (i.e., m) is small, the computational complexity of finding feasible solutions for (6) and (7) is dominated by (7). However, as m increases, the complexity of solving (7) remains the same, while that of solving (6) grows *linearly* with m using conventional algorithms such as the interior-point method. When m is large, algorithms with linear time complexity may become computationally expensive to use. In the next, we will present an algorithm whose time complexity grows only *sublinearly* with m and is more favorable for solving model validation problems in the regime of large datasets.

IV. A SUBLINEAR ALGORITHM FOR MODEL VALIDATION

In this section, we show how the multiplicative weights method can be applied to solve the data-driven model validation problem. The multiplicative weights method was originally proposed to obtain an *approximate* feasible solution to a set of affine constraints. Compared to conventional methods such as the interior-point method, the benefit of the multiplicative weights method is that its time complexity grows only sublinearly with the number of constraints, which is the same as the size of the dataset when applied to the problem of model validation. Moreover, with proper modifications, the multiplicative weights method will yield an exact feasible solution and hence a valid barrier certificate for the model validation problem.

A. The multiplicative weights method

The *multiplicative weights method* is a meta-algorithm that was originally proposed to solve the problem of expert selection with no regret, and it has a broad range of applications in machine learning, optimization, and game theory. In this paper, we consider the application of the multiplicative weights method proposed by Arora et al. [1] in finding a

feasible solution for affine constraints. Specifically, we would like to find $z \in \mathbb{R}^n$ such that

$$z \in \mathcal{Z} \quad \text{and} \quad Az \succeq b \tag{8}$$

for some given bounded convex set $\mathcal{Z} \subset \mathbb{R}^n$ and $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$. We consider the scenario where m is much larger than n, and it is relatively easy to check the feasibility $z \in \mathcal{Z}$, so that the time complexity is dominated by the affine constraints $Az \succeq b$.

For numerical analysis, it is often useful to consider solutions that are approximately feasible for the constraint (8). We say that z is a δ-approximate feasible solution to (8) if z satisfies

$$z \in \mathcal{Z} \quad \text{and} \quad Az \succeq b - \delta\mathbf{1}. \tag{9}$$

As $\delta \to 0$, one can obtain a δ-approximate solution that is arbitrarily close to an exact feasible solution. In addition, we assume that there is an oracle that is able to compute $z \in \mathbb{R}^n$ such that

$$z \in \mathcal{Z} \quad \text{and} \quad c^T z \geq d, \tag{10}$$

for any given $c \in \mathbb{R}^n$ and $d \in \mathbb{R}$ or report infeasible if no such z exists.

In Algorithm 1, we present the multiplicative weights method for obtaining a δ-approximate feasible solution for the affine constraints (8). The constant $\ell \in \mathbb{R}$ that appears in Algorithm 1 is called the *width* of the affine constraints $Az \succeq b$, and ℓ is defined such that

$$-\ell\mathbf{1} \preceq Az - b \preceq \ell\mathbf{1}, \quad \forall z \in \mathcal{Z}.$$

One way to compute the width ℓ is to use the fact that $\mathcal{Z}$ is bounded. Denote by a_i^T the i-th row of A. Then we have for all $i \in \{1, 2, \ldots, m\}$,

$$\ell \leq \max_{z\in\mathcal{Z}} |a_i^T z - b_i| \leq \max_{z\in\mathcal{Z}} |a_i^T z| + b_i \leq \|a_i\| \cdot \max_{z\in\mathcal{Z}} \|z\| + b_i.$$

Since the set $\mathcal{Z}$ is bounded, we know that $\max_{z\in\mathcal{Z}} \|z\|$ is also bounded, and hence ℓ is bounded.

The procedure in Algorithm 1 has an intuitive explanation. The weight vector w_t keeps track of how the constraints are violated. At the end of each iteration (step 3), the weights for the constraints that are violated are increased, whereas the weights for those that are satisfied are decreased. Hence, during the next iteration, it is more likely to find z_{t+1} in step 1 to satisfy the constraints that are previously violated.

Arora et al. [1] have shown that the maximum number of iterations required by Algorithm 1 for computing a δ-approximate solution is sublinear in the number of affine constraints m.

Proposition 3 (Arora et al. [1])**.** *Consider the affine feasibility problem as described by (8). Suppose the width of the affine constraints $Az \succeq b$ is ℓ. Then Algorithm 1 will either find a δ-approximate solution to the constraints (8) or report that (8) is infeasible within $T \leq \left\lceil \frac{8\ell^2 \log m}{\delta^2} \right\rceil$ steps.*

Note that the time complexity of Algorithm 1 is expressed in terms of the number of oracle calls (step 1). We ignore the time complexity of steps 2 and 3 since they are often relatively cheap to compute compared to the oracle. In order for the multiplicative weights method to be useful in practice, it is necessary that the computation required by the oracle is relatively cheap. This often applies in the case when m is much larger than the problem dimension n, so that the computational complexity of solving the feasibility of (8) is dominated by the affine constraints $Az \succeq b$.

Algorithm 1 Multiplicative weights method for approximate affine feasibility.

Input: $\mathcal{Z} \subset \mathbb{R}^n$, $A \in \mathbb{R}^{m\times n}$, $b \in \mathbb{R}^m$, ℓ, and $\delta > 0$.
Output: z satisfying constraint (9) or that constraint (9) is infeasible.
Choose $\epsilon = \delta/4\ell$ and $w_1 \in \mathbb{R}^m$ such that $w_{1,i} = 1$ for all $i \in \{1, 2, \ldots, m\}$.
For $t = 1, 2, \ldots, T$:

1) (Oracle) Find z_t such that $z_t \in \mathcal{Z}$ and $w_t^T A z_t \geq w_t^T b$. If no such z_t exists, terminate and report that the constraint (9) is infeasible.
2) Compute $m_t := (Az_t - b)/\ell$.
3) Update $w_{t+1} \in \mathbb{R}^m$ as follows:

$$w_{t+1,i} = \begin{cases} w_{t,i}(1-\epsilon)^{m_{t,i}} & \text{if } m_{t,i} \geq 0 \\ w_{t,i}(1+\epsilon)^{-m_{t,i}} & \text{if } m_{t,i} < 0. \end{cases}$$

Compute $z = \frac{1}{T}\sum_{t=1}^T z_t$.

B. Data-driven model validation in sublinear time

Recall that we can invalidate a model of dynamical systems from data using barrier certificates. In the case of polynomial dynamics, the search of barrier certificates becomes finding a feasible c such that constraints (6) and (7) are satisfied. We can compactly rewrite (7) as $c \in \mathcal{C}$ for some convex set $\mathcal{C}$ after relaxing of (7) into a linear matrix inequality using sums-of-squares techniques. Define the matrix $A \in \mathbb{R}^{m\times K}$ such that

$$a_{ik} = B_k(x_T^{(i)}, T) - B_k(x_0^{(i)}, 0). \tag{11}$$

Under the definitions of $\mathcal{C}$ and A, the problem of finding a barrier certificate becomes finding c that satisfies

$$c \in \mathcal{C} \quad \text{and} \quad Ac \succ 0. \tag{12}$$

There are two remaining issues before Algorithm (1) can be applied for solving the model validation problem. Firstly, the width of the affine constraints $Ac \succ 0$ is unbounded, since $\mathcal{C}$ is unbounded. Indeed, if c^* is a feasible solution to (7), then αc^* is also a feasible solution for any $\alpha > 0$. When the set $\mathcal{C}$ is unbounded, the width ℓ of the constraints $Ac \succ 0$ also becomes unbounded, which leads to unbounded number of iterations in Algorithm 1. In order to apply Algorithm 1, we can choose some constant $\bar{c} > 0$ and include an additional norm constraint $c \in \mathcal{B}_{\bar{c}} := \{c\colon \|c\| \leq \bar{c}\}$. Secondly, the affine constraints $Ac \succ 0$ are strict inequality constraints and are not amenable for numerical computation.

Algorithm 2 Multiplicative weights method for computing a barrier certificate.

Input: $\{(x_0^{(i)}, x_T^{(i)})\}_{i=1}^m$, $\mathcal{C}$, $\{B_k\}$, and $\bar{c}$.
Output: c satisfying constraint (14) or that constraint (14) is infeasible.
Choose

$$\ell = \bar{c} \cdot \max_{i \in \{1,2,\ldots,m\}} \left\| B_k(x_T^{(i)}, T) - B_k(x_0^{(i)}, 0) \right\|, \quad (15)$$

$\epsilon = 1/4\ell$, and $w_1 \in \mathbb{R}^m$ such that $w_{1,i} = 1$ for all $i \in \{1, 2, \ldots, m\}$.
For $t = 1, 2, \ldots, T$:

1) (Oracle) Find c_t such that $c_t \in \mathcal{C} \cap \mathcal{B}_{\bar{c}}$ and $w_t^T A c_t \geq w_t^T \mathbf{1}$, where A is defined in (11). If no such c_t exists, terminate and report that the constraint (14) is infeasible.
2) Compute $m_t := (Ac_t - \mathbf{1})/\ell$.
3) Update $w_{t+1} \in \mathbb{R}^m$ as follows:

$$w_{t+1,i} = \begin{cases} w_{t,i}(1-\epsilon)^{m_{t,i}} & \text{if } m_{t,i} \geq 0 \\ w_{t,i}(1+\epsilon)^{-m_{t,i}} & \text{if } m_{t,i} < 0. \end{cases}$$

Compute $c = \frac{1}{T}\sum_{t=1}^T c_t$.

For numerical purposes, we rewrite (6) as

$$\sum_{k=1}^K c_k \left[B_k(x_T^{(i)}, T) - B_k(x_0^{(i)}, 0) \right] \geq \gamma, \quad i = 1, 2, \ldots, m \quad (13)$$

for some $\gamma > 0$ to avoid handling strict inequality constraints. Under the new transformations, the problem becomes finding c such that

$$c \in \mathcal{C} \cap \mathcal{B}_{\bar{c}} \quad \text{and} \quad Ac \succeq \gamma \mathbf{1}. \quad (14)$$

It should be noted that the choices of $\bar{c}$ and γ are not independent. If γ is too large or $\bar{c}$ is too small, then the constraint (14) may become infeasible even if the original constraint (12) is feasible. For practical implementation, we can fix γ and keep increasing $\bar{c}$ until (14) becomes feasible. The limitation of this implementation, however, is that we have to terminate once $\bar{c}$ becomes large enough, after which the feasibility of the original constraint (12) becomes undecidable using Algorithm 2.

Even though the multiplicative weights method only gives an approximate feasible solution to constraints in (14), by properly choosing the parameters in the method, we can still obtain an exact barrier certificate, as shown in the following theorem. The modified algorithm is presented in Algorithm 2.

Theorem 4. *Consider the problem of searching for a barrier certificate as described by (6) and (7) for $\|c\| \leq \bar{c}$. Then Algorithm 2 will either find a barrier certificate or report that no barrier certificate exists within $T \leq \lceil 8\ell^2 \log m \rceil$ steps, where ℓ is given by (15).*

Proof: Recall that the problem of searching for a barrier certificate can be rewritten in a compact way as (14). Choose $\gamma = 1$. For any $\delta < 1$, consider a δ-approximate solution c that satisfies the constraints (14), i.e.,

$$c \in \mathcal{C} \cap \mathcal{B}_{\bar{c}} \quad \text{and} \quad Ac \succeq (1-\delta)\mathbf{1}.$$

It can be verified that such an approximate solution satisfies the original constraints (12) and hence yields a valid barrier certificate. Choose $\delta \to 1$ and substitute into Proposition 3 to complete the proof. ∎

Remark 5. Although the choice of $\gamma = 1$ may seem arbitrary, it does not affect the result on computational complexity. Note that the computational complexity is also affected by the width ℓ, which depends on $\bar{c}$. Denote by $\bar{c}^*$ the minimum $\bar{c}$ such that (14) is feasible for $\gamma = 1$. Then it can be verified that $\gamma\bar{c}^*$ is the minimum $\bar{c}$ such that (14) is feasible for any $\gamma > 0$, so that the quantity $\lceil 8\ell^2 \log m/\delta^2 \rceil$ remains the same as $\delta \to \gamma$.

Note that the quantity $\lceil 8\ell^2 \log m \rceil$ given by Theorem 4 is the number of iterations required to find a feasible solution in the worst case. In practice, we can terminate if the running average $\bar{c}_t := \frac{1}{t}\sum_{\tau=1}^t c_\tau$ is a feasible solution to $A\bar{c}_t \succ 0$. As will be seen in the next section, the actual number of iterations can be much less than the worst-case prediction given by Theorem 4.

V. Numerical experiments

We use an example adopted from the one used by Prajna [10]. The parameterized system dynamics are given by

$$\dot{x} = -px^3, \quad (16)$$

where $p \in \mathcal{P} = [0.5, 2] \subset \mathbb{R}$. The prior knowledge $\mathcal{X}$ about the system trajectory is given by $\mathcal{X} = \mathbb{R}$. The set of initial states $x_0^{(i)}$ and terminal states $x_T^{(i)}$ are generated by uniform sampling from the intervals $[0.85, 0.95]$ and $[0.55, 0.65]$, respectively, whereas the period of simulation $T = 4.0$. We note that the solution to (16) can be obtained in closed form so that it is not necessary to find a barrier certificate in order to validate the model. Nevertheless, we choose to use the dynamics (16) primarily for investigating how the multiplicative weights method scales with the size of the dataset.

We consider barrier certificates in the form

$$B(x, t) = B_1(x) + tB_2(x),$$

where both B_1 and B_2 are polynomials in x up to degree 4, so that the number of basis functions $K = 10$. All simulations are performed in MATLAB (R2014a) on a Linux workstation equipped with a quad-core 3.4 GHz Intel Core i7 processor and 8 GB of RAM. The oracle (step 1 of Algorithm 2) is computed using the sums-of-squares module [6] provided in YALMIP (Release 20150204) [5]. We have compared several optimization solvers and found that SeDuMi (version 1.34) [13] gives the best result. When running Algorithm 2, at the end of each iteration, we also compute $\bar{c}_t = \frac{1}{t}\sum_{\tau=1}^t c_\tau$ and terminate if $A\bar{c}_t \succ 0$.

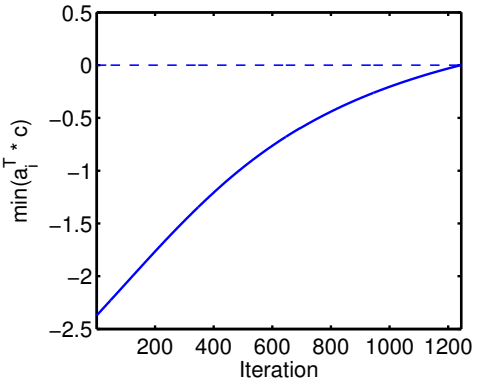

Fig. 1. The feasibility gap $\min_{i\in\{1,2,\ldots,m\}}\{a_i^T\bar{c}_t\}$ obtained as a function of the iteration t when running Algorithm 2 for $m = 100,000$. The algorithm terminates when $t = 1,244$ and $\bar{c}_t$ satisfies $A\bar{c}_t \succ 0$.

Fig. 1 shows the convergence of Algorithm 2 for $m = 100,000$. As the number of iterations t increases, the feasibility gap

$$\min_{i\in\{1,2,\ldots,m\}}\{a_i^T\bar{c}_t\}$$

approaches zero. At $t = 1,244$, we have the feasibility gap

$$\min_{i\in\{1,2,\ldots,m\}}\{a_i^T\bar{c}_t\} > 0,$$

i.e., $A\bar{c}_t \succ 0$. This implies that $\bar{c}_t$ is a feasible solution to the constraints (12) and that we have successfully found a barrier certificate to invalidate the model based on available data, i.e., there is no $p \in \mathcal{P}$ such that the data can be explained by the dynamics $\dot{x} = -px^3$. As a comparison, the number of iterations as predicted by Theorem 4 is $\lceil 8\ell^2 \log m \rceil \approx 2.4 \times 10^5$, which is much larger than the actual number of iterations for reaching convergence.

Table I compares the actual running time of the multiplicative weights method with the direct method that solves the problem using SeDuMi for different number of data points m. The entries with an asterisk $(*)$ correspond to cases in which the program terminates early due to numerical issues. For the multiplicative weights method, the computational time for the oracle (step 1 in Algorithm 2) has no dependence on m. The time required for each iteration in Algorithm 2 only slightly increases with m because of steps 2 and 3, so that the running time of the multiplicative weights method mainly depends on the number of iterations. Although the worst-case number of iterations should grow logarithmically with m according to Theorem 4, the actual number of iterations and hence the running time was found to be insensitive to changes in m. On the other hand, the running time of the direct method is expected to grow at least linearly with m. As can be seen from Table I, the running time of the multiplicative weights method outperforms that of the direct method when the size m of the dataset is large.

VI. Conclusions

In this paper, we explore the application of the multiplicative weights method in barrier-certificate-based data-driven model validation of dynamical systems in order to handle large datasets of trajectories. In order to invalidate a given model from available data, we need to find a barrier certificate by solving a convex feasibility problem with a large number of affine constraints (the same as the number of trajectories). Although the multiplicative weights method was originally proposed to compute an approximate feasible solution for affine constraints, we show that a modified multiplicative weights method is able to yield an exact solution and hence a valid barrier certificate. In theory, the method is able to achieve sublinear time complexity (measured by the number of iterations) in terms of the size of the given dataset. Through numerical simulations, we have found that the running time of the multiplicative weights method is very insensitive to the size of the dataset. Compared to conventional direct solution methods such as the interior-point method, the multiplicative weights method is particularly favorable in the regime of large datasets.

m	Multiplicative Weights (sec)	Direct Method (SeDuMi) (sec)
1,000	107.7	0.23
10,000	125.6	0.88
100,000	124.4	13.70*
1,000,000	124.8	569.21*

TABLE I
Running time comparison

References

[1] S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.

[2] D. Bertsimas, V. Gupta, and N. Kallus. Data-driven robust optimization. *arXiv preprint arXiv:1401.0212*, 2013.

[3] G. Calafiore and M. C. Campi. Uncertain convex programs: randomized solutions and confidence levels. *Mathematical Programming*, 102(1):25–46, 2005.

[4] D. P. De Farias and B. Van Roy. On constraint sampling in the linear programming approach to approximate dynamic programming. *Mathematics of Operations Research*, 29(3):462–478, 2004.

[5] J. Löfberg. YALMIP: A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[6] J. Löfberg. Pre- and post-processing sum-of-squares programs in practice. *IEEE Transactions on Automatic Control*, 54(5):1007–1011, 2009.

[7] Y. Nesterov, A. Nemirovskii, and Y. Ye. *Interior-point polynomial algorithms in convex programming*, volume 13. SIAM, 1994.

[8] N. Ozay, M. Sznaier, and C. Lagoa. Model (in)validation of switched ARX systems with unknown switches and its application to activity monitoring. In *IEEE Conference on Decision and Control (CDC)*, pages 7624–7630. IEEE, 2010.

[9] P. A. Parrilo. *Structured semidefinite programs and semialgebraic geometry methods in robustness and optimization*. PhD thesis, California Institute of Technology, 2000.

[10] S. Prajna. Barrier certificates for nonlinear model validation. *Automatica*, 42(1):117–126, 2006.

[11] S. Prajna, A. Jadbabaie, and G. J. Pappas. A framework for worst-case and stochastic safety verification using barrier certificates. *IEEE Transactions on Automatic Control*, 52(8):1415–1428, 2007.

[12] A. B. Sharma, F. Ivančić, A. Niculescu-Mizil, H. Chen, and G. Jiang. Modeling and analytics for cyber-physical systems in the age of big data. *ACM SIGMETRICS Performance Evaluation Review*, 41(4):74–77, 2014.

[13] J. F. Sturm. Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11(1-4):625–653, 1999.

[14] D. P. Woodruff. Sketching as a tool for numerical linear algebra. *Theoretical Computer Science*, 10(1-2):1–157, 2014.