Annual Reviews in Control 45 (2018) 267–280

Contents lists available at ScienceDirect

Annual Reviews in Control

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

Perturbation analysis: A framework for data-driven control and optimization of discrete event and hybrid systems☆,☆☆

Y. Wardi[a,*], C.G. Cassandras[b], X.R. Cao[c]

[a] School of Electrical and Computer Engineering, Georgia Institute of Technology, 777 Atlantic Drive, Atlanta, GA, 30332, USA
[b] Division of Systems Engineering, and Department of Electrical and Computer Engineering, Boston University, 8 St. Mary's Street, Boston, MA, 02215, USA
[c] Antai College of Economics and Management and School of Electronic, Information and Electrical Engineering, Shanghai Jiao Tong University, 1954 Huashan Road, Shanghai, 200030, People's Republic of China

ARTICLE INFO

Article history:
Received 10 January 2018
Revised 18 March 2018
Accepted 3 April 2018
Available online 13 April 2018

Keywords:
Perturbation analysis
Discrete-event dynamic systems
Hybrid systems
Performance optimization and control

ABSTRACT

The history of Perturbation Analysis (PA) is intimately related to that of Discrete Event Dynamic Systems (DEDS), starting with a solution of a long-standing problem in the late 1970s and continuing today with the control and optimization of Hybrid Systems and the emergence of event-driven control methods. We review the origins of the PA theory and how it became part of a broader framework for modelling, control and optimization of DEDS. We then discuss the theoretical underpinnings of Infinitesimal Perturbation Analysis (IPA) as a data-driven stochastic gradient estimation method and how it has been applied over the past few decades. We explain how IPA offers a basis for general-purpose stochastic optimization of Markovian systems through the notion of the performance potential and how it has evolved beyond DEDS and now provides a framework for control and optimization of Hybrid Systems and, more generally, event-driven methodologies.

1. The origin of perturbation analysis

In pioneering the field of Discrete Event Systems (DES) in the early 1980s, Y.C. Ho and his research group at Harvard University discovered that event-driven dynamics give rise to state trajectories (sample paths) from which one can very efficiently and nonintrusively extract sensitivities of state variables (therefore, various performance metrics as well) with respect to at least certain types of design or control parameters. This eventually led to the development of a theory for *Perturbation Analysis* (PA) in DES (Cassandras & Lafortune, 2008; Glasserman, 1991; Ho & Cao, 1991), the most successful branch of which is *Infinitesimal Perturbation Analysis* (IPA) due to its simplicity and ease of implementation. In fact, by the early 2000s, IPA was shown to apply to all virtually arbitrary Hybrid Systems (HS) and continues to be today one of the most attractive tools for data-driven control and optimization, especially in stochastic environments where modelling random aspects of a process is prohibitively hard.

The origin of the key concepts that form the cornerstones of the PA theory are found in a long-standing problem in operations research and industrial engineering known as the *buffer allocation problem*. In its industrial engineering version, it was presented to Ho's research group by the FIAT automobile company in the late 1970s as follows. A typical serial transfer line consists of N workstations in tandem, each with different characteristics in terms of its production rate, failure rate and repair time when failing. In order to accommodate this inhomogeneous behavior, a buffer is placed before the kth workstation, $k = 1, \dots, N$, with B_k discrete slots where production parts can be queued. Since the space within which this transfer line operates is limited, there is an upper bound B to the total number of buffer slots that can be allocated over the N workstations so that $\sum_{k=1}^{N} B_k = B$. The problem is to allocate these B buffer slots, i.e., determine a vector $[B_1 \dots B_N]$, so as to maximize the throughput of the transfer line while also maintaining a low overall average delay of the parts moving from an entry point before the first workstation to an exit point following the Nth workstation. Tackling this problem in a "brute force" manner requires considering all possible buffer allocations, a num-

☆ Wardi's work was supported in part by NSF under Grant Number CNS-1239225. Cassandras's work has been supported in part by NSF under grants CNS-1239021, ECCS-1509084, and IIP-1430145, by AFOSR under grant FA9550-15-1-0471, and by a grant from the MathWorks.

☆☆ This paper extends an earlier version which appeared in the Proceedings of the 20th IFAC World Congress, (Wardi, Cassandras, and Cao (2017)).

* Corresponding author.
E-mail addresses: ywardi@ece.gatech.edu (Y. Wardi), cgc@bu.edu (C.G. Cassandras), eecao@ust.hk (X.R. Cao).

https://doi.org/10.1016/j.arcontrol.2018.04.003
1367-5788/

ber given by $\binom{B+N-1}{B}$. For a reasonably small problem such as $B=24$ and $N=6$, this gives 118,755 possible solutions. A direct trial-and-error approach where one is allowed to test each allocation for about a week would require about 2300 years. If one were to reduce the initial solution space to only 1000 "good guesses" and use early 1980s simulation technology requiring about 3 min per trial to estimate the resulting performance, the overall task would take about 250 days of CPU time.

The approach taken by Ho's group and first reported in Ho, Eyler, and Chien (1979) was to study the serial transfer line as a dynamic system whose state includes the integer-valued buffer contents along with real-valued "clocks" associated with each workstation as it processes a part. The question then posed was: "what would happen if in a given allocation a specific value B_k were changed to B_k+1?" The "brute force" way to answer this question is to first simulate the system under the nominal allocation with the value B_k and estimate the system's performance over a (sufficiently long) time period T which may be denoted by $L_T(B_k)$. Then, repeat the simulation under B_k+1 to obtain $L_T(B_k+1)$. The difference $\Delta L_T(B_k)=L_T(B_k+1)-L_T(B_k)$ provides an estimate of the system's performance sensitivity with respect to B_k. What the research team realized, however, is that this is unnecessary: indeed, the initial simulation alone yielding $L_T(B_k)$ and a simple thought experiment can deliver the value of $\Delta L_T(B_k)$. Moreover, the same thought experiment can deliver the entire vector $[\Delta L_T(B_1),\dots,\Delta L_T(B_N)]$ with minimal extra effort.

The key observation that led to a formal procedure describing this thought experiment is the following. When B_k is replaced by B_k+1, no change in the state of the system can take place unless one of two "events" is observed at time t: (i) The kth buffer content, say $x_k(t)$, reaches its upper limit, i.e., $x_k(t)=B_k$ and a part is ready to leave the $(k-1)$th workstation. In this case, this upstream workstation is "blocked" since there is no place for the departing part to go. However, in a perturbed system with B_k replaced by B_k+1 that would not happen and one can simply predict a buffer content perturbation $\Delta x_k(t)=1$. Moreover, one can record when this blocking occurs at time $t\equiv t_{k,B}$ and the next time that a part departs from the kth workstation, $t_{k,D}$. Then, $t_{k,D}-t_{k,B}$ is the amount of time that would be gained (i.e., no blocking would have occurred) in a perturbed system realization. The important observation here is that $t_{k,D}$, $t_{k,B}$ are directly observed along the nominal system realization. (ii) The $(k+1)$th buffer content reaches its lower limit, i.e., $x_{k+1}(t)=0$ and a part is ready to leave the kth workstation. In this case, if $\Delta x_k(t)=1$, i.e., the kth workstation has already gained a part from an earlier blocking event, then this gain can now propagate downstream and we can set $\Delta x_{k+1}(t)=\Delta x_k(t)=1$.

This simple observation leads to the conclusion that estimating the effect of replacing B_k by B_k+1 boils down to observing just a few events along the nominal system realization: *blocking events* (when $x_{ki}(t)=B_k$ and a part departure from $k-1$ takes place) and *idling events* (when $x_k(t)=0$ at any $k=1,\dots,N$). This can be formalized into an "estimator" for buffer perturbations $\Delta x_k(t)$ and event timing perturbations for all part departures at workstations. More generally, this estimator transforms a given hypothetical perturbation $\Delta B_k(t)=1$ (or -1) into state perturbations, which can ultimately be used to estimate a performance perturbation $\Delta L_T(B_k)$. Most importantly, this is accomplished without ever having to implement the perturbation $\Delta B_k(t)$, since the estimator depends only on directly observable data from the nominal system realization; in particular, it suffices to observe selected events and associated event times and to perform extremely simple calculations.

This initial procedure pertaining to a very specific type of dynamic system and problem was given the name *Perturbation Analysis* (PA). It soon became clear that it could be extended to any system with a structure similar to that of the serial transfer line and to a perturbation in any system parameter. Thus, one could consider, for instance, speeding up the operation of a workstation and studying the effect of a perturbation Δr_k in the operation rate r_k of the kth workstation. The general procedure is one where some parameter perturbation $\Delta\theta$ *generates* a state perturbation $\Delta x_k(t)$ when a specific event occurs at time t. Subsequently, the system dynamics dictate how $\Delta x_k(t)$ *propagates* through the system by affecting $\Delta x_k(t)$ or $\Delta x_j(t)$ for $j\neq k$. Depending on a performance metric of interest, this ultimately yields $\Delta L_T(\Delta\theta)$, the change in performance due to $\Delta\theta$. As for the system structure amenable to this kind of efficient PA, it became obvious that it fits the general class of queueing networks.

An obvious next question was: "Does PA hold for any value of $\Delta\theta$ or do we have to restrict it to "small" $\Delta\theta$ when θ is real-valued?" There was ample empirical evidence collected over the early 1980s that $\Delta\theta$ had to be small but not necessarily "very small". In other words, the values of $\Delta L_T^{PA}(\Delta\theta)$ obtained through PA were identical to those obtained through the "brute force" finite difference $L_T(\theta+\Delta\theta)-L_T(\theta)$ for "sufficiently small" $\Delta\theta$. This led to the term *Infinitesimal Perturbation Analysis* (IPA) to capture the fact that the methodology was applicable to perturbations which were "infinitesimally" small, although a formal quantification characterizing limits for $\Delta\theta$ was lacking. Moreover, when $\Delta\theta$ became larger, it was still possible to satisfy $\Delta L_T^{PA}(\Delta\theta)=L_T(\theta+\Delta\theta)-L_T(\theta)$ at the expense of observing more "interesting events" and performing a few more calculations. For instance, in the case of the integer-valued buffer size parameter B_k, the minimal feasible perturbation is obviously either $+1$ or -1. To differentiate these cases, the term *Finite Perturbation Analysis* (FPA) was introduced. FPA reverts to IPA when parameters are real-valued and may be allowed to take "sufficiently small" values $\Delta\theta$.

To illustrate the distinction between IPA and FPA, we consider the case of a simple First-In-First-Out (FIFO) queueing system with a single server preceded by a queue. Let $\{A_i\}$ be the sequence of (generally random) arrival times, $i=1,2,\dots,$ and $\{D_i\}$ be the corresponding sequence of departure times from the system. If S_i denotes the service time of the ith entity (customer) processed, then the Lindley equation

$$D_i=\max(A_i,D_{i-1})+S_i \tag{1}$$

describes the departure time dynamics with $i=1,2,\dots$ Suppose that all (or just some selected subset) of the service times are perturbed by ΔS_i, $i=1,2,\dots$. Let $I_i=A_i-D_{i-1}$ and observe that when $I_i>0$ it captures an idle period (since the server must wait until $A_i>D_{i-1}$ to become busy again) and when $I_i<0$ it captures the waiting time $D_{i-1}-A_i$ of the ith arriving entity in the system. It is easy to obtain from (1) the following departure time perturbation equation:

$$\Delta D_i=\Delta S_i+\begin{cases}\Delta D_{i-1} & \text{if } I_k\le 0,\ \Delta D_{i-1}\ge I_i\\ 0 & \text{if } I_i>0,\ \Delta D_{i-1}\le I_i\\ I_i & \text{if } I_i\le 0,\ \Delta D_{i-1}\le I_i\\ \Delta D_{i-1}-I_i & \text{if } I_i>0,\ \Delta D_{i-1}\ge I_i\end{cases} \tag{2}$$

where ΔD_i can be obtained from the generated perturbations ΔS_i and directly observed data in the form of I_i. This is the FPA procedure for evaluating ΔD_i, $i=1,2,\dots$ Observe, however, that if we select $\Delta S_i>0$ to be sufficiently small so that $\Delta D_{i-1}>0$ can never exceed the finite value of $I_i>0$, then this reduces to

$$\Delta D_i=\Delta S_i+\begin{cases}\Delta D_{i-1} & \text{if } I_i\le 0\\ 0 & \text{otherwise}\end{cases} \tag{3}$$

which is the much simpler IPA version, requiring only the detection of an idling interval when $I_i > 0$, at which time the departure time perturbation is reset to $\Delta D_i = \Delta S_i$. Naturally, the question is: "How small do perturbations need to be before the IPA equation can be used?" In a stochastic system, the answer to this question is generally dependent on the specific realization based on which ΔD_i is evaluated. Thus, it is logical to extend the question of estimating $\Delta D_i(\Delta\theta)$, $i = 1, 2, \ldots,$ the departure time perturbations, to estimating the derivative $\frac{dD_i}{d\theta}$ by allowing $\Delta\theta \to 0$. This became one of the two important questions facing PA researchers in the early 1980s:

Q1. When can IPA be used instead of FPA and is it possible to transition the remarkably efficient PA methodology from one limited to sensitivity analysis for finite perturbations to one for gradient estimation of a large class of dynamic systems?

Q2. What is the class of dynamic systems for which the PA methodology applies?

To address **Q2**, it soon became clear that using traditional models for system dynamics of the general form $\dot{x} = f(x, u, t)$ was exteremely inefficient and ultimately pointless for systems such as queueing networks. In such systems, at least some state variables (i.e., queueing contents) are discrete and constant most of the time, changing only when specific events occur (i.e., an entity enters or leaves the queue). This led to the realization that traditional dynamic systems described through $\dot{x} = f(x, u, t)$ are *time-driven*, whereas this different class of systems is *event-driven*. The term *Discrete Event Dynamic System* (DEDS) was coined in 1980 and first appeared in the literature in Ho and Cassandras (1980, 1983), while a first general IPA and FPA framework for queueing networks appeared in Ho, Cao, and Cassandras (1983). However, the event-driven nature of this class of systems was not formalized until an "event domain formalism" was first proposed in Cassandras and Ho (1985). Over the next several years, it became clear that the class of DEDS is much broader than queueing networks and that PA techniques could be extended to all such systems (Glasserman, 1991; Ho & Cao, 1991). In parallel, a novel control theory for such systems, with the broader term *Discrete Event System* (DES) used, was being developed by Ramadge and Wonham culminating with what has become known as the *supervisory control theory* for DES (Ramadge & Wonham, 1980). It took about a decade before the supervisory control theory and PA were merged into complementary approaches for studying DES and are now viewed as a staple of any study of dynamic systems (Cassandras, 1993; Cassandras & Lafortune, 2008).

Returning to the first question **Q1** above, IPA in the form of (3) was successfully used in the early 1980s for many applications that involved stochastic systems with event-driven behaviors, including routing, scheduling and general resource allocation problems in complex manufacturing systems and computer and communication networks (e.g., Cassandras, Abidi, & Towsley, 1990). The generalization of (3) is to apply it to any performance metric $J(\theta) = E[L(\theta)]$ where $L(\theta)$ is a sample function dependent on θ. IPA is an efficient way to obtain the gradient $\nabla L(\theta)$ from observable data on a nominal system realization. However, what is ultimately of interest is $\nabla J(\theta) = \nabla E[L(\theta)]$, and its estimation through $\nabla L(\theta)$ can be used in a large class of gradient-based optimization problems. As IPA was applied to harder and harder problems (i.e., systems with event-driven dynamics) much more complex than Lindley equations such as (1), it became clear that IPA estimates $\nabla L(\theta)$ were not accurate compared to $\nabla J(\theta)$ when this could be evaluated through analytical methods in some simple cases or accurately approximated through exhaustive time-consuming simulation methods. Indeed, one could have situations where the signs of $\nabla J(\theta)$ and $\nabla L(\theta)$ were different, resulting in heavily *biased* IPA gradient estimates. It took several years and occasionally controversial debates to realize that the key issue was one of testing the validity of unbiasedness for IPA gradient estimation, i.e., formal conditions under which

$$\nabla E[L(\theta)] = E[\nabla L(\theta)]$$

holds, or in simpler scalar form:

$$\frac{d}{d\theta} E[L(\theta)] = E\left[\frac{dL(\theta)}{d\theta}\right]$$

The way this key issue was addressed is discussed in the next section.

2. Infinitesimal perturbation analysis

By the mid 1980s, it was realized that IPA provided a general framework for computing gradients of sample performance functions defined on the state space of an extensive class of DEDS beyond queueing networks. Furthermore, it was shown to admit especially simple computations by data gathered directly from the sample path of the system. Consequently IPA became the focal point of research in PA, with an eye on potential applications in performance optimization by stochastic gradient-descent algorithms.[1]

IPA is predicated on a stochastic dynamical system underscored by a common probability space $(\Omega, \mathcal{F}, P)$.[2] A typical sample in Ω, denoted by ξ, together with a particular value of the parameter θ, determine a realization of the state trajectory of the system at θ. It is called a sample path at θ, and the sample path is said to be associated with the pair (θ, ξ). The basic thinking about IPA, based on its formulation in Cao (1985), was shaped by the view that it essentially compares two sample paths, corresponding to two respective, but close parameters and a common $\xi \in \Omega$.

Let $L_T(\theta, \xi)$ be a performance function of the system in a given finite period $[0, T]$. IPA consists of the sample gradient (derivative) $\nabla L_T(\theta, \xi) := \frac{\partial}{\partial\theta}(L_T(\theta, \xi))$, called the *sample derivative*, or *sample gradient* Cao (1985). Due to the DEDS structure of the system, it was convenient to think of this derivative as the limit of finite differences with the common $\xi \in \Omega$, namely

$$\frac{\partial L_T(\theta, \xi)}{\partial\theta} = \lim_{\Delta\theta \to 0} \frac{L_T(\theta + \Delta\theta, \xi) - L_T(\theta, \xi)}{\Delta\theta}.$$

However, one of the expressed objectives of IPA is to estimate the derivative of the mean performance $J_T(\theta) := E[L_T(\theta, \xi)]$.[3] This naturally raises the question "is the sample derivative given by IPA an *unbiased* estimate of the derivative of the mean performance?" i.e.,

$$E\left[\frac{\partial}{\partial\theta} L_T(\theta)\right] = \frac{\partial}{\partial\theta} E[L_T(\theta)]? \tag{4}$$

The unbiasedness (4) explains why IPA gives accurate derivative estimates for some systems but not others. Roughly speaking, when the sample function $L_T(\theta)$ has a jump (discontinuity) at θ, the interchangeability of expectation and differentiation inherent in (4) would not hold. Intuitive conditions for this interchangeability were given in Cao (1985); essentially, if a parameter change at θ may cause a change in the order of events in a DEDS, the sample performance function may have a jump at θ, and then the

[1] At the same time there was also an interest in another sensitivity analysis technique, based on distributional perturbations and likelihood ratios, called the *likelihood ratios method*, or *score function method*. Surveys thereof can be found in Rubinstein and Shapiro (1993) and Reiman and Weiss (1989), and a comparison with IPA, in Cao (1987).

[2] The term "common", in this context, means independent of the parameter θ.

[3] To simplify the presentation we will drop the explicit notational dependence of the sample performance functions on ξ.

interchangeability in (4) may hold only if the probability of such jumps in $[\theta, \theta+\Delta\theta]$ is of the order $o(\Delta\theta)$. This intuition evolved from the consideration of many engineering problems defined on DEDS. An alternative view of the issue of unbiasedness is through the theory of multivariable calculus. After all, (4) amounts to an interchangeability of differentiation with respect to θ and integration (expectation) with respect to ξ, and this is roughly equivalent to the continuity of the function $L(\theta, \xi)$ in θ with probability one (in ξ). The former view is based on engineering intuition, whereas the latter one, founded upon well-known results in mathematics, often provides a practical way of discerning whether IPA is unbiased.

These realizations stimulated subsequent research in two directions:

1. Identifying classes of systems and problems where IPA is unbiased. Early works include (Suri & Zazanis, 1988), which proves the unbiasedness for the GI/G/1 queue, and Cao (1988), which proves it for closed Jackson networks. Glasserman (1991) presents IPA in the framework of generalized semi-Markov processes, and extends the results in Cao (1985, 1988) to a *commuting condition* (a simple to check structural condition) for the unbiasedness of the derivative estimates. For many other applications in this direction, please see Cassandras and Lafortune (2008). The question of unbiasedness of FPA has been addressed as well; see Heidergott (2000) and references therein.

2. The development of alternative perturbation-analytic techniques that provide unbiased derivative estimates, or reduce the bias, for problems for which IPA estimates are biased. Several such techniques have appeared in the literature; among them is *Smoothed Perturbation Analysis (SPA)* (Fu & Hu, 1997; Gong & Ho, 1987). The main idea of this technique is to use the derivative of a conditional mean of the sample function as the estimate of the derivative of the mean performance. While a sample function may have jumps and therefore its derivative is a biased estimate, a conditional mean of the sample function may be smooth enough to provide unbiased estimates. More precisely, suppose that there is a random variable (or vector) denoted as Z such that

$$E\left[\frac{\partial}{\partial\theta}E[L_T(\theta)|Z]\right] = \frac{\partial}{\partial\theta}E\big[E[L_T(\theta)|Z]\big] = \frac{\partial}{\partial\theta}E[L_T(\theta)]. \tag{5}$$

Then, we can use $\frac{\partial}{\partial\theta}E[L_T(\theta)|Z]$, i.e., the derivative of the conditional-mean sample function, as an unbiased estimate of the performance derivative $\frac{\partial}{\partial\theta}E[L_T(\theta)E]$.. A potential difficulty with this approach is that the conditional IPA estimator may require a significantly higher computational effort than the basic IPA to the point that it is rendered impractical. In other words, precision and accuracy can be obtained at the expense of higher computational complexity.

Around the same time, various other techniques were also developed based largely on so-called "cut-and-paste" operations on the sample path in order to smooth out discontinuities resulting from parameter perturbations. Surveys thereof can be found in Ho and Cao (1991) and Cassandras and Lafortune (2008).

The main ideas described in the previous paragraphs will be illustrated by simple examples in Section 2.1.

The preceding discussion pertains to finite-horizon sample performance functions. Another important class of functions concern long-run (infinite-horizon) averages. Denoted by $J(\theta)$, they have the form

$$J(\theta) = \lim_{T\to\infty}\frac{1}{T}L_T(\theta),$$

where the system is assumed to be ergodic for the above limit to exist and be independent of $\xi\in\Omega$ w.p.1. The time T can be either continuous or discrete. IPA gives the sample derivative $\frac{1}{T}\frac{\partial}{\partial\theta}L_T(\theta)$. The issue here is the strong consistency of the IPA derivative, i.e., whether the following limit is in force (see Cao, 1985):

$$\lim_{T\to\infty}\frac{1}{T}\frac{\partial}{\partial\theta}L_T(\theta) = \frac{\partial}{\partial\theta}J(\theta); \tag{6}$$

in other words, "are the operators of limit "$\lim_{T\to\infty}$" and derivative "$\frac{\partial}{\partial\theta}$" interchangeable?"

The study of this issue led to an important concept, the *perturbation realization*, later extended to the *performance potential*, which has been applied to several research areas like Markov decision processes and stochastic control. The main idea is as follows. In queueing networks, the effect of every single perturbation on the performance is finite and can be precisely measured; the total effect of a parameter change on the performance can be decomposed into the sum of the effects of every single perturbation generated (realized) by this parameter change. The performance derivative with respect to this parameter can then be calculated. To illustrate this concept, consider a closed Jackson network consisting of M servers with service rates μ_i, $i = 1, 2, \cdots, M$, and let

$$L_T(\theta) := \int_0^T f[\boldsymbol{n}(t)]dt,$$

where $\boldsymbol{n}(t) = (n_1, n_2, \cdots, n_M)$ is the system state at time t with n_i denoting the number of customers at server i, and $f[\boldsymbol{n}(t)]$ is a performance function of the state. Now, suppose at $t = 0$ with initial state $\boldsymbol{n}$, server i is subjected to a perturbation Δ, meaning, e.g., its completion time is delayed by the amount of Δ. Define the *perturbation realization factor* (Cao, 1994; Ho & Cao, 1993),

$$c(\boldsymbol{n}, i) = \lim_{T\to\infty}\lim_{\Delta\to 0}E\left\{\frac{1}{\Delta}\left[\int_0^T f(\boldsymbol{n}'(t))dt - \int_0^T f(\boldsymbol{n}(t))dt\right]\right\}. \tag{7}$$

The perturbed realization of the sample path $\boldsymbol{n}'(t)$ can be simply obtained by the propagation rule on a single sample path. With the help of the realization factors, we can prove that the strong consistency (6) indeed holds, and we can further derive (take $\theta = \mu_i$ as the perturbed parameter)

$$\lim_{T\to\infty}\left[\mu_i\frac{1}{T}\frac{\partial L_T(\mu_i)}{\partial\mu_i}\right] = \mu_i\frac{dJ(\mu_i)}{d\mu_i} = \sum_{\text{all } \boldsymbol{n}}\pi(\boldsymbol{n})c(\boldsymbol{n}, i), \qquad \text{w.p.1}, \tag{8}$$

where $\pi(\boldsymbol{n})$ is the steady-state probability of state $\boldsymbol{n}$. $c(\boldsymbol{n}, i)$ can be computed by a set of linear equations.

In this approach, the interchangeability of "$\lim_{T\to\infty}$" and "$\frac{\partial}{\partial\theta}$" is buried in $\lim_{T\to\infty}$ and $\lim_{\Delta\to 0}$ in (7) because in a strongly connected network, a perturbation can only affect a system in a finite period, and the difference of the two terms in (7) will be almost zero when T is large enough. The interchangeability can be proved along these lines. Many other examples with perturbation realizations can be found in Cao (1994).

The notions of perturbation realization and performance potential continue to underscore subsequent applications to large-scale systems. For non-Markovian queueing networks and other DEDS, the unbiasedness of IPA and the computational complexity inherent in alternative PA methods designed to circumvent it, led to a partial shift of IPA research from DEDS to stochastic hybrid systems. These developments, which have been taking place over the past fifteen years, are the subject of the next section.

2.1. IPA and SPA examples

To further explain the basic ideas of IPA and its extension to SPA, we provide a few simple examples which illustrate the salient features underscoring the concept of unbiasedness. A practical way of ascertaining unbiasedness is based on the determination of whether the sample performance function $L_T(\theta, \xi)$ is continuous w.p.1. The connection between unbiasedness and continuity is founded on the following result. Suppose that θ is constrained to a closed, bounded interval $\Theta \subset R$. Assuming that (i) for every $\theta \in \Theta$, the derivative $\frac{\partial}{\partial\theta}L_T(\theta)$ exists w.p.1; and (ii) w.p.1, the function $L_T(\cdot)$ is Lipschitz continuous throughout Θ, and its Lipschitz constant, $K = K(\xi)$, has a finite first moment, namely $E[K] < \infty$; then the IPA derivative $\frac{\partial}{\partial\theta}L_T(\theta)$ is unbiased. This follows directly from the Lebesgue dominated convergence theorem (Rudin, 1974), and the fact that the inequality

$$\left|\left|\frac{L_T(\theta+\Delta\theta)-L(\theta)}{\Delta\theta}\right|\right| \le K \tag{9}$$

is satisfied w.p.1 for every $\theta \in \Theta$ and $\Delta\theta \neq 0$ such that $\theta + \Delta\theta \in \Theta$.

This simple result explains the importance of the continuity condition of the sample performance functions to the unbiasedness of their sample derivatives.

Remark 1. We call attention to the fact that assumption (i) pertains to a given $\theta \in \Theta$ while assumption (ii) concerns the function $L_T(\cdot)$ as defined throughout Θ. Thus, assumption (i) does not imply that the derivative $\frac{\partial}{\partial\theta}L_T(\theta)$ exists for every $\theta \in \Theta$ w.p.1. In fact, this is generally not true, but the function $L_T(\cdot)$ typically is only piecewise differentiable due to the discrete nature of the system. Practically, the key condition for unbiasedness is continuity of the sample performance function $L_T(\theta)$ w.p.1. The lack of differentiablity at a given θ is not necessarily problematic if one-sided derivatives exist, in which case unbiasedness can be established for the one-sided derivatives. All of this will be illustrated by the following examples.

Example 1. Consider a FIFO GI/G/1 queue where $\theta \in R$ is a parameter of the distribution of the service times while the arrival process is independent of θ. Denote the arrival time of job (customer) $i = 1, 2, \ldots,$ by A_i, and denote the service time of job i by $S_i(\theta)$. The processes $\{A_i\}$ and $\{S_i(\theta)\}$ can be viewed as realizations of $\xi \in \Omega$; typically A_i depends on the distribution of interarrival times, whereas $S_i(\theta)$ depends on the service-time distribution which depends on θ. In the present discussion we stipulate a sample path of the queue at a fixed parameter θ where, at time A_i, a sample-realization of the service time $S_i(\theta)$ can be measured or computed, it is differentiable with respect to θ, and its derivative $\frac{\partial}{\partial\theta}S_i(\theta)$ can be computed. Regarding this sample derivative, common examples include the case of deterministic service times where $S_i(\theta) = \theta$, and the case where $S_i(\theta)$ is exponentially distributed with mean θ. In the deterministic case, $\frac{\partial}{\partial\theta}S_i(\theta) = 1$. In the exponentially-distributed case, $S_i(\theta) = -\theta ln(1-\omega)$ for a unit-variate ω, hence $\frac{\partial}{\partial\theta}S_i(\theta) = -ln(1-\omega) = \frac{S_i(\theta)}{\theta}$.

Fix $N > 0$, and consider the sample performance function defined as the mean delay (sojourn time) of the first N jobs that arrive at the queue, denoted by $L_N(\theta)$. Further denoting the delay (sojourn time) of job i by $d_i(\theta)$, the sample performance function is defined by

$$L_N(\theta) = \frac{1}{N}\sum_{i=1}^{N} d_i(\theta). \tag{10}$$

It is natural to define the state variable as the departure time of job i, $i = 1, 2, \ldots$. The state equation is provided by the Lindley equation , Eq. (1), where A_i is independent of θ, and S_i, D_{i-1} and D_i are functions of θ, hence denoted by $S_i(\theta)$, $D_{i-1}(\theta)$, and $D_i(\theta)$. The delay is expressed in terms of the state variable as

$$d_i(\theta) = D_i(\theta) - A_i. \tag{11}$$

This state-space formulation plays a dual role: it gives a simple formula for the IPA derivative $\frac{\partial}{\partial\theta}L_N(\theta)$, and provides a straightforward argument for the continuity of the sample performance function $L_N(\theta)$ and hence the unbiasedness of IPA. Eq. (3) and the discussion in the ensuing paragraph imply the following formula for $\frac{\partial}{\partial\theta}D_i(\theta)$: Define $k_i(\theta)$ to be the index of the job that started the busy period containing job i. Then,

$$\frac{\partial}{\partial\theta}D_i(\theta) = \sum_{j=k_i(\theta)}^{i}\frac{\partial}{\partial\theta}S_j(\theta). \tag{12}$$

By Eqs. (10)–(12), the IPA derivative is

$$\frac{\partial}{\partial\theta}L_N(\theta) = \frac{1}{N}\sum_{i=1}^{N}\sum_{j=k_i(\theta)}^{i}\frac{\partial}{\partial\theta}S_j(\theta). \tag{13}$$

Note the role of macro-events like the start of busy periods in this formula on the IPA derivative, which was alluded to in the introduction.

As for the continuity of the function $L_N(\theta)$, assume that realizations of the service times, $S_i(\theta)$, are monotone increasing in θ in addition to being differentiable. Furthermore, let θ be constrained to an interval $\Theta := [\theta_1, \theta_2]$ where $0 < \theta_1 < \theta_2 < \infty$, and suppose that the queue is stable at $\theta = \theta_2$. The special recursive structure of the state Eq. (1) preserves continuity and monotonicity, and since by assumption $S_i(\theta)$, $i = 1, \ldots,$ are continuous and monotone increasing in θ, so are $D_i(\theta)$ and hence $d_i(\theta)$ as well. The assumed monotonicity of $\{S_i(\theta)\}$, together with the stability of the queue at θ_2, imply the existence of a random variable K_N having a finite first moment, and providing an upper bound on the IPA derivative $\frac{\partial}{\partial\theta}L_N(\theta)$ over $\theta \in \Theta$. K_N acts as a Lipschitz constant for $L_N(\cdot)$ over Θ thereby implying the unbiasedness of the IPA derivative.

We mention the special case of deterministic service times, or its extension where $S_i(\theta) = \theta + \nu_i$, for a random variable ν_i which is independent of θ. In these cases $\frac{\partial}{\partial\theta}S_i(\theta) = 1$, and Eq. (13) is reduced to the following expression,

$$\frac{\partial}{\partial\theta}L_N(\theta) = \frac{1}{N}\sum_{i=1}^{N}\big(i - k_i(\theta)\big). \tag{14}$$

This is a simple formula which is independent of the probability distribution of the interarrival times. Therefore it has the potential for implementation in real-time, control-based optimization in addition to off-line simulation.

The critical role played by the state Eq. (1) in proving the continuity of $L_N(\theta)$ and deriving the formula for its IPA derivative (13) suggests its extensions to DEDS whose state equations consist of the operators of max and plus. Such systems, classified as max-plus algebras, arise as models in various engineering disciplines beyond queueing networks; see Baccelli, Cohen, Olsder, and Quadrat (1992). For example, a class of decision-free Petri nets, event graphs, acting as models in production control, have had IPA applied to them for parameter optimization in Proth, Sauer, Wardi, and Xie (1996).

The next example concerns a situation where the sample performance functions are discontinuous and hence their IPA derivatives are biased, and how an alternative sample representation based on SPA results in unbiased derivatives. Further details and analysis can be found in Wardi, Gong, Cassandras, and Kallmes (1992).

Example 2. The system is the same as in Example 1 including the state equation Eq. (1), and is subjected to the same assumptions. In addition, we assume that the service-time distribution at a given $\theta \in \Theta$ has a density function. We denote by $F(t, \theta) := P(S(\theta) \le t)$ the distribution function, and its derivative, $\frac{\partial}{\partial t}F(t,\theta) := f(t,\theta)$, its corresponding density function.

Let $J_N(\theta)$ denote the a-priori probability that a job's delay from among the first N jobs exceeds a given threshold-value $r > 0$. Note that $J_N(\theta)$ is an expected-value function, and a sample representation of it can be obtained by simulating the system (queue), computing the number (m) of jobs whose delays exceed r, and taking the term $\frac{m}{N}$. Formally, define $Q_i(\theta) := \mathbf{1}(D_i(\theta) > r)$ where $\mathbf{1}(\cdot)$ denotes the indicator function, and define the sample performance function $L_N(\theta)$ as

$$L_N(\theta) := \frac{1}{N}\sum_{i=1}^{N} Q_i(\theta). \tag{15}$$

Now for a given $\xi \in \Omega$, $Q_i(\theta) \in \{0, 1\}$. Furthermore, since $d_i(\theta)$ is monotone increasing in θ, there are only three possibilities concerning the graph of $Q_i(\theta)$, as follows: (*i*) $Q_i(\theta) = 0$ for every $\theta \in \Theta$, (*ii*) $Q_i(\theta) = 1$ for every $\theta \in \Theta$, and (*iii*) the graph of $Q_i(\theta)$ switched from 0 to 1 at a single point $\bar{\theta}_1 \in \Theta$. By Eq. (15), $L_N(\theta)$ is a step function and hence can be discontinuous. At a given $\xi \in \Omega$, $\frac{\partial}{\partial\theta}L_N(\theta) = 0$ unless θ is a jump point. However, due to the assumed density function of the service times, the probability that a given θ would be a jump point of $L_N(\cdot)$ is zero. Thus, in the course of a sample path at a given θ, the IPA derivative $\frac{\partial}{\partial\theta}L_N(\theta) = 0$ w.p.1. Clearly this IPA derivative is biased since $J_N(\theta)$ is monotone increasing in θ and hence $\frac{\partial}{\partial\theta}J_N(\theta) > 0$.

To get around this problem one can use SPA by conditioning $Q_i(\theta)$ on the waiting time of job i before its service time is drawn. In other words, the term Z in Eq. (5) is the waiting time, denoted by $W_i(\theta)$. Thus, we have that

$$J_N(\theta) = \frac{1}{N}\sum_{i=1}^{N} E\Big[E[Q_i(\theta)|W_i(\theta)]\Big]. \tag{16}$$

By the facts that $d_i(\theta) = W_i(\theta) + S_i(\theta)$ and $Q_i(\theta) = 1(S_i(\theta) > r)$, we have that

$$\begin{aligned} E[Q_i(\theta)|W_i(\theta)] &= P(d_i(\theta) > r|W_i(\theta)) = 1 - P(d_i(\theta) \le r|W_i(\theta)) \\ &= 1 - P(W_i(\theta) + S_i(\theta) \le r|W_i(\theta)) \\ &= 1 - F(r - W_i(\theta), \theta). \end{aligned} \tag{17}$$

We have seen in the previous example that $d_i(\theta)$ is continuous throughout $\theta \in \Theta$ w.p.1, and hence $W_i(\theta) := d_i(\theta) - S_i(\theta)$ is continuous as well. Therefore, by (17) and the assumed density function for the distribution of service times, $E[Q_i(\theta)|W_i(\theta)]$ is a continuous function of θ. By Eq. (15), $\frac{1}{N}\sum_{i=1}^{N} E[Q_i(\theta)|W_i(\theta)]$ is continuous as well. By (16), the latter term is a representative sample of $J_N(\theta)$, and its continuity implies that its sample derivative provides an unbiased estimate of $\frac{\partial}{\partial\theta}L_N(\theta)$. This is the SPA derivative. By Eq. (17) it has the following form,

$$\begin{aligned} &\frac{1}{N}\sum_{i=1}^{N}\frac{\partial}{\partial\theta}Q_i(\theta) \\ &= \frac{1}{N}\sum_{i=1}^{N}\Big(f(r - W_i(\theta), \theta)\frac{\partial}{\partial\theta}W_i(\theta) - \frac{\partial}{\partial\theta}F(r - W_i(\theta), \theta)\Big). \end{aligned} \tag{18}$$

As for the derivative $\frac{\partial}{\partial\theta}W_i(\theta)$, it is zero if job i starts a busy period at the queue, and given by

$$\frac{\partial}{\partial\theta}W_i(\theta) = \sum_{j=k_i(\theta)}^{i-1}\frac{\partial}{\partial\theta}S_j(\theta) \tag{19}$$

if job i does not start a busy period; see (3).

This example points out that SPA can smooth out discontinuities inherent in the sample performance functions, thereby providing unbiased derivative estimators in situations where IPA is biased. However, this comes at the expense of more complicated calculations and, moreover, their reliance on the underlying distributions of the system. The last point is apparent in Eq. (18) which explicitly depends on the distribution function of the service times. Generally, the explicit reliance on such data can make the difference between the use of a derivative estimator in off-line simulation-based optimization, vs. real-time optimization where the sample paths are observed directly from the system. For instance, in the present example, if the distribution function of the service times is not known then the SPA estimator cannot be used in real time. In contrast, $L_N(\theta)$, as given by (15), is model free, and so is its IPA derivative. Of course in this case the IPA derivative is zero w.p.1, hence useless in optimization, be it off line or in real time.

The next example shows that, in contrast to Example 2, practically-computable SPA estimators cannot always be found.

Example 3. Consider the same system as in Example 1 except that the queue has a finite buffer, and jobs arriving at a full queue are being discarded. Let the expected-value performance function, denoted by $J_N(\theta)$, be the job-loss probability from among the first N jobs that arrive at the queue. $J_N(\theta)$ can be represented by the sample performance function, $L_N(\theta)$, defined as the fraction of discarded jobs from among the first N to arrive. Clearly $L_N(\theta)$ is discontinuous in θ and its IPA derivative is biased. Furthermore, an SPA procedure similar to the one used in Example 2 would give biased derivatives as well.

The reason for the failure of SPA to give unbiased derivative estimators is that the discontinuity of the sample performance function is inherent in the state trajectory. To explain this point we draw a comparison between the present system and the one discussed in Example 2. In both systems the state trajectory consists of the process $\{d_i(\theta)\}$, or alternatively, by the related processes $\{A_i\}$ and $\{D_i\}$. In Example 2, the discontinuity is not in $d_i(\theta)$ or $D_i(\theta)$, but rather in the way the sample performance function $L_T(\theta)$ is computed from the state trajectory. Therefore we say that the discontinuity is in the terminal reward, or output function, but not in the state trajectory. Thus, $W_i(\theta) := d_i(\theta) - S_i(\theta)$ is continuous in θ, and therefore, the conditioning of $L_N(\theta)$ on $W_i(\theta)$ can smooth out the discontinuities in the terminal reward. In contrast, in the current example, $W_i(\theta)$ or alternative state-related processes are discontinuous, and hence the conditioning on them will not smooth out the discontinuities of $L_N(\theta)$. We say that the discontinuities are inherent in the state trajectory.

In summary, the discontinuities in Example 2 are only in the terminal reward, while in Example 3, they are inherent in the state trajectory. This difference is the reason that the SPA works well for the former system but not for the current one.

3. Stochastic hybrid systems

In 2002 a new approach to IPA emerged, based on Stochastic Flow Models (SFM) (Cassandras, Wardi, Melamed, Sun, & Panayiotou, 2002).[4] Unlike SPA it does not consist of alternative sample-path representations of $J(\theta)$, but rather on an alternative modelling framework that yields approximate estimates for $J(\theta)$, and, more importantly, whose gradients are *unbiased* and provide approximations to $\nabla J(\theta)$. The SFM concept formulated in Cassandras et al. (2002) grew out of the concept of the fluid queue, and subsequently extended to flow networks and, more generally, to a general setting of Stochastic Hybrid Systems (SHS); see Cassandras, Wardi, Panayiotou, and Yao (2010).

Consider Fig. 1 for an illustration of the SHS concept. The system in question is the DEDS shown in the figure, and the control parameter assigned to it is $\theta \in R^n$. The system with the particular control variable θ generates a sample path which is fed to two places: (*i*) an algorithm which computes $L(\theta)$ and its IPA gradient $\nabla L(\theta)$, and (*ii*) a continuous-flow modelling artifact, indicated by SHS in the figure. An algorithm which is based on SHS computes

[4] To simplify the exposition in the forthcoming discussion, we omit the notational dependence of the sample function $L_T(\theta)$ and its mean $J_T(\theta)$ on T.

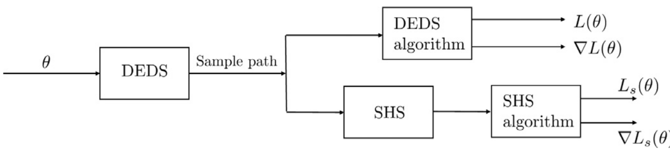

Fig. 1. Relationship between a DEDS and its SHS approximation.

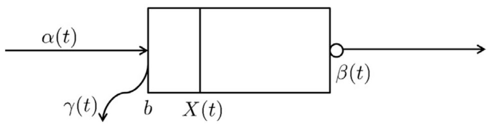

Fig. 2. Elemental fluid-queue model.

the value of another sample performance function, $L_s(\theta)$, and its IPA gradient $\nabla L_s(\theta)$. It is pointed out that the same sample path, associated with the underlying DEDS is used to compute the sample performance functions associated with both the discrete system and the hybrid system, as well as their respective IPA gradients.

Suppose that $\nabla L(\theta)$ is a biased estimator of $\nabla J(\theta)$, and hence it does not provide a good approximation to it. The SHS framework is useful as long as $\nabla L_s(\theta)$ provides a close approximation to $\nabla J(\theta)$, which often is the case when the IPA derivative $\nabla L_s(\theta)$ is unbiased. It must be pointed out that approximating $J(\theta)$ usually is a major concern in performance evaluation or estimation, whereas approximations of $\nabla J(\theta)$ are a concern in optimization and control. Since our interest is in the latter but not the former, we have no intrinsic interest in the quality of approximations provided by $L_s(\theta)$ to $J(\theta)$. In fact, extensive testing by simulation has shown that such approximations can be inadequate for the purpose of performance evaluation, whereas the approximation of $\nabla J(\theta)$ provided by $\nabla L_s(\theta)$ suffices to achieve optimization and control objectives. Examples of this point can be found in Cassandras et al. (2002, 2010).

The first example analyzed in this context concerns a continuous-flow single-server queue with a finite buffer, where fluid arriving at a full queue is being discarded as a matter of overflow. The basic stochastic-flow modelling construct, depicted in Fig. 2, is defined as follows: Given a time-horizon [0, T], let $\{\alpha(t)\}$ and $\{\beta(t)\}$ be the instantaneous fluid-arrival rate to the queue, and the instantaneous fluid service rate at the queue, respectively. These are assumed to be exogenous stochastic processes defined on a common probability space $(\Omega, \mathcal{F}, P)$. Let $b > 0$ denote the size of the buffer. Denote by $X(t)$ and $\gamma(t)$ the instantaneous buffer-contents (amount of fluid in the buffer) and instantaneous spillover rate from the queue due to overflow. Then the processes $\{X(t)\}$ and $\{\gamma(t)\}$ are defined as follows (see Cassandras et al., 2002):

$$\frac{dX(t)}{dt^+} = \begin{cases} 0, & \text{if } X(t) = 0, \text{ and } \alpha(t) \le \beta(t) \\ 0, & \text{if } X(t) = b \text{ and } \alpha(t) \ge \beta(t), \\ \alpha(t) - \beta(t), & \text{otherwise}, \end{cases} \tag{20}$$

and

$$\gamma(t) = \begin{cases} \alpha(t) - \beta(t), & X(t) = b \\ 0, & \text{if } X(t) < b. \end{cases} \tag{21}$$

A typical control variable consists of a parameter of the arrival-rate process, the service-rate process, or the buffer size. For example, $\beta(\theta; t) = \theta\beta(t)$, where θ represents a controlled flow parameter and $\{\beta(t)\}$ is an exogenous process depending on time t but not on θ. This can arise in communication networks where θ represents the total transmission rate of a channel and $\beta(t)$ is the fraction of it which is allocated to a particular session. Observe the notation $\beta(\theta; t)$, indicating that the instantaneous service rate depends on θ. As a result, the buffer-occupancy and spillover-rate processes depend on θ as well via (20) and (21), and hence denoted by $\{X(\theta; t)\}$ and $\{\gamma(\theta; t)\}$. In Cassandras et al. (2002) various sample performance functions of such control variables are considered. We next present the first SFM analyzed from the standpoint of IPA, which exhibits the salient features of an extensive suite of fluid queueing networks.

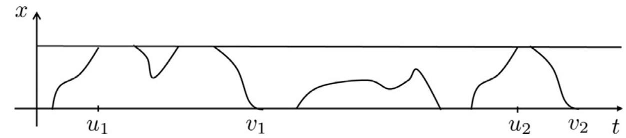

Fig. 3. Typical state trajectory $\{X(\theta; t)\}$.

Consider the case where the control parameter is the buffer capacity, and the performance function is the amount of fluid which is discarded during a given horizon interval [0, T]. Thus, $\theta = b$, and the performance function, denoted by $L_\gamma(\theta)$, is

$$L_\gamma(\theta) := \int_0^T \gamma(\theta; t) dt. \tag{22}$$

Note that $L_\gamma(\theta)$ is related to the fraction of discarded fluid from the total arrival volume during the time $t \in [0, T]$, which is $L_\gamma(\theta) / \int_0^T \alpha(t) dt$.

Regarding the IPA derivative $\frac{dL_\gamma(\theta)}{d\theta}$, we note that it is unbiased since the function $L_\gamma(\theta)$ is continuous. The following formula for it was obtained in Cassandras et al. (2002): Define N_T as the number of lossy busy periods in the horizon interval [0, T], namely the number of busy periods during which the buffer becomes full at some time. Then,

$$\frac{dL_\gamma(\theta)}{d\theta} = -N_T. \tag{23}$$

As an example, consider the realization of the state trajectory $\{X(\theta; t)\}$ shown in Fig. 3. It is evident that the first and third busy periods are lossy while the second in not, therefore $\frac{dL_\gamma(\theta)}{d\theta} = -2$.

As another example, consider the cumulative workload as a function of the buffer capacity. Thus, $\theta = b$ as in the previous example, and let the performance function, denoted by $L_x(\theta)$, be defined as

$$L_x(\theta) = \int_0^T X(\theta; t) dt. \tag{24}$$

Note that the term $L_x(\theta) / \int_0^T \alpha(t) dt$ serves to approximate the average delay of fluid "molecules" by Little's law. As for the IPA derivative, fix $\theta > 0$, and let B_m, $m = 1, \ldots, M$ denote the lossy busy periods in the interval [0, T] in increasing order. For every $m = 1, \ldots, M$, let $u_m \in B_m$ denote the first time the buffer becomes full in B_m, and let v_m be the end-time of B_m. Then $\frac{dL_x(\theta)}{d\theta}$ has the following form,

$$\frac{dL_x(\theta)}{d\theta} = \sum_{m=1}^{M} (v_m - u_m). \tag{25}$$

For example, in Fig. 3, $\frac{dL_x(\theta)}{d\theta} = \sum_{m=1}^{2} (v_m - u_m)$.

We point out that Eqs. (23) and (25) were derived (in Cassandras et al., 2002) under minimal assumptions on the system. In fact, the only assumption made is that the processes $\{\alpha(t)\}$ and $\{\beta(t)\}$ be piecewise continuous and of bounded variation in $t \in [0, T]$ w.p.1. Therefore, for the purpose of computing Eqs. (23) and (25), these processes can be generated not only from an SFM but also from a DEDS. Such DEDS can be a modelling artifact of the system or the system itself. Furthermore, the formulas (23) and (25) do not depend on observations of the detailed dynamics associated with arrivals or departures of each customer at the queue, nor on the instantaneous values of $\alpha(t)$ or $\beta(t)$. Instead, they require only the observations of macro events like the beginning and end of busy periods and full-buffer periods. Thus, comparing and contrasting these formulas with the analogous equations derived for IPA in the traditional, discrete-queueing setting (e.g., Cassandras & Lafortune, 2008), we see that the SFM-based formulas are unbiased, simpler, and do not require any details of the probability laws underscoring the arrival and service processes. Due to the last property we say that the IPA derivative is *nonparametric*. Furthermore, as mentioned above, the IPA derivatives can be computed from data generated from DEDS as well as SFM. In fact, Cassandras et al. (2002) and subsequent papers on IPA in the SFM setting (e.g., Cassandras et al., 2010; Sun, Cassandras, & Panayiotou, 2004; Zhang & Cassandras, 2002 and references therein) report on successful solutions of DEDS optimization problems where the IPA gradients are computed from SFM-derived formulas. All of this suggests that the SFM setting provides an alternative framework to DEDS for the application of IPA, which holds out promise of real-time optimization via closed-loop control.

As we mentioned earlier, following the basic formulation of the SFM and derivation of the aforementioned IPA gradients, there were several efforts to extend the model and results to fluid queueing systems and other multiflow networks. The main results concern the development of IPA gradients for prototypical problems, and their implementation in optimization environments. By and large the simplicity and unbiasedness of the IPA gradients are maintained. The nonparametric property and reliance only on observations of macro events has been almost maintained, but held ground for close approximations where, practically, the errors can be neglected. Moreover, the key structure of perturbation propagation, which has rendered IPA attractive from its onset, applies to the general framework of *Stochastic Hybrid Systems* (SHS) based on a formal calculus (referred to as the *IPA Calculus*) of event-driven propagations as described in Cassandras et al. (2010). We review this general setting in what follows.

3.1. The IPA calculus for hybrid systems

We begin by adopting a standard hybrid automaton formalism to model the operation of a (generally stochastic) hybrid system as in Cassandras et al. (2010). Let $q \in Q$ (a countable set) denote the discrete state (or mode) and $x \in X \subseteq \mathbb{R}^n$ denote the continuous state. Let $\upsilon \in \Upsilon$ (a countable set) denote a discrete control input and $u \in U \subseteq \mathbb{R}^m$ a continuous control input. Similarly, let $\delta \in \Delta$ (a countable set) denote a discrete disturbance input and $d \in D \subseteq \mathbb{R}^p$ a continuous disturbance input. The state evolution is determined by means of (*i*) a vector field f: $Q \times X \times U \times D \to X$, (*ii*) an invariant (or domain) set *Inv*: $Q \times \Upsilon \times \Delta \to 2^X$, (*iii*) a guard set *Guard*: $Q \times Q \times \Upsilon \times \Delta \to 2^X$, and (*iv*) a reset function r: $Q \times Q \times X \times \Upsilon \times \Delta \to X$.

A sample path of such a system consists of a sequence of intervals of continuous evolution followed by a discrete transition. The system remains at a discrete state q as long as the continuous (time-driven) state x does not leave the set $Inv(q, \upsilon, \delta)$. If x reaches a set $Guard(q, q', \upsilon, \delta)$ for some $q' \in Q$, a discrete transition can take place. If this transition does take place, the state instantaneously resets to (q', x') where x' is determined by the reset map $r(q, q', x, \upsilon, \delta)$. Changes in υ and δ are discrete events that either *enable* a transition from q to q' by making sure $x \in Guard(q, q', \upsilon, \delta)$ or *force* a transition out of q by making sure $x \notin Inv(q, \upsilon, \delta)$. We will also use $\mathcal{E}$ to denote the set of all events that cause discrete state transitions and will classify events in a manner that suits the purposes of perturbation analysis.

Let $\theta \in \Theta \subset \mathbb{R}^l$ be a controllable parameter vector, where Θ is a given compact, convex set. This may contain system design parameters or parameters that characterize a policy used in controlling this system. The disturbance input $d \in D$ encompasses various random processes that affect the evolution of the state (q, x) so that, in general, we can deal with an SHS. We will assume that all such processes are defined over a common probability space, $(\Omega, \mathcal{F}, P)$. Let us fix a particular value of the parameter $\theta \in \Theta$ and study a resulting sample path of the SHS. Over such a sample path, let $\tau_k(\theta)$, $k = 1, 2, \dots,$ denote the occurrence times of the discrete events in increasing order, and define $\tau_0(\theta) = 0$ for convenience. We will use the notation τ_k instead of $\tau_k(\theta)$ when no confusion arises. The continuous state is also generally a function of θ, as well as of t, and is thus denoted by $x(\theta, t)$. Over an interval $[\tau_k(\theta), \tau_{k+1}(\theta))$, the system is at some mode during which the time-driven state satisfies:

$$\dot{x} = f_k(x, \theta, t) \tag{26}$$

where $\dot{x}$ denotes $\frac{\partial x}{\partial t}$. Note that we suppress the dependence of f_k on the inputs $u \in U$ and $d \in D$ and stress instead its dependence on the parameter θ which may generally affect either u or d or both. The purpose of perturbation analysis is to study how changes in θ influence the state $x(\theta, t)$ and the event times $\tau_k(\theta)$ and, ultimately, how they influence interesting performance metrics which are generally expressed in terms of these variables. Note that under standard technical conditions (see Cassandras et al., 2010), Eq. (26) has a unique solution w.p.1 for a given initial boundary condition $x(\theta, \tau_k)$ at time $\tau_k(\theta)$.

An event occurring at time $\tau_{k+1}(\theta)$ triggers a change in the mode of the system, which may also result in new dynamics represented by f_{k+1}, although this may not always be the case; for example, two modes may be distinct because the state $x(\theta, t)$ enters a new region where the system's performance is measured differently without altering its time-driven dynamics (i.e., $f_{k+1} = f_k$). The event times $\{\tau_k(\theta)\}$ play an important role in defining the interactions between the time-driven and event-driven dynamics of the system.

We now classify events that define the set $\mathcal{E}$ as follows:

1. Exogenous events. An event is *exogenous* if it causes a discrete state transition at time τ_k independent of the controllable vector θ and satisfies $\frac{d\tau_k}{d\theta} = 0$. Exogenous events typically correspond to uncontrolled random changes in input processes.

2. Endogenous events. An event occurring at time τ_k is *endogenous* if there exists a continuously differentiable function g_k : $\mathbb{R}^n \times \Theta \to \mathbb{R}$ such that

$$\tau_k = \min\{t > \tau_{k-1} \ : \ g_k(x(\theta, t), \theta) = 0\} \tag{27}$$

The function g_k normally corresponds to a guard condition in a hybrid automaton model.

3. Induced events. An event at time τ_k is *induced* if it is triggered by the occurrence of another event at time $\tau_m \le \tau_k$. The triggering event may be exogenous, endogenous, or itself an induced event. The events that trigger induced events are identified by a subset of the event set, $\mathcal{E}_I \subseteq \mathcal{E}$.

Next, consider a performance function of the control parameter θ:

$$J(\theta; x(\theta, 0), T) = E[\mathcal{L}(\theta; x(\theta, 0), T)]$$

where $\mathcal{L}(\theta; x(\theta, 0), T)$ is a sample function of interest evaluated in the interval $[0, T]$ with initial conditions $x(\theta, 0)$. For simplicity, we write $J(\theta)$ and $\mathcal{L}(\theta)$. Suppose that there are N events (generally dependent on θ) occurring during the time interval $[0, T]$ and define $\tau_0 = 0$ and $\tau_{N+1} = T$. For functions $L_k : \mathbb{R}^n \times \Theta \times \mathbb{R}^+ \to \mathbb{R}$, $k = 1, \dots, N$, set

$$\mathcal{L}(\theta) := \sum_{k=0}^{N} \int_{\tau_k}^{\tau_{k+1}} L_k(x, \theta, t) dt \tag{28}$$

where we reiterate that $x = x(\theta, t)$ is a function of θ and t. Given that we do not wish to impose any limitations (other than mild technical conditions) on the random processes that characterize the discrete or continuous disturbance inputs in our hybrid automaton model, it is infeasible to obtain closed-form expressions for $J(\theta)$. Therefore, for the purpose of optimization, we resort to iterative methods such as stochastic approximation algorithms which are driven by estimates of the cost function gradient with respect to the parameter vector of interest. Thus, we are interested in estimating $dJ/d\theta$ based on sample path data, where a sample path of the system may be directly observed or it may be obtained through simulation. We then seek to obtain θ^* minimizing $J(\theta)$ through an iterative scheme of the form

$$\theta_{n+1} = \theta_n - \eta_n H_n(\theta_n; x(\theta, 0), T, \omega_n), \quad n = 0, 1, \ldots \tag{29}$$

where $H_n(\theta_n; x(0), T, \omega_n)$ is an estimate of $dJ/d\theta$ evaluated at θ and based on information obtained from a sample path denoted by ω_n, and $\{\eta_n\}$ is an appropriately selected step size sequence. In order to execute an algorithm such as (29), we need the estimate $H_n(\theta_n)$ of $dJ/d\theta$. As already discussed, the IPA approach is based on using the sample derivative $d\mathcal{L}/d\theta$ as an estimate of $dJ/d\theta$ capitalizing on the fact that $d\mathcal{L}/d\theta$ can be obtained from observable sample path data alone and, usually, in a very simple manner that can be readily implemented on line. Moreover, it is often the case that $d\mathcal{L}/d\theta$ is an *unbiased* estimate of $dJ/d\theta$, a property that allows us to use (29) in obtaining θ^* (we will return to this issue later).

Let us now fix $\theta \in \Theta$, consider a particular sample path, and assume for the time being that all derivatives mentioned in the sequel do exist. To simplify notation, we define the following for all state and event time sample derivatives:

$$x'(t) \equiv \frac{\partial x(\theta, t)}{\partial \theta}, \quad \tau_k' \equiv \frac{\partial \tau_k}{\partial \theta}, \quad k = 0, \ldots, N \tag{30}$$

In addition, we will write $f_k(t)$ instead of $f_k(x, \theta, t)$ whenever no ambiguity arises. By taking derivatives with respect to θ in (26) on the interval $[\tau_k(\theta), \tau_{k+1}(\theta))$ we get

$$\frac{d}{dt}x'(t) = \frac{\partial f_k(t)}{\partial x}x'(t) + \frac{\partial f_k(t)}{\partial \theta} \tag{31}$$

The boundary (initial) condition of this linear equation is specified at time $t = \tau_k$, and by writing (26) in an integral form and taking derivatives with respect to θ when $x(\theta, t)$ is continuous in t at $t = \tau_k$, we obtain for $k = 1, \ldots, N$:

$$x'(\tau_k^+) = x'(\tau_k^-) + \left[f_{k-1}(\tau_k^-) - f_k(\tau_k^+)\right]\tau_k' \tag{32}$$

We note that whereas $x(\theta, t)$ is often continuous in t, $x'(t)$ may be discontinuous in t at the event times τ_k, hence the left and right limits above are generally different. If $x(\theta, t)$ is not continuous in t at $t = \tau_k$, the value of $x(\tau_k^+)$ is determined by the reset function $r(q, q', x, \upsilon, \delta)$ discussed earlier and

$$x'(\tau_k^+) = \frac{dr(q, q', x, \upsilon, \delta)}{d\theta} \tag{33}$$

Furthermore, once the initial condition $x'(\tau_k^+)$ is given, the linearized state trajectory $\{x'(t)\}$ can be computed in the interval $t \in [\tau_k(\theta), \tau_{k+1}(\theta))$ by solving (31) to obtain:

$$x'(t) = e^{\int_{\tau_k}^{t} \frac{\partial f_k(u)}{\partial x} du}\left[\int_{\tau_k}^{t} \frac{\partial f_k(v)}{\partial \theta} e^{-\int_{\tau_k}^{t} \frac{\partial f_k(u)}{\partial x} du} dv + c_k\right] \tag{34}$$

with the constant c_k determined from $x'(\tau_k^+)$ in (32), since $x'(\tau_k^-)$ is the final-time boundary condition in the interval $[\tau_{k-1}(\theta), \tau_k(\theta))$, or it is obtained from (33).

Clearly, to complete the description of the trajectory of the linearized system (31)–(32) we have to specify the derivative τ_k' which appears in (32). Since τ_k, $k = 1, 2, \ldots$, are the mode-switching times, these derivatives explicitly depend on the interaction between the time-driven dynamics and the event-driven dynamics, and specifically on the type of event occurring at time τ_k. Using the event classification given earlier, we have the following.

1. Exogenous events. By definition, such events are independent of θ, therefore $\tau_k' = 0$.

2. Endogenous events. In this case, (27) holds and taking derivatives with respect to θ we get:

$$\frac{\partial g_k}{\partial x}\left[x'(\tau_k^-) + f_k(\tau_k^-)\tau_k'\right] + \frac{\partial g_k}{\partial \theta} = 0 \tag{35}$$

which, assuming $\frac{\partial g_k}{\partial x} f_k(\tau_k^-) \neq 0$, can be rewritten as

$$\tau_k' = -\left[\frac{\partial g_k}{\partial x} f_k(\tau_k^-)\right]^{-1}\left(\frac{\partial g_k}{\partial \theta} + \frac{\partial g_k}{\partial x}x'(\tau_k^-)\right) \tag{36}$$

3. Induced events. If an induced event occurs at $t = \tau_k$, the value of τ_k' depends on the derivative τ_m' where $\tau_m \leq \tau_k$ is the time when the associated triggering event takes place. The event induced at τ_m will occur at some time $\tau_m + \omega(\tau_m)$, where $\omega(\tau_m)$ is a random variable which is generally dependent on the continuous and discrete states $x(\tau_m)$ and $q(\tau_m)$ respectively. This implies the need for additional state variables, denoted by $y_m(\theta, t)$, $m = 1, 2, \ldots$, associated with events occurring at times τ_m, $m = 1, 2 \ldots$ The role of each such state variable is to provide a "timer" activated when a triggering event occurs. Recalling that triggering events are identified as belonging to a set $\mathcal{E}_I \subseteq \mathcal{E}$, let e_k denote the event occurring at τ_k and define $\digamma_k = \{m : e_m \in \mathcal{E}_I, \ m \leq k\}$ to be the set of all indices with corresponding triggering events up to τ_k. Omitting the dependence on θ for simplicity, the dynamics of $y_m(t)$ are then given by

$$\dot{y}_m(t) = \begin{cases} -C(t) & \tau_m \leq t < \tau_m + \omega(\tau_m), \ m \in \digamma_m \\ 0 & \text{otherwise} \end{cases} \tag{37}$$

$$y_m(\tau_m^+) = \begin{cases} y_0 & y_m(\tau_m^-) = 0, \ m \in \digamma_m \\ 0 & \text{otherwise} \end{cases}$$

where y_0 is an initial value for the timer $y_m(t)$ which decreases at a "clock rate" $C(t) > 0$ until $y_m(\tau_m + \omega(\tau_m)) = 0$ and the associated induced event takes place. Clearly, these state variables are only used for induced events, so that $y_m(t) = 0$ unless $m \in \digamma_m$. The value of y_0 may depend on θ or on the continuous and discrete states $x(\tau_m)$ and $q(\tau_m)$, while the clock rate $C(t)$ may depend on $x(t)$ and $q(t)$ in general, and possibly θ. However, in most simple cases where we are interested in modelling an induced event to occur at time $\tau_m + \omega(\tau_m)$, we have $y_0 = \omega(\tau_m)$ and $C(t) = 1$, i.e, the timer simply counts down for a total of $\omega(\tau_m)$ time units until the induced event takes place. Henceforth, we will consider $y_m(t)$, $m = 1, 2, \ldots$, as part of the continuous state of the SHS and, similar to (30), we set

$$y_m'(t) \equiv \frac{\partial y_m(t)}{\partial \theta}, \quad m = 1, \ldots, N. \tag{38}$$

For the common case where y_0 is independent of θ and $C(t)$ is a constant $c > 0$ in (37), it is shown in Cassandras et al. (2010) that $\tau_k' = \tau_m'$.

With the inclusion of the state variables $y_m(t)$, $m = 1, \ldots, N$, the derivatives $x'(t)$, τ_k', and $y_m'(t)$ can be evaluated through (31)–(36) along with (38). This very general set of equations represents the "IPA calculus". In general, the derivative evaluation is recursive over the event (mode switching) index $k = 0, 1, \ldots$ In some cases, however, it can be reduced to simple expressions, as seen in the analysis of many SFMs discussed earlier in this section. Observe that if a SHS does not involve induced events and if the state does not experience discontinuities when a mode-switching event occurs, then the full extent of the IPA calculus reduces to three equations: (*i*) Eq. (34), which describes how the state derivative $x'(t)$ evolves over $[\tau_k(\theta), \tau_{k+1}(\theta))$, (*ii*) Eq. (32), which specifies the initial condition ξ_k in (34), and (*iii*) Either $\tau_k' = 0$ or (36) is satisfied, depending on the event type at $\tau_k(\theta)$, which specifies the event time derivative present in (32).

Now the IPA derivative $d\mathcal{L}/d\theta$ can be obtained by taking derivatives in (28) with respect to θ:

$$\frac{d\mathcal{L}(\theta)}{d\theta} = \sum_{k=0}^{N} \frac{d}{d\theta} \int_{\tau_k}^{\tau_{k+1}} L_k(x,\theta,t)dt. \tag{39}$$

Applying the Leibnitz rule we obtain, for every $k = 0, \ldots, N$,

$$\begin{aligned} &\frac{d}{d\theta}\int_{\tau_k}^{\tau_{k+1}} L_k(x,\theta,t)dt \\ &= \int_{\tau_k}^{\tau_{k+1}} \left[\frac{\partial L_k}{\partial x}(x,\theta,t)x'(t) + \frac{\partial L_k}{\partial \theta}(x,\theta,t) \right] dt \\ &\quad + L_k(x(\tau_{k+1}),\theta,\tau_{k+1})\tau'_{k+1} - L_k(x(\tau_k),\theta,\tau_k)\tau'_k \end{aligned} \tag{40}$$

where $x'(t)$ and τ'_k are determined through (31)–(36). What makes IPA appealing is the simple form the right-hand-side above often assumes.

The IPA calculus formalism derived in this subsection via Eqs. (26)–(40) appears to be quite complicated. However, this is due to its wide scope in stochastic hybrid systems. In fact, for a particular system, or a class of systems with a specific structure, the resulting IPA derivative can be quite simple and elegant. For a simple example, consider the fluid queue depicted in Fig. 2, where the control variable θ is the buffer size, and the sample performance function is the total loss volume, $L_\gamma(\theta)$, as defined by Eq. (22). The functions $f_k(x,\ \theta,\ t)$, $k = 1, 2, \ldots,$ are given by the right-hand side of Eq. (20), namely, $f_k(x,\theta,t) = 0$ if t lies in the interior of an empty-buffer period or a full-buffer period, and $f_k(x,\theta,t) = \alpha(t) - \beta(t)$ otherwise. There are exogenous events and endogenous events but no induced events. Exogenous events are discontinuities in realizations of the random functions $\alpha(\cdot)$ or $\beta(\cdot)$ as well as the end of an empty-buffer period or a full-buffer period. Endogenous events are the starting times of full-buffer periods or empty-buffer periods. The functions $g_k(x,\ \theta)$ (see Eq. (27)) have the respective forms $g_k(x,\theta) = x - \theta$ if τ_k is the starting time of a full-buffer period, and $g_k(x,\theta) = x$ if τ_k is the starting time of an empty-buffer period. The analysis of the IPA derivative $\frac{dL_\gamma(\theta)}{d\theta}$, carried out in Cassandras et al. (2002), essentially follows the formalism developed in this subsection albeit for a simple example, and yielded the result expressed in Eq. (23).

We conclude this overview of the IPA calculus with a comment on the unbiasedness of the IPA derivative $d\mathcal{L}/d\theta$. This IPA derivative is indeed unbiased under very mild technical conditions, as shown in Cassandras et al. (2010). The most crucial condition is the continuity of the sample performance function $\mathcal{L}(\theta)$, which in many SHS is readily guaranteed. An additional condition is the Lipschitz continuity of $\mathcal{L}(\theta)$ which follows from upper boundedness of $|\frac{d\mathcal{L}(\theta)}{d\theta}|$ by an absolutely integrable random variable, generally a weak assumption.

4. Recent and current trends

This section presents some of the main research directions in IPA which emerged during the past decade. In particular we discuss applications to large-scale Markov processes and stochastic hybrid systems, performance regulation of systems, and the use of event-driven (as opposed to time-driven) methods for control and optimization.

4.1. IPA of Markov systems and stochastic optimization

Until the mid 1990s, IPA was largely limited to "infinitesimal" perturbations, and could not be applied to perturbations with finite size. Around this time, however, it was realized that the perturbation realization principle applies to finite jumps of states as well. This made it possible to develop IPA algorithms for Markov processes.

Consider an irreducible and aperiodic Markov chain $\mathbf{X} = \{X_n : n \geq 0\}$ on a finite state space $\mathcal{S} = \{1, 2, \cdots, M\}$ with transition probability matrix $P = [p(j|i)] \in [0,1]^{M\times M}$. Let $\pi = (\pi_1, \ldots, \pi_M)$ be the vector representing its steady-state probabilities, and $f = (f_1, f_2, \cdots, f_M)^T$ be the performance vector, where "T" represents its transpose. We have $Pe = e$, where $e = (1, 1, \cdots, 1)^T$ is an M-dimensional vector whose all components equal 1, and $\pi = \pi P$. The performance measure is the long term average defined as

$$\eta = \sum_{i=1}^{M} \pi_i f_i = \pi f = \lim_{L\to\infty} \frac{1}{L} \sum_{l=0}^{L-1} f(X_l) = \lim_{L\to\infty} \frac{F_L}{L}, \quad w.p.1, \tag{41}$$

where $F_L := \sum_{l=0}^{L-1} f(X_l)$.

Let P' be another irreducible transition probability matrix on the same state space. Suppose P changes to $P(\delta) = P + \delta Q = \delta P' + (1-\delta)P$, with $\delta > 0$, $Q = P' - P = [q(j|i)]$, and the reward function f keeps the same. We have $Qe = 0$. The performance measure will change to $\eta(\delta) = \eta + \Delta\eta(\delta)$. The derivative of η in the direction of Q is defined as

$$\frac{d\eta(\delta)}{d\delta} = \lim_{\delta\to 0} \frac{\Delta\eta(\delta)}{\delta}. \tag{42}$$

In this system, a perturbation means that the system is perturbed from one state i to another state j. Following the same idea as in (7), we study two independent Markov chains $\mathbf{X} = \{X_n; n \geq 0\}$ and $\mathbf{X}' = \{X'_n; n \geq 0\}$ with $X_0 = i$ and $X'_0 = j$; both of them have the same transition matrix P. The *realization factor* is defined as in (Cao, 2007):

$$\begin{aligned} d(i,j) &= \lim_{L\to\infty} E\left[\sum_{l=0}^{L-1} (f(X'_l) - f(X_l)) \,\middle|\, X_0 = i,\ X'_0 = j \right], \\ &\quad i, j = 1, \ldots, M\,. \end{aligned} \tag{43}$$

Thus, $d(i, j)$ represents the average effect of a jump from i to j on F_L in (41). From (43), it is easy to see that $d(i, j)$ satisfies the conservation law as in physics:

$$g(i,k) = g(i,j) + g(j,k), \quad i,j,k \in \mathcal{S},$$

Thus, we can define a vector $g = (g(1), g(2), \cdots, g(M))^T$, called *performance potential*, such that

$$d(i,j) = g(j) - g(i), \quad i,j \in \mathcal{S},$$

and we can verify that it satisfies the *Poisson equation*

$$(I - P + e\pi)g = f, \tag{44}$$

where I is the $M \times M$ identity matrix. Multiplying both sides of the Poisson equation with π on the left, we get

$$\pi g = \pi f = \eta. \tag{45}$$

Multiplying both sides of the Poisson equation with π' on the left yields

$$\pi' Q g = \pi'(P' - P)g = \pi'(I - P)g = \pi' f - \pi g = \pi' f - \eta.$$

That is,

$$\eta' - \eta = \pi' Q g. \tag{46}$$

Setting $P(\delta) = P + \delta Q$ and $\eta' = \eta(\delta)$ and letting $\delta \to 0$ in (46), we get the desired performance derivative along the direction Q:

$$\frac{d\eta(\delta)}{d\delta} = \pi Q g. \tag{47}$$

This equation is consistent with the well known results in the area of matrix-algebra Markov chain perturbation analysis, which dates back to Schweitzer's work in 1968 (Schweitzer, 1968). Caswell (2013) contains many interesting results on this topic;

also see Abbas, Berkhout, and Heidergott (2016) and the reference therein for more discussion. However, the goal of perturbation analysis of Markov chains, in the context of this paper, is not to derive the formula per se, but rather to offer a dynamic point of view for these sensitivity formulas, hence leading naturally to many applications to dynamic systems, including the sample path based learning and optimization algorithms, and a new approach to Markov decision problems and event-based optimization, as described below.

The derivative in Eq. (47) can be used in performance optimization (Marbach & Tsitsiklis, 2001). Efficient algorithms can be derived for estimating g and estimating the derivative $\frac{d\eta(\delta)}{d\delta}$ directly (Cao, 2007; 2009). We have

$$\frac{d\eta}{d\delta} = \sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{S}}\pi(i)q(j|i)g(j) = \sum_{i\in\mathcal{S}}\sum_{j\in\mathcal{S}}\pi(i)p(j|i)\left[\frac{q(j|i)}{p(j|i)}g(j)\right]$$
$$= E\left[\frac{q(X_{k+1}|X_k)}{p(X_{k+1}|X_k)}g(X_{k+1})\right], \tag{48}$$

where "E" denotes the steady-state probability, and $g(i)$ can be estimated from a sample path. Based on this equation, various efficient on-line algorithms can be developed to estimate the performance derivatives Cao (2005). Now, there is a new area in reinforcement learning, called policy gradients, devoted to this subject, see e.g., Baxter and Bartlett (2001), Baxter, Bartlett, and Weaver (2001) and Sutton, McAllester, Singh, and Mansour (2000), etc.

If the reward function also changes from f to f', let $h := f' - f$. It is easy to check that

$$\eta' - \eta = \pi'(Qg + h) = \pi'\{(P'g + f') - (Pg + f)\}. \tag{49}$$

This is the Performance Difference Formula (PDF); it initiates a new direction in performance optimization, the *direct-comparison based approach*. In fact, it is observed that the PDF contains all the information in comparing the performance of any two policies, and an optimality condition can be simply derived from this equation without dynamic programming or discounting for long-run average performance. For example, because $\pi' > 0$, from (49), we conclude

$$If\ P'g + f' \leq\ Pg + f,\ then\ \eta' \leq \eta. \tag{50}$$

This leads to the optimality condition: a policy (P^*, f^*) with potential g^* is optimal if and only if $Pg^* + f \leq P^*g^* + f^*$ for all policies P. Policy iteration algorithms can also be developed from (49).

The Direct-Comparison (DC) based approach is an alternative to dynamic programming (DP) to performance optimization of dynamic systems. As illustrated above, this approach is very simple and intuitive for long-run average performance; in fact, a complete theory based on nth bias optimality for long-run average performance can be developed with no discounting (Cao, 2007; 2009). Next, the PDF provides global information to performance comparison in the entire period; while dynamic programming works backwards in time at a particular time instant (continuous or discrete), and hence it only provides local information. Therefore, the DC-based approach opens a new horizon for problems requiring global considerations.

For example, the approach naturally solves a long existing issue in time non-homogenous Markov systems, the under selectivity, which means that in performance optimization of time non-homogenous systems, where the transition probabilities and reward functions are different at different time $k = 1, 2, \cdots$, the long-run average performance, and for that matter its optimal policy, does not depend the actions (transition probabilities and rewards) in any finite periods (Cao, 2015). The approach has also been applied to stochastic control problems with diffusion processes (continuous time and continuous states); it solves the control problem with non-smooth value functions without resorting to viscosity solutions (Cao, 2017).

Eqs. (47) and (49) provide a sensitivity-based view to performance optimization. For problems where the performance is not additive, (47) may be used. The DC-based approach links both together naturally, Research in this direction is ongoing and the influence of the sensitivity-based view extends beyond the area of DEDS.

Last but not the least, combined with the aggregation technique, the DC-based approach leads to the theory of *event-based optimization (control)*, in which control actions depend on events rather than the states. This may dramatically reduce the computation since the number of events (event space) is much smaller than that of states (state space). Conditions have been derived under which Hamilton-Jacobi-Bellman (HJB) type of optimality equation holds for event-based control; because the sequence of events is not Markov and aggregation is used, approximation is usually involved, see Cao (2007) and Xia, Xia, and Cao (2014).

More precisely, let e denote an event, which can be understood as a statistics based on the past history, e.g., an estimate of the state in a partially observable Markov decision process (POMDP), or simply a short period of the history, or a physical event such as a customer arrival or departure in a queueing network; and let V be the space of events. The advantage of using such events instead of states is that event is observable, while state may not, and the number of events may be much less than that of states. Let $\pi(e|i)$ be the conditional steady state probability of event e given the current state is i, and $p(j|i, e)$ be the conditional transition probability from i to j given the event e. Then from the PDF (49), we have

$$\begin{aligned}
&\eta' - \eta \\
&= \pi'[(P' - P)g + (f' - f)] \\
&= \sum_{i\in\mathcal{S}}\pi'(i)\left\{\sum_{i\in\mathcal{S}}\sum_{e\in V}\pi'(e|i)[p'(j|i,e) - p(j|i,e)]g(j) + [f'(i) - f(i)]\right\} \\
&= \sum_{i\in\mathcal{S}}\left\{\sum_{i\in\mathcal{S}}\sum_{e\in V}\pi'(i,e)[p'(j|i,e) - p(j|i,e)]g(j) + \pi'(i)[f'(i) - f(i)]\right\} \\
&= \sum_{e\in V}\pi'(i)\sum_{i\in\mathcal{S}}\pi'(i|e)\left\{\sum_{j\in\mathcal{S}}[p'(j|i,e) - p(j|i,e)]g(j) + [f'(i) - f(i)]\right\}
\end{aligned} \tag{51}$$

If we further assume that

$$\pi'(i|e) = \pi(i|e), \tag{52}$$

then the above PDF becomes

$$\begin{aligned}
&\eta' - \eta \\
&= \sum_{e\in V}\pi'(i)\sum_{i\in\mathcal{S}}\pi(i|e)\left\{\sum_{j\in\mathcal{S}}[p'(j|i,e) - p(j|i,e)]g(j) + [f'(i) - f(i)]\right\}.
\end{aligned} \tag{53}$$

Therefore, if

$$\begin{aligned}
&\sum_{i\in\mathcal{S}}\pi(i|e)\left\{\sum_{j\in\mathcal{S}}p'(j|i,e)g(j) + f'(i)\right\} \\
&\geq \sum_{i\in\mathcal{S}}\pi(i|e)\left\{\sum_{j\in\mathcal{S}}p(j|i,e)g(j) + f(i)\right\},
\end{aligned} \tag{54}$$

then $\eta' \geq \eta$. Define

$$Q(e, a) = \sum_{i\in\mathcal{S}}\pi(i|e)\left\{\sum_{j\in\mathcal{S}}p^a(j|i,e)g(j) + f^a(i)\right\},$$

in which a is the action taken at event e, and the superscript "a" denote quantities associated with action a. $Q(e, a)$ is called Q factor associated with event e and action a. Then policy iteration and the optimal policy can be determined according to (54). When condition (52) does not hold, approximate results can be developed, see Xia et al. (2014) and Cao (2007).

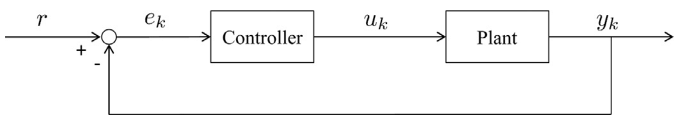

Fig. 4. Control system.

4.2. Performance regulation

Emerging applications of IPA concern the tracking of a referencee input to a dynamical system by its output process. An abstract, discrete-time single-input-single-output system is depicted in Fig. 4, where k denotes time, $r \in R$ is the reference input, y_k is the system's output, u_k is the control input to the plant, and $e_k := r - y_k$ is the error signal. The plant generally is a time-varying dynamical system lacking an accurate model and subjected to unpredictable variations.

As an example of interest, it is desirable to regulate the instructions' throughput in a computer processor by adjusting its clock rate, or frequency. More specifically, the time axis is divided into contiguous periods called *control cycles*, during each of which the frequency is set (fixed) and the average throughput is measured. At the end of the control cycle the frequency is changed by the controller according to the difference between the given target-throughput (setpoint) and the average throughput. In this setting the time counter k indicates the index of the control cycle denoted henceforth by C_k, u_k is the value of the clock frequency during C_k, and y_k is the average instruction-throughput computed during C_k.

The plant in Fig. 4 is the processor, and any model thereof would describe the frequency-to-throughput relationship. Since we enact a real-time control, there is no need for a model to close the loop since the output y_k is measured. However, we shall see that a model is needed in order to implement the controller that we have in mind. An established model of an out-of-order architecture is provided by a queueing system (see Hennessey & Patterson, 2012, or a simplified exposition in Wardi, Seatzu, Chen, & Yalamanchili, 2016) which defies analysis. We use it nonetheless in an effective way, as described below.

Returning to the abstract system in Fig. 4, the objective is to design a controller which can deliver the desired tracking without a detailed knowledge of the plant-model while facing wide-ranging variations in the system's input-output relationships. Moreover, the controller has to achieve that in very short time-frames and hence by simple computations. Tracking typically involves an integrator in the loop, and to have the controller be as simple as possible we first considered a standalone integrator. Now it is well known that a standalone integrator may destabilize the closed-loop system and otherwise have poor stability margins. Furthermore, to be effective its gain may have to be determined by data, gathered off line, concerning the system's response. However, we cannot obtain such meaningful data due to the unpredictable variability in a processor's workload during program executions. Therefore we adopted a *variable-gain integrator*, whose gain is recomputed at the beginning of each control cycle as a part of the control loop, hence based only on measurements, in a way that extends the stability margins and provides the desirable tracking. In fact, simulation testing showed that this obviates the addition of a proportional element to the controller.

The controller has the following form,

$$u_k = u_{k-1} + A_k e_{k-1},$$

where u_k is the control variable set at the start of C_k, A_k is computed from measurements made during C_{k-1} and hence available at the start of C_k, $e_{k-1} = r - y_{k-1}$, and y_{k-1} is computed from measurements during C_{k-1}. When the gain A_k is independent of $k = 1, 2, \ldots$ we recognize this as an adder, a discrete-time equivalent of an integrator. The gain A_k is computed by the following formula,

$$A_k = A_{k-1} + \frac{1}{\frac{\partial y_{k-1}}{\partial u_{k-1}} + \eta_{k-1}} e_{k-1}. \tag{55}$$

The output y_{k-1} depends not only on u_{k-1} but possibly also on noise and other exogenous processes as well as past output like y_{k-2}, etc. These variables are not factored in the term $\frac{\partial y_{k-1}}{\partial u_{k-1}}$ which hence literally stands for the partial derivative.

The term $\frac{\partial y_{k-1}}{\partial u_{k-1}}$ has to be estimated in real time during C_{k-1} as a part of the control loop. However, in the computer application described above, and in other DEDS and SHS, we were unable to compute it due to the absence of analytical models for the plant. Therefore, in Eq. (55), we allow for an additive error, η_{k-1}, in its computation. Convergence results of the resulting tracking algorithm, derived in Wardi et al. (2016), account for the presence of such error terms. Simulation tests verify these results with substantial relative erros, which can be 30% or higher. In other words, the performance of the regulation technique is robust with respect to computational errors in the loop. Leveraging this robustness, we have estimated $\frac{\partial y_{k-1}}{\partial u_{k-1}}$ by IPA. But unlike unbiasedness and exact computation, which have been principal concerns in the use of IPA throughout much its development, we were primarily concerned with fast computations while allowing for substantial bias and computational errors.

Results of simulation experiments can be found in Wardi et al. (2016) and references therein. These include queueing networks, Petri nets, transportation models, and other DEDS. Of a particular interest is the case where the plant is a queueing system with biased IPA. For the original problem of interest, namely instruction-throughput regulation in computer processors, we used a detailed system-level simulation platform for computer architectures, called *Manifold* (Yalamancili, Riley, & Conte, 2016). IPA is biased, and we also induced further errors deliberately in order to simplify the computations of its derivative. Lately we implemented the regulation technique on Haswell, Intel's fourth-generation microarchitecture, Hammarlund et al. (2014), and tested it on industry-benchmark programs. In this implementation we actually adopted a simpler computation of $\frac{\partial y_{k-1}}{\partial u_{k-1}}$ than IPA can provide, one that is based on linear approximation. Various results can be found in Chen, Wardi, and Yalamanchili (2016).

To summarize, the technique described in this subsection extends the research area in IPA in two directions. First, it explores applications to systems' performance regulation rather than optimization. Second, it does not pursue the objectives of unbiased gradient estimates and their precise computations, but rather seeks simple control laws with fast computations in the loop.

4.3. Event-driven control and optimization

The emergence of DEDS in the 1980s brought to the forefront an alternative viewpoint to the traditional *time-driven* paradigm in which time is an independent variable and, as it evolves, so does the state of a dynamic system. The *event-driven* paradigm offers an alternative, complementary look at modelling, control, communication, and optimization (Cassandras, 2014; Miskowicz, 2015). The key idea is that a clock should not be assumed to dictate actions simply because a time step is taken; rather, an action should be triggered by an "event" specified as a well-defined condition on the system state or as a consequence of environmental uncertainties that result in random state transitions. Observing that such an event could actually be defined to be the occurrence of a "clock tick", it follows that this framework may in fact incorporate time-driven methods as well. On the other hand, defining the proper "events" requires more sophisticated techniques compared to simply reacting to time steps. In the development of DEDS, such events were seen as the natural means to drive the dynamics of a large class of systems including computer networks, manufacturing systems, and supply chains among many. By the early 1990s, however, it became evident that many interesting dynamic systems are in fact "hybrid" in nature, i.e., at least some of their state transitions are caused by (possibly controllable) events. This has been reinforced by technological advances through which sensing and actuating devices are embedded into systems allowing physical processes to interface with such devices which are inherently event-driven. More recently, the term *Cyber-Physical System* (CPS) has emerged to describe the hybrid structure of systems where some components operate as physical processes modeled through time-driven dynamics, while other components (mostly digital devices empowered by software) operate in event-driven mode.

Moreover, many systems of interest are now networked and spatially distributed. In such settings, especially when energy-constrained wireless devices are involved, frequent communication among system components can be inefficient, unnecessary, and sometimes infeasible. Thus, rather than imposing a rigid time-driven communication mechanism, it is reasonable to seek instead to define specific events which dictate when a particular node in a network needs to exchange information with one or more other nodes. When, in addition, the environment is stochastic, significant changes in the operation of a system are the result of random event occurrences, so that, once again, understanding the implications of such events and reacting to them is crucial. In distributed systems, event-driven mechanisms have the advantage of significantly reducing communication among networked components without affecting desired performance objectives. In multi-agent systems where the goal is for networked components to cooperatively maximize (or minimize) a given objective, it is shown in Zhong and Cassandras (2010) that an event-driven scheme can still achieve the optimization objective while drastically reducing communication (hence, prolong the lifetime of a wireless network), even when delays are present (as long as they are bounded). Event-driven approaches are also attractive in receding horizon control, where it is computationally inefficient to re-evaluate a control value over small time increments as opposed to event occurrences defining appropriate planning horizons for the controller. Finally, as already pointed out in Section 4.1, the use of event-driven optimization methods has the benefit of scaling with the size of the event-space and not the (generally much larger) state space of a system.

In Section 3, we discussed how IPA is used in the control and optimization of SHS based on the general-purpose IPA Calculus. However, even when a hybrid system is studied in a deterministic setting, IPA proves extremely useful in evaluating performance gradients *on line* that can be used for the purpose of optimizing the operation of complex multi-agent systems. These are commonly modeled as hybrid systems with time-driven dynamics describing the motion of the agents or the evolution of physical processes in a given environment, while event-driven behavior characterizes events that may occur randomly (e.g., an agent failure) or in accordance with control policies (e.g., an agent stopping to sense the environment or to change direction). As such, a multi-agent system can be studied in the context of the IPA Calculus with parameterized controllers aiming to meet certain specifications or to optimize a given performance metric. In some cases, the solution of a multi-agent dynamic optimization problem is reduced to a policy which is naturally parametric. Therefore, IPA may be used to evaluate on line performance gradients through which one can drive the system towards optimal (at least locally) points; recent examples of this approach may be found in Cassandras, Lin, and Ding (2013) and Zhou, Yu, Andersson, and Cassandras (2016).

5. Conclusions

This paper provides a narrative of the evolution of PA from an algorithm for a specific buffer allocation problem in a production line to a general framework for sensitivity analysis of stochastic hybrid dynamical systems. Central to PA is Infinitesimal Perturbation Analysis, a data-driven technique for computing realizations of stochastic gradients of performance metrics with respect to finite-dimensional system-parameters. Such gradients can be used in performance optimization, be it off line via simulation or on-line by observing data from a real system in operation.

One of the key features of IPA is the simplicity of its gradient-formulas in a great number of systems of interest. This, coupled with its reliance on observed data, suggests its use in control via real-time optimization. However, shortly after the inception of IPA in the DEDS setting, it was discovered that its gradient estimators often are statistically biased, thereby raising questions about the viability of the technique. Its subsequent development has had to wrestle with this issue; various alternative sample-based (hence data-based) algorithms were developed which, though unbiased, were more complicated than the basic IPA.

An approach that gained traction, initially based on continuous-flow queues, subsequently has been developed into a modelling framework of stochastic hybrid systems. In its setting, the IPA gradients are unbiased in a far-larger class of systems than in equivalent DEDS models, while preserving their simplicity of computations and reliance on observed data. This approach recently has been used in various applications.

Another approach of current interest concerns the sensitivity of quantiles of any random variable (Hong, 2009). It has a potential in many large-scale systems, and recently has been applied to performance sensitivities in financial engineering (Cao & Wan, 2017).

Future developments of IPA likely will focus on large-scale, complex systems and problems, with emphasis on control via real-time parameter optimization. In this, it naturally can be used in conjunction with big-data techniques which provide finite-dimensional parametrization of a system's behavior. Its scalability with systems' dimensions can be attained by leveraging the fact that it essentially is event-driven and hence suitable in event-driven control, thereby circumventing computational complexities associated with state-space explosions. Another potential research problem concerns decentralized real-time optimization in complex, multi-agent systems. In all of these problems the role of IPA can be summarized as providing gradient estimates for sensitivity analysis and on-line system optimization while managing computational complexity.

References

Abbas, K., Berkhout, J., & Heidergott, B. (2016). A critical account of perturbation analysis of markov chains. *Markov Processes and Related Fields, 22*, 227–265.
Baccelli, F., Cohen, G., Olsder, G., & Quadrat, J. (1992). *Synchronization and linearity*. Wiley.
Baxter, J., & Bartlett, P. L. (2001). Infinite-horizon policy-gradient estimation. *Journal of Artificial Intelligence Research, 15*, 319–350.
Baxter, J., Bartlett, P. L., & Weaver, L. (2001). Experiments with infinite-horizon, policy-gradient estimation. *Journal of Artificial Intelligence Research, 15*, 351–381.
Cao, X. R. (1985). Convergence of parameter sensitivity estimates in a stochastic experiment. *IEEE Transactions on Automatic Control, 30*(9), 845–853.
Cao, X. R. (1987). Sensitivity estimates based on one realization of a stochastic system. *Journal of Statistical Computation and Simulation, 27*, 211–232.
Cao, X. R. (1988). A sample performance function of closed Jackson queueing networks. *Operations Research, 36*(1), 128–136.
Cao, X. R. (1994). *Realization probabilities: The dynamics of queueing systems*. New York, NY: Springer-Verlag.
Cao, X. R. (2005). A basic formula for on-line policy-gradient algorithms. *IEEE Transactions on Automatic Control, 50*(5), 696–699.
Cao, X. R. (2007). *Stochastic learning and optimization – A sensitivity-based approach*. New York, NY: Springer.
Cao, X. R. (2009). Stochastic learning and optimization – A sensitivity-based approach. *(IFAC) Annual Review in Control, 33*, 11–24.

Cao, X. R. (2015). Optimization of average rewards of time nonhomogeneous markov chains. *IEEE Transactions on Automatic Control, 60*(7), 1841–1856.

Cao, X. R. (2017). Relative time and stochastic control with non-smooth features. *IEEE Transactions on Automatic Control, 62*(2), 837–852.

Cao, X. R., & Wan, X. (2017). Sensitivity analysis of nonlinear behavior with distorted probability. *Mathematical Finance, 27*, 115–150.

Cassandras, C. G. (1993). *Discrete event systems: Modeling and performance analysis*. Homewood, IL: Irwin Publ..

Cassandras, C. G. (2014). The event-driven paradigm for control, communication, and optimization. *Journal of Control and Decision, 1*(1), 3–17.

Cassandras, C. G., Abidi, M. V., & Towsley, D. (1990). Distributed routing with on-line marginal delay estimation. *IEEE Transactions on Communications, 38*(3), 348–359.

Cassandras, C. G., & Ho, Y. C. (1985). An event domain formalism for sample path perturbation analysis of discrete event dynamic systems. *IEEE Transactions on Automatic Control, 30*(12), 1217–1221.

Cassandras, C. G., & Lafortune, S. (2008). *Introduction to discrete event systems* (2nd ed.). Springer.

Cassandras, C. G., Lin, X., & Ding, X. C. (2013). An optimal control approach to the multi-agent persistent monitoring problem. *IEEE Transactions on Automatic Control, 58*(4), 947–961.

Cassandras, C. G., Wardi, Y., Melamed, B., Sun, G., & Panayiotou, C. G. (2002). Perturbation analysis for on-line control and optimization of stochastic fluid models. *IEEE Transactions on Automatic Control, 47*(8), 1234–1248.

Cassandras, C. G., Wardi, Y., Panayiotou, C. G., & Yao, C. (2010). Perturbation analysis and optimization of stochastic hybrid systems. *European Journal of Control, 16*(6), 642–664.

Caswell, H. (2013). Sensitivity analysis of discrete markov chains via matrix calculus. *Linear Algebra and its Applications, 438*(4), 1727–1745.

Chen, X., Wardi, Y., & Yalamanchili, S. (2016). Ipa in the loop: Control design for throughput regulation in computer processors. *13th international workshop on discrete event systems (wodes)*.

Fu, M. C., & Hu, J. Q. (1997). *Conditional Monte Carlo: Gradient estimation and optimization applications*. Boston, MA: Kluwer Academic Publishers.

Glasserman, P. (1991). *Gradient estimation via perturbation analysis*. Kluwer Academic Pub..

Gong, W. B., & Ho, Y. C. (1987). Smoothed perturbation analysis of discrete-event dynamic systems. *IEEE Transactions on Automatic Control, 32*, 858–866.

Hammarlund, P., Martinez, A., Bajwa, A., Hill, D., Hallnor, E., Jiang, H., et al. (2014). Haswell: The fourth-generation intel core processor. *IEEE Micro, 34*(2), 6–20.

Heidergott, B. (2000). Finite customer-oriented finite perturbation analysis for queuing networks. *Discrete Event Dynamic Systems: Theory and Applications, 10*(3), 201–232.

Hennessey, J., & Patterson, D. (2012). *Computer architecture: A quantitative approach*. Morgan Kaufmann.

Ho, Y. C., & Cao, X. R. (1991). *Perturbation analysis of discrete event dynamic systems*. Kluwer Academic Pub.

Ho, Y. C., & Cao, X. R. (1993). Perturbation analysis and optimization of queueing networks. *Journal of Optimization Theory and Applications, 40*(4), 559–582.

Ho, Y. C., Cao, X. R., & Cassandras, C. G. (1983). Infinitesimal and finite perturbation analysis for queueing networks. *Automatica, 19*, 439–445.

Ho, Y. C., & Cassandras, C. G. (1980). Computing costate variables for discrete event dynamic systems. In *19th IEEE conference on decision and control* (pp. 149–167).

Ho, Y. C., & Cassandras, C. G. (1983). A new approach to the analysis of discrete event dynamic systems. *Automatica, 19*, 149–167.

Ho, Y. C., Eyler, A., & Chien, D. T. (1979). A gradient technique for general buffer storage design in a serial production line. *International Journal of Production Research, 17*, 557–580.

Hong, L. (2009). Estimating quantile sensitivities. *Operations Research, 57*(1), 118–130.

Marbach, P., & Tsitsiklis, T. N. (2001). Simulation-based optimization of markov reward processes. *IEEE Transactions on Automatic Control, 46*(2), 191–209.

(2015). In M. Miskowicz (Ed.), *Event-based control and signal processing*. CRC Press/Taylor and Francis.

Proth, J., Sauer, N., Wardi, Y., & Xie, X. (1996). Marking optimization of stochastic timed event graphs by ipa. *Journal of Discrete Event Dynamic Systems, 6*(3), 221–239.

Ramadge, P. J., & Wonham, W. M. (1980). The control of discrete event systems. *Proceedings of the IEEE, 77*(1), 81–98.

Reiman, M., & Weiss, A. (1989). Sensitivity analysis for simulations via likelihood ratios. *Operations Research, 37*, 830–844.

Rubinstein, R., & Shapiro, A. (1993). *Sensitivity analysis and stochastic optimization by the score function method*. John Wiley & Sons, New York.

Rudin, W. (1974). *Real & complex analysis* (2nd ed.). McGraw-Hill Series in Higher Mathematics.

Schweitzer, P. (1968). Perturbation theory and finite markov chains. *Journal of Applied Probability, 5*, 410–413.

Sun, G., Cassandras, C. G., & Panayiotou, C. G. (2004). Perturbation analysis of multiclass stochastic fluid models. *Journal of Discrete Event Dynamic Systems, 14*(3), 267–307.

Suri, R., & Zazanis, M. (1988). Perturbation analysis gives strong consistent sensitivity estimates for the m/g/1 queue. *Management Science, 34*(1), 39–64.

Sutton, R., McAllester, D., Singh, S., & Mansour, Y. (2000). Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems, 13*, 1057–1063.

Wardi, Y., Cassandras, C., & Cao, X. R. (2017). Perturbation analysis: A framework for data-driven control and optimization of discrete event and hybrid systems. In *20th ifac world congress* (pp. 3083–3093).

Wardi, Y., Gong, W., Cassandras, C., & Kallmes, M. (1992). Smoothed perturbation analysis for a class of piecewise constant sample performance functions. *Journal of Discrete Event Dynamic Systems, 1*(4), 393–414.

Wardi, Y., Seatzu, C., Chen, X., & Yalamanchili, S. (2016). Performance regulation of event-driven dynamical systems using infinitesimal perturbation analysis. *Nonlinear Analysis: Hybrid Systems, 22*, 116–136.

Xia, L., Xia, Q.-S., & Cao, X. (2014). A tutorial on event-based optimization — A new optimization framework. *Discrete Event Dynamic Systems: Theory and Applications, 24*, 103–132.

Yalamancili, S., Riley, G., & Conte, T. (2016). *Manifold*. http://www.manifold.gatech.edu.

Zhang, P., & Cassandras, C. G. (2002). An improved forward algorithm for optimal control of a class of hybrid systems. *IEEE Transactions on Automatic Control, 47*(10), 1735–1739.

Zhong, M., & Cassandras, C. G. (2010). Asynchronous distributed optimization with event-driven communication. *IEEE Transactions on Automatic Control, 55*(12), 2735–2750.

Zhou, N., Yu, X., Andersson, S. B., & Cassandras, C. G. (2016). Optimal event-driven multi-agent persistent monitoring of a finite set of targets. In *Proc. of 55th IEEE conference on decision and control* (pp. 1814–1819).