POLITECNICO DI MILANO

SCUOLA DI INGEGNERIA DELL'INFORMAZIONE

Corso di Laurea Specialistica in Ingegneria
dell'Automazione

Approssimazione di sistemi ibridi stocastici attraverso l'approccio a scenario

RELATORE
Prof. Maria Prandini

CORRELATORE
Prof. Simone Garatti

TESI DI LAUREA DI
Riccardo Maria Vignali
Matr. N. 755646

Anno Accademico 2010/2011

Alla mia famiglia

Indice

Elenco delle figure

Elenco delle tabelle

Sommario

In questa tesi viene affrontato il problema dell'approssimazione di un sistema ibrido stocastico mediante un modello più semplice, ai fini della verifica di proprietà di raggiungibilità come quella di safety.

I sistemi ibridi stocastici sono sistemi dinamici caratterizzati da una componente dinamica continua ed una discreta che interagiscono tra di loro, e la cui evoluzione è affetta da incertezza descritta mediante leggi probabilistiche. L'analisi di raggiungibilità di un sistema ibrido stocastico consiste nella valutazione della probabilità che, per esempio, una certa variabile del sistema entri in un certo insieme desiderato e, nel frattempo, rimanga confinata all'interno di un insieme "safe". I metodi di analisi attualmente disponibili sono difficilmente applicabili a casi di sistemi complessi, con uno spazio di stato continuo di dimensione elevata. Diventa quindi fondamentale per poter affrontare applicazioni realistiche di analisi di sistemi di larga scala poter astrarre il sistema ed ottenerne un modello semplificato. Se il modello approssimante riproduce con una certa accuratezza l'evoluzione della variabile del sistema da cui dipende la proprietà di raggiungibilità, è infatti possibile effettuare la verifica sul modello invece che sul sistema, fornendo delle garanzie sulla qualità del risultato ottenuto.

L'idea di questo lavoro di tesi è quella di formulare il problema dell'approssimazione di un sistema ibrido stocastico mediante un modello semplificato come un problema di ottimizzazione "chance-constrained" in cui si cerca di minimizzare la distanza fra la variabile di interesse del sistema e quella corrispondente del modello, per tutte le istanze di incertezza stocastica eccetto un insieme di probabilità ϵ predefinita.

Il problema di ottimizzazione chance-constrained risultante è difficile da risolvere in modo esatto. Una soluzione approssimata può essere ottenuta mediante l'approccio a scenario, che consiste nell'estrarre un certo numero finito di istanze di incertezza e nell'affrontare il problema chance-constrained

come se gli “scenari” estratti fossero i soli casi possibili. Sotto opportune ipotesi, il numero di scenari da estrarre può essere scelto in modo da garantire che la soluzione randomizzata ottenuta soddisfi anche tutte le istanze di incertezza non viste, eccetto un insieme di probabilità al più uguale a ϵ. Le prestazioni dell'approccio proposto sono confrontate con quelle di un approccio alternativo presente in letteratura basato sulla funzione stocastica di simulazione.

La tesi è articolata in cinque capitoli. Nel primo capitolo viene fatta una panoramica generale sui sistemi ibridi e sul problema della riduzione di modello in relazione alla verifica di raggiungibilità. Nel secondo capitolo viene descritto l'approccio basato sulla funzione stocastica di simulazione proposto in letteratura. Nel terzo capitolo è presentato il nuovo metodo basato sull'approccio a scenario. Nel quarto capitolo vengono riportati i risultati ottenuti in un esempio numerico, con relativi commenti, grafici e confronti. Nell'ultimo capitolo si discutono vantaggi e limiti del metodo proposto e possibili estensioni di interesse.

Capitolo 1

Verifica di sistemi ibridi stocastici e modelli semplificati

1.1 I sistemi ibridi: modelli deterministici, non deterministici, stocastici

I sistemi ibridi sono sistemi dinamici caratterizzati dall'interazione tra una componente dinamica a tempo continuo (time-driven) e una componente a eventi discreti (event-driven). Esempio classico di fenomeno descrivibile tramite un sistema ibrido è il rimbalzo di una pallina sul suolo: tra un rimbalzo e l'altro il movimento della pallina può essere descritto attraverso le classiche leggi della meccanica, caratterizzanti la dinamica a tempo continuo. Il rimbalzo, invece, può essere naturalmente descritto come un evento discreto che si presenta quando la pallina impatta sul suolo. Le due componenti discrete e continue, sono strettamente interagenti: da un lato, i rimbalzi avvengono solo se le variabili a tempo continuo, posizione e velocità, assumono un determinato valore, dall'altro, i rimbalzi stessi possono provocare a loro volta un cambiamento del valore di queste variabili, con una diminuzione della velocità della pallina in caso di urto anelastico. Un altro esempio di sistema ibrido è l'automobile in cui i processi fisici a tempo continuo che determinano il moto sono coordinati da un equipaggiamento di bordo che opera ad eventi discreti. Si pensi ad esempio all'innesco di un dispositivo ABS o, più semplicemente, ad un cambio di marcia: eventi di questo genere causano delle variazioni immediate nel comportamento del sistema, modificando istantaneamente il valore di alcuni parametri (ad esempio il valore

dei rapporti di trasmissione nel caso di cambio di marcia) o la forma delle equazioni in gioco. In generale, ogniqualvolta sia presente un processo fisico monitorato o controllato da un apparato digitale, può essere utile far ricorso ad un sistema ibrido per modellizzare il sistema complessivo. I sistemi ibridi risultano adeguati per descrivere una vastissima gamma di fenomeni negli ambiti più disparati: matematica finanziaria, biologia, telecomunicazioni e reti di generazione e distribuzione dell'energia, solo per citarne alcuni (una panoramica di questi ambiti è data in [4] e [8]).

1.1.1 I sistemi ibridi non-deterministici

Quando si ha una completa conoscenza del sistema da descrivere e non esistono elementi di incertezza, si può fare ricorso in fase modellistica a sistemi ibridi deterministici, in cui l'evoluzione futura del sistema è caratterizzata univocamente date le condizioni iniziali e l'andamento della variabile d'ingresso. Il più delle volte è però utile, se non addirittura indispensabile, introdurre dell'incertezza nei modelli. Questo può essere fatto per diverse ragioni, come ad esempio la presenza di segnali incerti, la conoscenza incompleta dei parametri del sistema o la volontà di sottomodellizzarne alcune parti. Una volta introdotta l'incertezza, il modello diventa non-deterministico, e l'evoluzione del sistema non è più identificata univocamente.

Per poter meglio comprendere la classe di modelli che saranno utilizzati in questo lavoro, viene qui di seguito data la definizione formale di Automa Ibrido (tratta da [9]) e di cosa si intenda per una sua soluzione (anche detta "esecuzione"). Per semplicità non verranno precisate tutte le ipotesi sulle funzioni e sugli insiemi in gioco necessarie per evitare casi "patologici" come mancanza di soluzioni, fenomeni zenoniani, fenomeni di chattering, e altro ancora.

Definizione 1. Un *automa ibrido* H è una collezione di 8 elementi:

$$H = \{Q,\, X,\, f,\, Init,\, Dom,\, E,\, G,\, \mathcal{R}\}\ ,$$

dove:

- $Q = \{q_1, q_2, ...\}$ è lo spazio di stato discreto i cui elementi sono detti *modi*.
- $X = R^n$ è lo spazio di stato continuo.

- $f : Q \times X \to \mathbb{R}^n$ è un insieme di campi vettoriali su X :
  - Per ogni $q \in Q$, $f(q, \cdot)$ è il campo vettoriale dell'equazione differenziale ordinaria che governa l'evoluzione della componente continua dello stato x nel modo q.
- $Init \subseteq Q \times X$ è l'insieme degli stati iniziali.
- $Dom : Q \to 2^X$ è una mappa che assegna ad ogni $q \in Q$ un sottoinsieme aperto di $\mathbb{R}^n$ all'interno del quale può avvenire l'evoluzione continua.
- $E \subseteq Q \times Q$ è un insieme di transizioni:
  - Ogni $e \in E$ è nella forma $e = (q, q')$ e rappresenta una transizione da q a q'.
- $G : E \to 2^X$ è un insieme di guardie:
  - Per ogni $e = (q, q')$ ogni volta che la componente continua dello stato raggiunge $G(e)$ dall'interno di $Dom(q)$, la transizione e è abilitata.
- $\mathcal{R} : E \times X \to 2^X$ è una funzione polidroma di reset:
  - Per ogni $e = (q, q') \in E$ e $x \in Dom(q)$, $\mathcal{R}(e, x) \subseteq Dom(q')$ è l'insieme dei valori che lo stato continuo x può assumere dopo essere stato resettato dalla transizione e.

La definizione di esecuzione di un automa ibrido viene data in maniera non formale, tralasciando anche in questo caso i dettagli matematici.

L'esecuzione di un automa ibrido è una sequenza di evoluzioni continue, intervallate da transizioni discrete. A partire da uno stato iniziale $(q_0, x_0) \in Init$ lo stato continuo x evolve in accordo con la soluzione dell'equazione differenziale $\dot{x} = f(q_0, x)$, finché non abbandona l'insieme $Dom(q_0)$. Durante questa intera evoluzione lo stato discreto rimane costante. Se in un certo istante x raggiunge l'insieme $G(q_0, q')$, per un qualche valore di $q' \in Q$, allora la transizione discreta $e = (q_0,q')$ è abilitata e può scattare; se questa transizione avviene, lo stato ibrido istantaneamente viene re-inizializzato al valore (q', x'), dove x' è dato dalla mappa dei reset: $x' \in \mathcal{R}(q, q')$. A

questo punto l'evoluzione si ripete come appena visto, a partire dallo stato (q', x'). Questa descrizione, seppur breve e poco formale, mette bene in luce la presenza dei molteplici livelli di scelta che possono entrare in gioco nell'evoluzione del sistema, e, quindi, nella definizione della sua soluzione. Ad esempio:

1. È possibile scegliere tra più condizioni iniziali all'interno di $Init$.

2. Il valore dello stato continuo a seguito di un reset non è univocamente determinato, dal momento che $\mathcal{R}$ è una funzione polidroma.

3. In un certo istante possono essere abilitate più guardie: è il caso in cui $x \in G(q, \tilde{q}) \cap G(q, q')$. Di conseguenza il valore assunto dallo stato x a seguito della transizione non è univocamente determinato.

4. Potrebbe presentarsi la scelta tra proseguire nell'evoluzione continua o effettuare una transizione. Se $x \in Dom(q) \cap G(q, q')$, infatti, il sistema può sia continuare ad evolvere a tempo continuo nel modo q, sia prendere la transizione discreta che porta al modo q'.

La gamma di possibili scelte nell'evoluzione del modello risulta ancora più ampia se consideriamo anche ingressi manipolabili, discreti e continui, e ingressi non manipolabili, discreti e continui.

Grazie alla loro complessità, e alla loro componente di "scelta" appena descritta, i sistemi ibridi non-deterministici permettono quindi di modellizzare una grandissima varietà di fenomeni, e sono adeguati per analisi di tipo worst-case in cui tutte le esecuzioni sono ugualmente possibili e si valuta qual è il caso pessimo. In certi ambiti un'analisi di questo tipo non è sufficiente: nel controllo del traffico aereo, ad esempio, sapere che esiste la possibilità che si verifichi un incidente è poco utile e quello che bisogna invece valutare è la probabilità con cui si può verificare. Queste considerazioni mostrano come sia necessaria un'estensione del concetto di sistema ibrido non-deterministico, attraverso l'introduzione di una componente stocastica e l'attribuzione di "pesi" diversi alle diverse traiettorie del sistema.

1.1.2 I sistemi ibridi stocastici

I Sistemi Ibridi Stocastici - SIS, d'ora in avanti - rappresentano un'estensione della classe di sistemi ibridi vista finora in cui non solo si tiene conto di

fonti di incertezza che agiscono sul sistema, ma anche della probabilità con cui le diverse istanze di incertezza si verificano. L'interesse verso questi tipi di modelli ha portato alla definizione di un elevato numero di SIS, caratterizzati dalle differenti modalità con cui la componente stocastica influenza il sistema. Tale componente, infatti, può manifestarsi in ognuna delle parti in cui è consentita una "scelta" nei Sistemi Ibridi Non-deterministici. Ad esempio può essere presente un'evoluzione continua descritta tramite equazioni differenziali stocastiche, possono esistere transizioni spontanee che avvengono ad istanti di tempo casuali (la cui "frequenza" può anche dipendere dallo stato), o re-inizializzazioni casuali a seguito di una transizione discreta.

In analogia con quanto fatto precedentemente, viene ora introdotta la definizione di automa ibrido stocastico tratta da [9] (omettendo le ipotesi tecniche per semplicità) e spiegato in modo informale che cosa si intende per sua esecuzione.

Definizione 2. Un *automa ibrido stocastico* M è una collezione di 8 elementi:

$$M = \{Q,\, X,\, Dom,\, f,\, g,\, Init,\, \lambda,\, \mathcal{R}\}\,, \tag{1.1.1}$$

dove:

- Q è lo spazio di stato discreto.
- $X = \mathbb{R}^n$ è lo spazio di stato continuo.
- $Dom : Q \to 2^X$ è la mappa che assegna ad ogni modo $q \in Q$ il sottoinsieme aperto di $\mathbb{R}^n$ in cui può avvenire l'evoluzione continua.
- $f : Q \times X \to \mathbb{R}^n$ è una collezione di campi vettoriali. Per ogni $q \in Q$, $f(q, \cdot)$ è il termine di deriva dell'equazione differenziale stocastica che governa l'evoluzione della componente continua dello stato x nel modo q.
- $g : Q \times X \to \mathbb{R}^{n\times m}$ è una collezione di matrici. Per ogni $q \in Q$, $g(q, \cdot)$ rappresenta il termine di diffusione dell'equazione differenziale stocastica che governa l'evoluzione della componente continua dello stato e determina il peso del contributo alla dinamica del moto Browniano.
- $Init : \mathcal{B}(S) \to [0, 1]$ è una misura di probabilità su $(S, \mathcal{B}(S))$ dove $S = Q \times X$ è lo spazio di stato ibrido e $\mathcal{B}(S)$ è la σ-algebra di Borel su S.

- $\lambda : S \to \mathbb{R}_+$ è una funzione che descrive l'intensità con cui si verificano le transizioni spontanee.
- $\mathcal{R} : S \times \mathcal{B}(S) \to [0, 1]$ è una misura di probabilità su $(S, \mathcal{B}(S))$ condizionata che determina la ri-inizializzazione dello stato ibrido associato ad una transizione discreta.

L'esecuzione di un automa ibrido stocastico è simile a quella di un automa ibrido non deterministico, ma presenta qualche piccola sostanziale differenza. Ipotizzando, senza perdere di generalità, che l'istante iniziale sia $t = 0$, si comincia estraendo in modo casuale lo stato iniziale (q_0, x_0) dall'insieme S, in accordo con la misura di probabilità $Init$:

$$P\{(q_0, x_0) \in A\} = Init(A) \quad \forall A \in \mathcal{B}(S) \ .$$

Questa è la prima differenza: la possibilità di ottenere il valore dello stato iniziale da una ben precisa misura di probabilità, e non con una scelta arbitraria tra tutti i possibili valori in un insieme, come succedeva invece nell'automa ibrido non-deterministico. La seconda differenza sta nella forma delle equazioni differenziali che governano la dinamica a tempo continuo; si tratta infatti di equazioni differenziali stocastiche (SDE), e lo stato continuo evolverà quindi secondo una legge del tipo:

$$dx_t = f(q_0, x_t)dt + g(q_0, x_t)\, dw_t \ , \tag{1.1.2}$$

dove g, come anticipato, è il peso del contributo dato dal processo standard di Wiener m-dimensionale (moto Browniano) w_t. Durante l'evoluzione continua lo stato discreto rimane costante: $q_t = q_0$. Un suo cambiamento si ha solo in corrispondenza di una transizione; a tal proposito, emerge una terza differenza rispetto al caso non deterministico, ovvero la distinzione tra transizioni forzate e transizioni spontanee. Le transizioni forzate avvengono quando lo stato continuo x_t sta per abbandonare l'insieme $Dom(q_0)$, mentre le transizioni spontanee avvengono all'interno dell'insieme S, e sono tipicamente modellizzate come un processo di arrivi di Poisson con tasso $\lambda(q_0, x_0)$ dipendente dal valore assunto dallo stato. Viene dunque a mancare il concetto di transizione abilitata da una guardia G, presente invece nei modelli ibridi non deterministici. Infine si nota un'ultima differenza: la mappa di reset $\mathcal{R}$ restituisce valori casuali a seguito delle transizioni, estratti in accordo ad una certa misura di probabilità.

La classe dei modelli ibridi stocastici così descritta è molto ampia e variegata, e possiede una grande capacità descrittiva, per contro, è però di difficile analisi. Non solo, la complessità può essere ulteriormente aumentata con l'introduzione di ingressi manipolabili agenti sulla componente continua o discreta. Questo è il motivo che ha spinto ad analizzare i SIS per sottoclassi, considerando di volta in volta soltanto alcune delle componenti stocastiche di questi sistemi. Vengono qui di seguito riportati due esempi di sottoclassi che sono state ampiamente studiate (i Piecewise Deterministic Markov Processes e gli Switching Diffusion Processes), e, a seguire, la particolare sottoclasse di sistemi ibridi stocastici a cui si fa riferimento in questo lavoro: i Jump Linear Stochastic System (JLSS).

1. I Piecewise Deterministic Markov Processes (PDMP) sono caratterizzati da un'evoluzione a tempo continuo deterministica: le equazioni differenziali stocastiche sono sostituite da equazioni differenziali ordinarie. Rimangono invece le transizioni spontanee, le transizioni forzate e le re-inizializzazioni casuali per lo stato discreto e continuo a seguito di una transizione.

2. Gli Switching Diffusion Processes (SDP) sono caratterizzati da equazioni differenziali stocastiche, mancanza di transizioni forzate ($Dom(q) = X \ \forall q \in Q$) e stato $x(t)$ continuo rispetto al tempo (non ho fenomeni di "salto" dovuti ai reset poiché $\sum_{q' \in Q} \mathcal{R}(q, x, \{q', x\}) = 1 \ \forall (q, x) \in \bar{S}$)

Per queste due sottoclassi di sistemi sono già stati sviluppati opportuni metodi per l'analisi e il controllo.

Passiamo dunque alla sottoclasse di modelli a cui faremo esplicito riferimento in questo lavoro: i JLSS. In riferimento alla Definizione 2 un Jump Linear Stochastic System è un particolare automa ibrido stocastico in cui:

- $Q = \{q\}$: è presente un solo stato discreto.

- La mappa $Dom : q \to X$, associa all'unico stato discreto q l'intero spazio di stato continuo X.

- I termini di deriva e diffusione $f(q, \cdot)$ e $g(q, \cdot)$ sono lineari. La SDE (1.1.2) assume quindi la forma:

$$dx_t = Ax_t dt + Fx_t dw_t \ . \tag{1.1.3}$$

- La funzione λ è costante e rappresenta l'intensità con cui si verificano le transizioni spontanee, modellizzate come arrivi di un processo di Poisson p_t con passo λ.

- $\mathcal{R}$ è una misura di probabilità degenere perché concentrata in un punto che riproduce reset deterministici della forma:

$$x_{t_n} = (I + R) \lim_{t \to t_n} x_t \ , \tag{1.1.4}$$

  dove R è una matrice costante e con t_n si è indicato il generico istante di reset.

Tra due eventi generati dal processo di Poisson p_t l'evoluzione continua del sistema avviene in accordo all'equazione differenziale stocastica (1.1.3). All'arrivo del processo di Poisson il sistema effettua un salto, e il valore dello stato viene resettato secondo la (1.1.4). Un JLSS può essere descritto in modo compatto mediante l'equazione:

$$dx_t = Ax_t dt + Fx_t dw_t + Rx_t dp_t \ . \tag{1.1.5}$$

Questo modello di JLSS può essere esteso, introducendo N processi di Poisson indipendenti ciascuno con un proprio rate λ_i che determinano reset deterministici in accordo a matrici R_i diverse:

$$dx_t = Ax_t dt + Fx_t dw_t + \sum_{i=1}^{N} R_i x_t dp_t^i \ . \tag{1.1.6}$$

In questo caso la mappa di reset diventa probabilistica.

Prima di procedere con la trattazione del secondo principale argomento di questo capitolo, ovvero i problemi di verifica di proprietà, può essere opportuno raccogliere in una tabella le caratteristiche principali di un sistema ibrido stocastico, mostrando quali componenti vengano catturate da ciascuna delle tre sottoclassi analizzate fin qui.

Caratteristiche	SIS (1.1.1)	PDMP	SDP	JLSS 1.1.5
Equazioni differenziali stocastiche	✓		✓	✓
Reset stocastici	✓	✓		
Transizioni spontanee	✓	✓	✓	✓
Transizioni forzate	✓	✓		

Tabella 1.1: Panoramica di alcuni Sistemi Ibridi Stocastici

1.2 Problemi di verifica e relativi metodi

L'analisi e verifica di proprietà legate all'evoluzione temporale di un sistema ibrido è un problema difficile da affrontare, e di natura diversa a seconda che si coinvolga un modello ibrido non deterministico o un modello ibrido stocastico. Nel primo caso si parla di problemi di analisi "worst-case" in cui tutte le evoluzioni ammissibili sono trattate allo stesso modo, mentre nel secondo caso si parla di analisi "probabilistica" in cui si valuta la probabilità dell'insieme delle esecuzioni del sistema che soddisfano la proprietà di interesse. Un esempio tipico è quello della verifica di raggiungibilità di una regione dello spazio di stato: per un sistema non deterministico le uniche risposte che si possono ottenere a riguardo, sono un "sì, è raggiungibile" o un "no, non è raggiungibile". Non sono ammesse risposte "intermedie" e l'informazione ottenuta è di tipo qualitativo. Per un sistema stocastico, invece, una verifica di raggiungibilità assume connotati meno netti, più sfumati: ci si può infatti domandare quale è la probabilità che il sistema raggiunga la regione di interesse o quale è la distribuzione dell'istante in cui viene raggiunta, etc.

1.2.1 Problemi di verifica

I principali problemi di verifica per sistemi ibridi studiati in letteratura riguardano proprietà comuni ai sistemi dinamici a tempo continuo e a tempo discreto, come osservabilità e raggiungibilità:

- Verifica di raggiungibilità: nella sua forma più semplice consiste nel valutare se il sistema raggiunge un certo insieme in un certo intervallo di tempo a partire da un dato stato iniziale, oppure quali sono le condizioni iniziali tali che questa proprietà è soddisfatta. Proprietà

di raggiungibilità più complicate possono coinvolgere più insiemi che devono essere raggiunti in una certa successione, o di cui alcuni devono essere evitati ed altri raggiunti. Un caso particolare di verifica di raggiungibilità è la verifica di "safety", in cui si vuole verificare se lo stato di un dato sistema può raggiungere una regione "unsafe" dello spazio di stato.

- Verifica di controllabilità: concetto analogo alla raggiungibilità, che coinvolge però l'introduzione delle variabili di controllo. Si tratta di progettare la legge di controllo in modo da raggiungere una determinata regione dello spazio di stato.

- Verifica di osservabilità: consiste nella ricostruzione dello stato di un sistema ibrido, a partire dalla conoscenza degli ingressi e delle uscite. È un problema classico nell'ambito della fault detection, utile per il controllo di sistemi ibridi con stato non accessibile.

La verifica di una determinata proprietà o la sintesi di una legge di controllo possono essere molto complicate e onerose dal punto di vista computazionale, soprattutto quando si ha a che fare con sistemi di larga scala. Per far fronte a questi problemi, sono stati sviluppati tecniche e strumenti specifici, applicabili solo a determinate sottoclassi di SIS. Una panoramica di queste tecniche è data qui di seguito.

1.2.2 Metodi per la verifica

Il grande interesse per i SIS ha portato alla realizzazione di varie tecniche di analisi, riconducibili a due grandi famiglie principali:

1. Metodi di model checking

2. Metodi randomizzati

Il model checking è una tecnica nata in ambito informatico per lo studio della correttezza di un software. Nell'ambito della verifica di sistemi dinamici, consiste nel valutare se una certa proprietà è soddisfatta o meno attraverso l'esplorazione di tutte le possibili esecuzioni del sistema ottenute a partire da un suo modello. Le tecniche di model checking si prestano bene all'analisi di sistemi non stocastici con uno spazio di stato finito in cui tutte le possibili esecuzioni possono essere analizzate in modo esaustivo. Nel caso di sistemi

con spazio di stato continuo, si può ricorrere alla loro approssimazione con sistemi equivalenti per quanto riguarda la proprietà da verificare, ma con uno spazio di stato discreto. Questo è possibile in modo esatto solo in casi molto particolari. Alternativamente, si approssima l'evoluzione dell'insieme delle condizioni iniziali mediante insiemi facilmente rappresentabili quali poliedri o ellissoidi. L'elevata dimensionalità del modello può però rendere comunque intrattabile il problema da un punto di vista computazionale.

Negli ultimi anni sono state sviluppate tecniche di model checking per sistemi stocastici (catene di Markov a stati finiti), e gli algoritmi sono stati implementati con successo in software tools come PRISM o MRMC. Nel campo dei sistemi ibridi stocastici con uno spazio di stato continuo, il model checking sta muovendo i suoi primi passi e i risultati più importanti sono stati ottenuti per sistemi ibridi stocastici a tempo discreto (si veda a proposito [1]).

I metodi randomizzati possono essere utilizzati sia in fase di analisi, sia in fase di sintesi, e sembrano i migliori candidati al superamento delle difficoltà dovute ai problemi computazionali. Per questi motivi l'interesse verso questi metodi è cresciuto notevolmente nella comunità scientifica negli ultimi 15 anni. Per quanto riguarda la verifica, il metodo più semplice consiste nell'eseguire molteplici simulazioni del sistema e dai risultati osservati determinare una stima che la proprietà di interesse si verifichi (stima Monte-Carlo). Un altro campo di applicazione è l'analisi delle prestazioni per un sistema avente parametri incerti; in particolare la valutazione del livello di prestazione minimo garantito per le varie istanze dell'incertezza. Come verrà evidenziato in questo lavoro, tale problema può essere riformulato come problema chance-constrained e risolto con l'utilizzo di tecniche randomizzate.

L'intrinseca complessità dei sistemi ibridi stocastici, unita all'elevata dimensionalità dei modelli utilizzati nelle applicazioni (reti di telecomunicazioni, reti di distribuzione dell'energia, controllo del traffico aereo) rappresenta un grosso scoglio nell'affrontare problemi di analisi e, ancor più, di sintesi. L'utilizzo di tecniche di riduzione d'ordine di modelli può essere quindi necessario in certi casi.

1.3 La riduzione di modello

Con riduzione di modello si intende l'insieme di strategie e tecniche volte alla sintesi di un modello semplificato (ridotto) a partire da un modello dato. Il modello semplificato deve essere più semplice da analizzare del modello originale, ma al contempo deve essere in grado di riprodurne con una certa accuratezza il comportamento. Queste tecniche sono fondamentali nell'ambito di analisi e sintesi: infatti, molti dei problemi che risultano intrattabili per un modello complesso, possono diventare risolubili sul modello semplificato. Noto l'errore tra i due modelli, è possibile poi dare delle garanzie sul risultato ottenuto sul modello originale. La verifica di safety tramite model checking ne è un tipico esempio: se il modello da analizzare presenta uno spazio di stato continuo di dimensione elevata e si ricorre a tecniche di model checking sul sistema con spazio di stato discretizzato, allora ci si scontra con la "maledizione della dimensionalità". Se però si riesce ad ottenere un modello semplificato con un uno spazio di stato continuo di dimensione inferiore, il problema può diventare trattabile dal punto di vista computazionale.

Nello studio della riduzione dei modelli vanno affrontati due aspetti interlacciati:

- La *scelta* del modello semplificato.
- La *valutazione* della bontà del modello approssimato.

Quando si ha la necessità di ridurre un modello è naturale ricercare un criterio che guidi la scelta nell'eliminazione di una componente piuttosto che un'altra, ed il criterio di scelta deve essere legato all'utilizzo previsto per il modello ottenuto.

1.3.1 Qualità del modello approssimante ai fini delle verifiche di raggiungibilità

Si supponga di voler effettuare una verifica di raggiungibilità per un sistema ibrido stocastico S_1 e che la proprietà di raggiungibilità da analizzare sia esprimibile con riferimento a una variabile $y_{S_1,t}$ a valori in $Y = \mathbb{R}^p$ (per esempio $y_{S_1,t}$ può essere una combinazione lineare della componente continua dello stato x_t del sistema). Il sistema approssimante S_2, alimentato dagli stessi ingressi stocastici di S_1 e opportunamente inizializzato, dovrà quindi

cercare di riprodurre l'andamento di $y_{S_1,t}$ tramite un segnale $y_{S_2,t} \in Y$. L'accuratezza con cui $y_{S_2,t}$ riproduce $y_{S_1,t}$ può essere misurata attraverso una funzione distanza tra le realizzazioni dei due segnali $d : Y^{[0,T]} \times Y^{[0,T]} \to \mathbb{R}_+$, dove $[0,T]$ è l'intervallo temporale di riferimento. Questa funzione può essere definita in diversi modi, quali ad esempio:

$$d(y_{S_1}, y_{S_2}) = \sup_{t\in[0,T]} ||y_{S_1}(t) - y_{S_2}(t)||^2 \tag{1.3.1}$$

$$d(y_{S_1}, y_{S_2}) = \sup_{t\in[0,T]} \inf_{\tau\in[0,T]} ||y_{S_1}(\tau) - y_{S_2}(\tau)||^2 \, . \tag{1.3.2}$$

La (1.3.1) è la classica norma L_∞, mentre la (1.3.2) è nota come *distanza di Hausdorff*: intuitivamente essa quantifica la distanza tra le traiettorie di y_{S_1} e y_{S_2} senza tener conto della coordinata temporale. Questa distanza può essere utile nell'affrontare problemi di safety.

Fatte queste premesse si può dare la seguente definizione formale della qualità di S_2 come approssimante di S_1.

Definizione 3. Un sistema S_2 è detto h-approssimante di livello $1-\epsilon$ del sistema S_1 se vale la seguente proprietà:

$$P(d(y_{S_1}, y_{S_2}) \leq h) \geq 1 - \epsilon \, , \tag{1.3.3}$$

dove $\epsilon \in [0,1]$ rappresenta la probabilità di violazione ammissibile.

Se ϵ viene posto uguale a 0, si richiede che S_2 riproduca il comportamento di S_1 per ogni realizzazione degli ingressi stocastici, e alcune di queste realizzazioni potrebbero portare ad un valore di h molto elevato, rendendo poco significativo il bound. Al crescere di ϵ, h diminuisce, ma l'insieme delle realizzazioni per cui la distanza $d(y_{S_1}, y_{S_2})$ è inferiore ad h si riduce. Al limite per $\epsilon = 1$, la garanzia sulla distanza è data su un insieme di realizzazioni di probabilità 0. ϵ deve essere quindi fissato > 0, ma sufficientemente piccolo.

La riduzione di modello ai fini di verifiche di safety

Supponendo che si desideri valutare la probabilità che S_1 entri nell'insieme unsafe A in $[0,T]$, se S_2 è un h-approssimante di livello $1-\epsilon$ di S_1 allora si può ricondurre il problema a quello della stima della probabilità che S_1 entri nell'insieme A_h ottenuto "allargando" A di un valore pari ad h (si veda

la Figura 1.3.1):

$$A_h = \{y_{S_2} | \exists y_{S_1} \in A \; t.c. \; ||y_{S_2} - y_{S_1}||^2 \leq h\} \; .$$

Più precisamente vale il seguente risultato:

$$P(\exists t \in [0,T], \, t.c. \, y_{S_1}(t) \in A) \leq P(\exists t \in [0,T] \, t.c. \, y_{S_2}(t) \in A_h) + \epsilon \; , \quad (1.3.4)$$

che giustifica quantitativamente l'idea iniziale di effettuare la verifica di safety sul modello ridotto S_2, invece che sul sistema S_1, e di fornire risultati con certe garanzie. Intuitivamente, per le realizzazioni per cui è soddisfatta la condizione

$$d(y_{S_1}, y_{S_2}) \leq h \; , \quad (1.3.5)$$

se y_{S_2} non entra in A_h, allora la y_{S_1} corrispondente non entrerà in A. Per quanto riguarda le realizzazioni per le quali la (1.3.5) non è soddisfatta non si può dire nulla, ma comunque la loro probabilità è $\leq \epsilon$. Nuovamente, richiedere una probabilità di violazione $\epsilon = 0$ porta ad ottenere valori di h elevati che si traducono in una eccessiva estensione dell'insieme A_h. D'altro canto se $\epsilon \simeq 1$ il bound (1.3.4) diventa poco significativo dato che la quantità da stimare è una probabilità e quindi necessariamente ≤ 1.

Rimane il problema di come determinare h fissato $\epsilon \in (0,1)$. Questo problema viene affrontato nei prossimi capitoli: una prima soluzione, basata sulla definizione di funzione stocastica di simulazione, è descritta nel Capitolo 2, mentre un nuovo approccio basato su metodi randomizzati è descritto nel Capitolo 3.

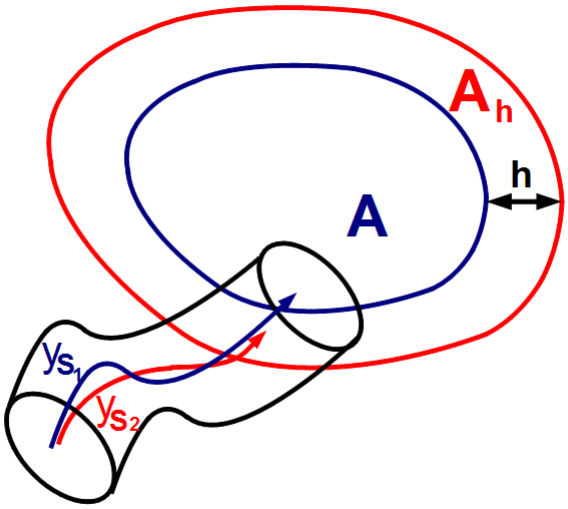


Figura 1.3.1: Visualizzazione grafica della valutazione di safety tramite il modello h-approssimante.

Capitolo 2

La funzione stocastica di simulazione nella riduzione di modelli

Il problema della valutazione della qualità d'approssimazione di un modello ibrido stocastico è il tema centrale dell'articolo [2], dove viene introdotto un approccio basato sul concetto di Funzione Stocastica di Simulazione (Stochastic Simulation Function - SSF d'ora in avanti), valido per una classe abbastanza ampia di sistemi ibridi stocastici che include i Jump Linear Stochastic Systems. Da un punto di vista implementativo il calcolo della SSF risulta trattabile per la sola sottoclasse dei JLSS. Per questo motivo illustreremo l'approccio con riferimento a tale classe di modelli.

2.1 La funzione stocastica di simulazione: definizione e proprietà

Siano dati due JLSS S_1 e S_2, descritti in modo compatto dalla seguente equazione:

$$S_i : \; dx_{i,t} = A_i x_{i,t} dt + F_i x_{i,t} dw_t + R_i x_{i,t} dp_t \quad i = 1, 2 \; , \tag{2.1.1}$$

dove $x_{i,t}$ è lo stato continuo e assume valori in X_i, w_t è un moto Browniano, p_t è un processo di Poisson. Si suppongono che $y_{S_i,t} = C_i x_{i,t}$, $i = 1, 2$ siano le variabili di interesse e a cui fare riferimento per valutare la bontà dell'approssimazione. I sistemi S_1 e S_2 sono alimentati dalla medesima coppia

di processi stocastici w_t e p_t e si vuole valutare qual è la distanza tra y_{S_1} e y_{S_2} garantita con probabilità $\geq 1-\epsilon$, dove $\epsilon \in (0,1)$ è la probabilità di violazione massima prefissata. La misura di probabilità adottata è quella indotta dai processi w_t e p_t, a condizione iniziale nota e fissata per S_1 e S_2.

Definendo i processi:

$$x_t := \begin{bmatrix} x_{1,t} \\ x_{2,t} \end{bmatrix}, \; y_t := y_{S_1,t} - y_{S_2,t} \; ,$$

è possibile costruire il seguente sistema JLSS esteso:

$$S : \begin{cases} dx_t = Ax_t dt + Fx_t dw_t + Rx_t dp_t \\ y_t = Cx_t \end{cases} \; , \tag{2.1.2}$$

dove:

$$A = \begin{bmatrix} A_1 & 0 \\ 0 & A_2 \end{bmatrix}, \; F = \begin{bmatrix} F_1 & 0 \\ 0 & F_2 \end{bmatrix}$$
$$R = \begin{bmatrix} R_1 & 0 \\ 0 & R_2 \end{bmatrix}, \; C = \begin{bmatrix} C_1 & -C_2 \end{bmatrix} \; .$$

Il vettore di stato x assume valori nello spazio di stato esteso:

$$X := X_1 \times X_2 \; .$$

La proprietà di h-approssimazione di livello $1-\epsilon$ di S_1 mediante S_2 secondo la Definizione 3 del Capitolo 1 può essere espressa nel modo seguente:

$$P\{ \sup_{t \in [0,T]} ||y_t||^2 < h \} \geq 1 - \epsilon \; ,$$

dove si è adottata la definizione di distanza data in (1.3.1). Vengono ora date di seguito le definizioni di martingala e supermartingala, necessarie alla comprensione della funzione di simulazione stocastica.

Definizione 4. Un processo stocastico X_t, con $t \geq 0$, è una *martingala* se:

1. $E[X_t] < \infty, \; \forall t$
2. $E[X_t | \{X_u\}_{u \leq s}] = X_s, \; \forall s \leq t$

La proprietà 1 afferma che il valore atteso del processo stocastico X_t deve mantenersi limitato. La proprietà 2 afferma che il valore atteso del processo all'istante t, condizionato ai valori del processo agli istanti di tempo

precedenti s è uguale ad X_s. Un esempio classico di questa proprietà della martingala è illustrabile con il gioco del lancio della moneta: in un gioco di testa o croce con moneta non truccata, poniamo che si vinca 1€ se esce testa, si perda 1€ se esce croce, e denotiamo con la sequenza di variabili aleatorie $X_0, X_1, X_2...$ il denaro posseduto dal giocatore dopo ciascun lancio. Senza altre informazioni, il valore atteso della vincita all'istante n sarà esattamente X_0, il denaro posseduto inizialmente dal giocatore; conoscendo però X_s, la vincita dopo s lanci, il valore atteso (condizionato da questa conoscenza) cambierà e diventerà X_s stesso.

Se la proprietà 2 della definizione precedente è verificata con il $\leq$, cioè

$$E\left[X_t | \{X_u\}_{u \leq s}\right] \leq X_s, \ \forall s \leq t \ ,$$

allora X_t è detta *supermartingala*. Rifacendosi all'esempio precedente, otteniamo una supermartingala se la moneta è truccata in modo che la probabilità di ottenere testa sia minore di 0.5: in questo caso il giocatore mediamente perderà e il valore atteso della vincita andrà diminuendo.

Definizione 5. Una funzione continua $\phi : X \to \mathbb{R}_+ \cup \{+\infty\}$ è una *Funzione Stocastica di Simulazione* (SSF) di S_1 da S_2 se valgono le seguenti due proprietà:

1. Per ogni $x \in X, \ \phi(x) \geq ||Cx||^2$
2. Il processo stocastico $\phi(x_t)$ è una *supermartingala* per ogni stato iniziale x_0.

Le condizioni che definiscono una SSF sono di facile interpretazione. La prima condizione garantisce che ϕ sia un upper bound alla distanza tra i segnali y_{S_1} e y_{S_2} di S_1 e S_2; la seconda condizione afferma che il valore atteso di ϕ, ovvero il valore atteso della distanza tra le y_{S_1} e y_{S_2}, è mediamente decrescente. S_2 è quindi in grado di seguire mediamente S_1. Appare dunque evidente come la SSF possa svolgere un ruolo importante nella valutazione della bontà di un modello stocastico approssimato. A riguardo, vale il seguente teorema (per la dimostrazione si rimanda a [2]).

Teorema 6. Sia dato un sistema nella forma descritta in (2.1.2) e sia data $\phi(\cdot)$, Funzione Stocastica di Simulazione. Vale la seguente relazione:

$$P\left\{\sup_{t \in [0,\infty)} ||y_t||^2 > \delta | x_0\right\} \leq \frac{\phi(x_0)}{\delta} \ ,$$

dove x_0 è la condizione iniziale del sistema (2.1.1) e $P(\cdot|x_0)$ indica la probabilità condizionata ad essa.

Si noti che la probabilità con cui $\sup_{t\in[0,\infty)} ||y_t||^2$ eccede il bound δ diminuisce al crescere di δ. La probabilità di violazione stimata mediante questo approccio è inversamente proporzionale a δ e direttamente proporzionale a $\phi(x_0)$. Si noti che $\sup_{t\in[0,T]} ||y_t||^2 \leq \sup_{t\in[0,\infty)} ||y_t||^2 \ \forall T < \infty$, per cui il Teorema 6 vale anche se si fa riferimento ad un orizzonte temporale finito.

Il Teorema 6 è riscrivibile in una forma equivalente in cui però il bound sulla norma di y_t dipende da $\phi(x_0)$.

Corollario 7. Siano dati due sistemi S_1 e S_2 descritti in (2.1.1) e sia data $\phi(\cdot)$ funzione stocastica di simulazione di S_1 da S_2. Allora si ha che:

$$P\left\{\sup_{t\in[0,\infty)} ||y_{S_1,t} - y_{S_2,t}||^2 > \frac{1}{\epsilon}\phi(x_0)|x_0\right\} \leq \epsilon \ , \tag{2.1.3}$$

dove $\epsilon \in [0,1]$ è un limite superiore alla probabilità di violazione.

Se si sceglie ad esempio $\epsilon = 0.1$ e $\epsilon = 0.05$, si ottengono rispettivamente:

$$P\left\{\sup_{t\in[0,\infty)} ||y_{S_1,t} - y_{S_2,t}||^2 > 10\phi(x_0)|x_0\right\} \leq 0.1$$
$$P\left\{\sup_{t\in[0,\infty)} ||y_{S_1,t} - y_{S_2,t}||^2 > 20\phi(x_0)|x_0\right\} \leq 0.05 \ .$$

Ovvero S_2 è in grado di simulare S_1, in modo tale che la probabilità che il supremo della norma al quadrato della differenza delle osservazioni superi $10\phi(x_0)$ e $20\phi(x_0)$ è al massimo pari a 0.1 o 0.05.

Per mostrare come questo risultato si ricolleghi alla Definizione (1.3.3) di h-approssimante data nel capitolo precedente è necessario eliminare il condizionamento rispetto a x_0 nell'equazione (2.1.3). Questo può essere fatto nel seguente modo:

$$P\left\{\sup_{t\in[0,T]} ||y_{S_1,t} - y_{S_2,t}||^2 > \frac{1}{\epsilon}\phi(x_0)\right\} \leq P\left\{\sup_{t\in[0,\infty)} ||y_{S_1,t} - y_{S_2,t}||^2 > \frac{1}{\epsilon}\phi(x_0)\right\} =$$
$$= \int_X P\left\{\sup_{t\in[0,\infty)} ||y_{S_1,t} - y_{S_2,t}||^2 > \frac{1}{\epsilon}\phi(x_0)|x_0\right\} \cdot f_{x_0}(x_0)dx_0 \leq$$
$$\leq \int_X \epsilon \cdot f_{x_0}(x_0)dx_0 = \epsilon \ .$$

con f_{x_0} densità di probabilità di x_0.

Il corollario mostra dunque che S_2 è un h-approssimante di livello $1-\epsilon$ per S_1, con $h = \frac{1}{\epsilon}\phi(x_0)$ dipendente dalla condizione iniziale x_0 del sistema unificato S e cioè dipendente dalle condizioni iniziali di S_1 e del suo approssimante S_2. In generale avere un bound h dipendente da x_0 risulta utile, dal momento che al variare dello stato iniziale possono variare di molto i valori di $\sup_{t\in[0,\infty)} ||y_t||^2$. Se comunque si desidera ottenere un bound costante, indipendente da x_0, è possibile ottenerlo calcolando l'estremo superiore di $h(x_0)$ al variare di x_0 sul supporto $\mathcal{D}$ della distribuzione f_{x_0}:

$$h = \sup_{x_0\in\mathcal{D}} h(x_0) \ .$$

Se però ad esempio il supporto $\mathcal{D}$ è illimitato e la funzione $h(x_0)$ non è limitata radialmente, si ottiene un valore h poco significativo, poiché infinito.

In conclusione la funzione stocastica di simulazione si dimostra uno strumento utile per la valutazione della bontà di approssimazione di un modello in accordo alla Definizione 3 data nel Capitolo 1. Nella seguente sottosezione verrà trattato il problema non banale di come calcolare effettivamente una SSF.

2.2 Il calcolo della funzione stocastica di simulazione

Il calcolo della funzione stocastica di simulazione per un sistema ibrido stocastico arbitrario può essere problematico. In [2] questo calcolo viene effettuato solo per il caso in cui sia S_1, sia S_2 appartengono alla classe dei JLSS su cui abbiamo appunto focalizzato la presentazione.

Si consideri il sistema JLSS S descritto in (2.1.2). Grazie alla dinamica lineare, il calcolo della SSF per questo tipo di modello può essere riscritto come una Linear Matrix Inequality (LMI): un problema in cui si effettua la ricerca di una matrice M che soddisfa alcuni vincoli di semidefinita positività di matrici linearmente dipendenti da M (per un'ampia trattazione delle LMI e del loro utilizzo nell'ambito dell'analisi e del controllo si faccia riferimento a [16]). Per non appesantire troppo la trattazione non verrà riportato l'intero procedimento di riscrittura del problema in forma LMI, ma solo il risultato finale, utile ai fini del presente lavoro.

Proposizione 8. Dato il JLSS S descritto da (2.1.2), una funzione quadratica $\phi(x) = x^T M x$, con M simmetrica semidefinita positiva e $x \in X$, è una Funzione Stocastica di Simulazione di S_1 da S_2 se e solo se:

$$\begin{cases} M - C^T C \succeq 0 \\ x^T Q x \preceq 0 \quad \forall x \in X \end{cases}, \tag{2.2.1}$$

dove Q è una matrice linearmente dipendente da M:

$$Q := M(A+R) + (A+R)M + F^T M F + \lambda R^T M R\,. \tag{2.2.2}$$

Il problema (2.2.1) è un classico problema LMI e può essere facilmente risolto utilizzando, per esempio, tool per la risoluzione di problemi convessi come CVX [15], o YALMIP [10].

2.3 Approssimazione di Jump Linear Stochastic System: un esempio numerico

In questa sezione verrà introdotto un esempio numerico, ripreso da [2], dove la SSF verrà valutata per due particolari JLSS e conseguentemente verranno confrontate le capacità di approssimazione effettive con quelle restituite dalla SSF in accordo alla teoria esposta nelle sezioni precedenti. Su questo esempio verrà testata anche la tecnica descritta nel terzo capitolo, in modo da poter poi confrontare i risultati.

Il sistema originale S_1 è un JLSS del sesto ordine, descritto da:

$$S_1 : \begin{cases} dx_{1,t} = A_1 x_{1,t} dt + F_1 x_{1,t} dw_t + R_1 x_{1,t} dp_t \\ y_{S_1,t} = C_1 x_{1,t} \end{cases}, \tag{2.3.1}$$

dove $y_{S_1,t}$ è la variabile di interesse. Le matrici che compaiono in (2.3.1)

sono date da:

$$A_1 = diag\left(\begin{bmatrix} -1 & -10 \\ 10 & -1 \end{bmatrix}, \begin{bmatrix} -2 & -20 \\ 20 & -1 \end{bmatrix}, \begin{bmatrix} -2 & 0 \\ 0 & -2.5 \end{bmatrix}\right)$$

$$F_1 = 0.5 \cdot \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}, \; R_1 = 0.7 \cdot I_6$$

$$C_1 = \begin{bmatrix} 0.84 & -1.03 & 1.07 & -0.88 & 0.5 & 0 \\ -0.6 & -1.35 & -0.26 & -0.27 & 0 & -0.5 \end{bmatrix} .$$

Si suppone che lo stato iniziale sia $x_{1,0} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T$, deterministicamente noto e sia nota deterministicamente anche la mappa $\rho(\cdot)$ che definisce lo stato x_{20} :

$$x_{2,0} = \rho(x_{1,0}) \; .$$

Verranno analizzati tre sistemi ridotti S_2:

1. Uno ottenuto eliminando le ultime due variabili di stato, e quindi riducendo l'ordine della dinamica continua di S_1.

2. Uno ottenuto eliminando la componente di moto Browniano e quindi rendendo deterministica l'evoluzione continua di S_1.

3. Uno ottenuto eliminando l'influenza del processo di Poisson, e quindi la componente discreta di S_1.

Per ciascuno di questi modelli verrà calcolata la funzione stocastica di simulazione tra il modello e il sistema S descritto in (2.3.1). Tale funzione verrà poi utilizzata per calcolare un bound garantito con probabilità $\geq 1-\epsilon$, dove $\epsilon = 0.25$, sulla differenza delle uscite, sfruttando il Corollario 7. I risultati ottenuti sono ora illustrati.

1) Dinamica continua ridotta. Il modello ridotto è ottenuto attraverso la rimozione delle ultime due variabili di stato e delle corrispondenti equazioni di stato, ponendo a zero il loro contributo all'evoluzione delle altre variabili

di stato.

$$S_2 : \begin{cases} dx_{2,t} = A_2 x_{2,t} dt + F_2 x_{2t} dw_t + R_2 x_{2,t} dp_t \\ y_{S_2,t} = C_2 x_{2,t} \end{cases} , \tag{2.3.2}$$

con

$$A_2 = diag\left(\begin{bmatrix} -1 & -10 \\ 10 & -1 \end{bmatrix}, \begin{bmatrix} -2 & -20 \\ 20 & -1 \end{bmatrix} \right), \; F_2 = 0.5 \cdot I_4$$
$$C_2 = \begin{bmatrix} 0.84 & -1.03 & 1.07 & -0.88 \\ -0.6 & -1.35 & -0.26 & -0.27 \end{bmatrix}, \; R_2 = 0.7 \cdot I_4$$
$$x_2 \in \mathbb{R}^4 .$$

La mappa $\rho(\cdot)$ è definita in questo modo:

$$\rho(x_{1,0}) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} x_{1,0} ,$$

di conseguenza $x_{2,0} = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T$. Attraverso la risoluzione del problema LMI (2.2.1) è possibile ottenere la matrice M simmetrica e semidefinita positiva della funzione stocastica di simulazione $\phi(x) = x^T M x$, e calcolare così un bound garantito al $(1-\epsilon) \cdot 100\%$ sulla distanza tra le realizzazioni dei due modelli sfruttando il Corollario 7. Il risultato ottenuto può essere quindi riportato in un grafico che contiene più realizzazioni del quadrato della distanza tra $y_{S_1,t}$ e $y_{S_2,t}$. Al più l'$\epsilon \cdot 100\%$ di esse potrà eccedere il bound. I bound riportati in questi grafici non saranno gli stessi di [2], dal momento che all'interno dell'articolo sono state riscontrate delle incoerenze tra quanto scritto e quanto rappresentato, ma saranno quelli ottenuti risolvendo nuovamente il problema.

Il procedimento per ottenere la SSF e l'upper bound ricercato può essere riassunto in due punti, validi per ognuno dei tre casi studiati:

1. Calcolo della matrice M della SSF attraverso la risoluzione del problema LMI (2.2.1). A tal proposito è importante notare che in [2] sono solamente forniti i vincoli che M deve soddisfare e non è indicata nessuna cifra di merito da minimizzare. Con lo scopo di ottenere il bound più stringente possibile (si ricordi il Corollario 7) si è scelta come cifra di merito da minimizzare proprio il valore di $\phi(x_0)$. Il problema LMI risolto per ognuno dei 3 casi è stato quindi:

$$\begin{aligned} &\min_{M} \phi(x_0) = x_0^T M x_0 \\ &\text{soggetta a:} \\ &M - C^T C \succeq 0 \\ &Q \preceq 0 \, . \end{aligned} \tag{2.3.3}$$

Per risolvere questo problema si è fatto ricorso al tool CVX per MATLAB.

2. Calcolo dell'upper bound UB_{SSF} (Upper Bound Stochastic Simulation Function) sulla distanza delle realizzazioni con probabilità massima di violazione ϵ sfruttando il Corollario 7:

$$UB_{SSF} = \frac{1}{\epsilon}\phi(x_0) = \frac{1}{\epsilon} x_0^T M x_0 \, .$$

Il valore ottenuto è stato poi rappresentato in un grafico, insieme a 100 realizzazioni di $||y_{S_1,t} - y_{S_2,t}||^2$. Il risultato complessivo è mostrato nella Figura 2.3.1 dove si è scelto $\epsilon = 0.25$.

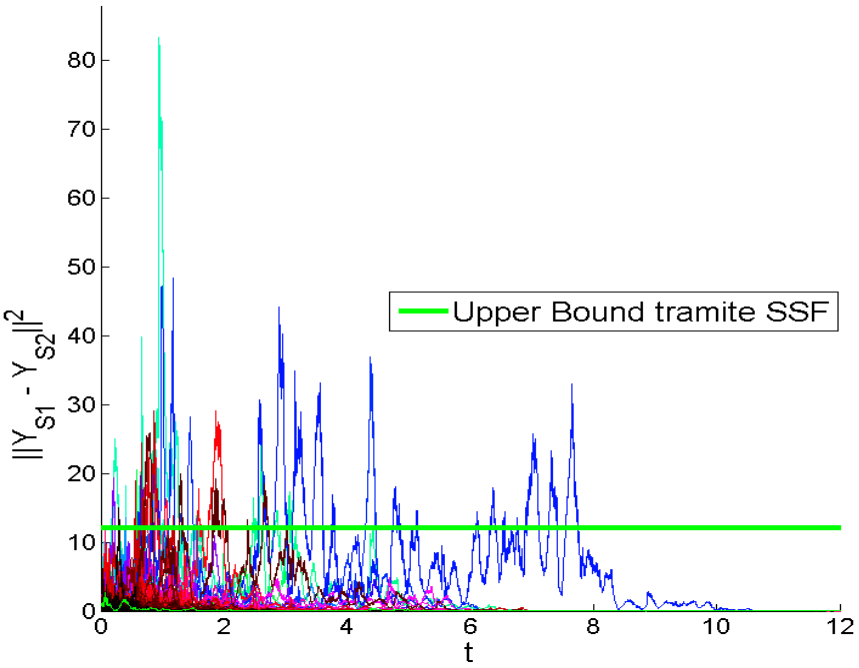


Figura 2.3.1: Caso 1: Dinamica continua ridotta – Sovrapposizione di 100 realizzazioni di $||y_{S_1,t} - y_{S_2,t}||^2$ e upper bound UB_{SSF} calcolato con $\epsilon = 0.25$.

Da questo grafico si nota come l'upper bound trovato ($UB_{SSF} = 12.04$) non sia molto stringente. In particolare esso dovrebbe limitare il 75% delle

realizzazioni, ma sulle 100 rappresentate solo 5 superano UB_{SSF}. Il livello di violazione empirico risulta ben più basso: $\epsilon_{empirico} = 0.05$. Questo risultato è ulteriormente convalidato da una prova su 3000 nuove realizzazioni indipendenti: in questo caso si ottengono 42 realizzazioni di $||y_{S_1,t} - y_{S_2,t}||^2$ che superano UB_{SSF}, e, di conseguenza, $\epsilon_{empirico} \approx 0.014$. In formule si ha che:

$$P\left\{\sup_{t\in[0,\infty)} ||y_{S_1,t} - y_{S_2,t}|| > UB_{SSF}\right\} \approx 0.014 \ll 0.25 \ .$$

Questo risultato è sicuramente accettabile, nel senso che UB_{SSF} effettivamente soddisfa il Corollario 7, ma il livello di violazione realmente ottenuto è fin troppo basso: $\epsilon_{empirico} \approx 0.014$ rispetto al $\epsilon = 0.25$ garantito dal Teorema 6. Il bound ottenuto con la SSF, dunque, è troppo conservativo.

2) Eliminazione del moto Browniano. Il modello ridotto è ottenuto attraverso la rimozione del moto Browniano. La matrice F_2 in (2.1.1) è dunque posta pari a 0.

$$S_2 : \begin{cases} dx_{2,t} = A_1 x_{2,t} dt + R_1 x_{2,t} dp_t \\ y_{S_2,t} = C_1 x_t \end{cases} \ . \tag{2.3.4}$$

La mappa $\rho(\cdot)$ è definita in questo modo:

$$\rho(x_{1,0}) = x_{1,0} \ ,$$

di conseguenza lo stato iniziale di S_2 è $x_{2,0} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T$. Procedendo come illustrato nel caso precedente, si ottiene il risultato illustrato in Figura 2.3.2.

Anche qui UB_{SSF} si dimostra troppo conservativo: solo 5 realizzazioni su 100 superano quota $UB_{SSF} = 21.8$. Su 100 realizzazioni la violazione effettivamente ottenuta è dunque $\epsilon_{empirico} = 0.05$, mentre se si esegue un test su 3000 nuove realizzazioni indipendenti si ottiene $\epsilon_{empirico} = 0.027$. Come nel caso precedente $\epsilon_{empirico} \ll \epsilon_{desiderato}$; la soluzione pertanto è troppo conservativa.

3) Eliminazione del processo di Poisson. Il modello ridotto è ottenuto attraverso la rimozione del processo di Poisson. La matrice R_2 in (2.1.1) è

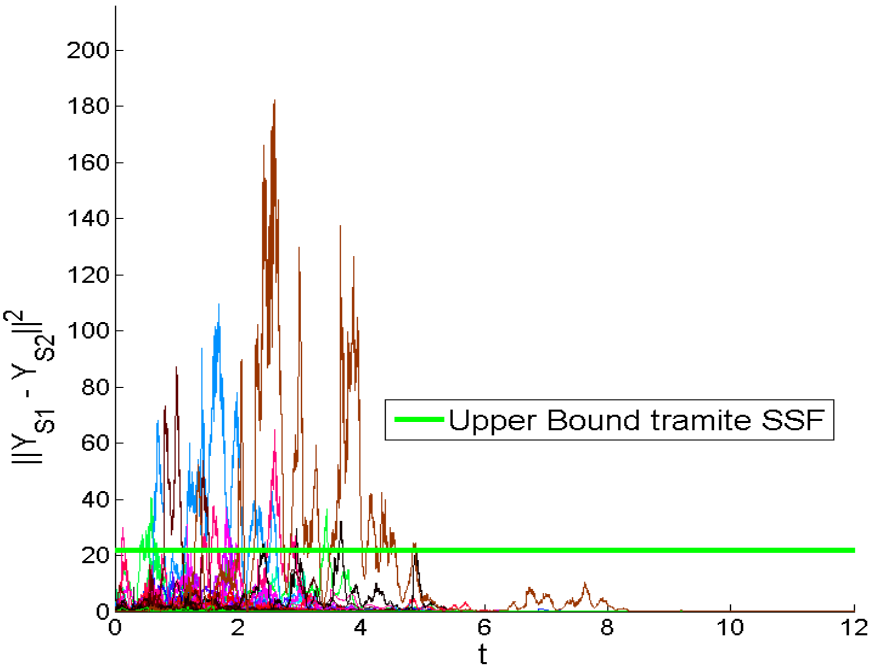


Figura 2.3.2: Caso 2: Eliminazione del moto Browniano – Sovrapposizione di 100 realizzazioni di $||y_{S_1,t} - y_{S_2,t}||^2$ e upper bound UB_{SSF} calcolato con $\epsilon = 0.25$.

dunque posta pari a 0.

$$S_2 : \begin{cases} dx_{2,t} = A_1 x_{2,t} dt + F_1 x_{2,t} dw_t \\ y_{S_2,t} = C_1 x_t \end{cases} . \tag{2.3.5}$$

La mappa $\rho(\cdot)$ è uguale a quella del caso precedente:

$$\rho(x_{1,0}) = x_{1,0} \, ,$$

lo stato iniziale di S_2 pertanto è $x_{2,0} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T$. Il risultato ottenuto è riportato in Figura 2.3.3:

Nuovamente si ottiene un bound troppo conservativo. Qui $\epsilon_{empirico} = 0.04$ sulle 100 realizzazioni in Figura (2.3.3). Analizzando 3000 nuove realizzazioni indipendenti e, mantenendo invariato lo stato iniziale, si ottiene un livello di violazione $\epsilon_{empirico} = 0.006$ decisamente troppo piccolo per le richieste fatte.

In tutti i casi, dunque, l'approccio basato sulla funzione stocastica di simulazione porta a risultati molto conservativi, nonostante si sia cercata la

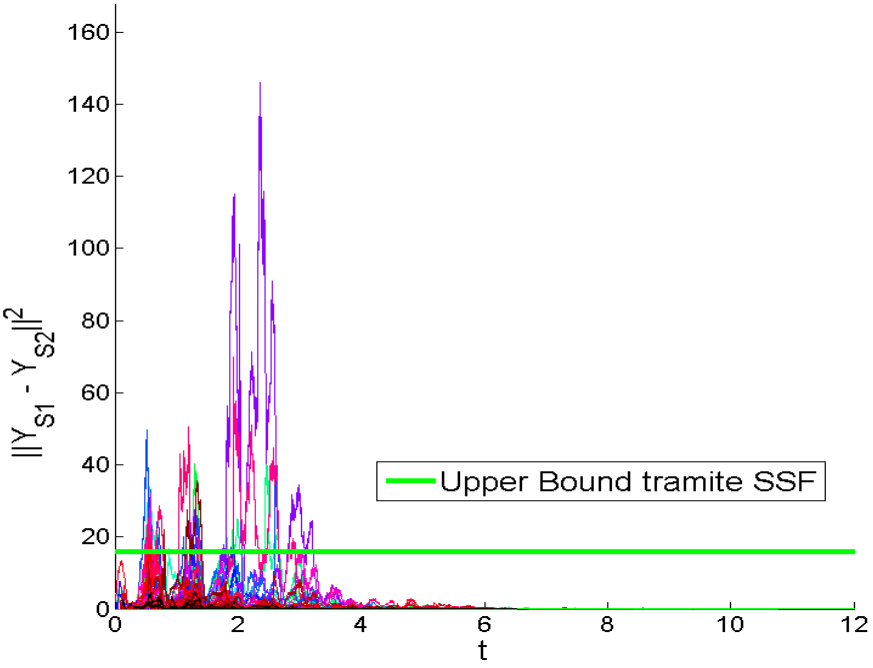


Figura 2.3.3: Caso 3: Eliminazione del processo Poisson – Sovrapposizione di 100 realizzazioni di $||y_{S_1,t} - y_{S_2,t}||^2$ e upper bound UB_{SSF} calcolato con $\epsilon = 0.25$.

matrice M minimizzando il valore del bound sulla discrepanza delle traiettorie (si veda (2.3.3)). Il motivo di ciò è che il Teorema 6 fornisce una condizione solo necessaria affinché $P\left\{\sup_{t\in[0,\infty)} ||y_t||^2 > \frac{1}{\epsilon}\phi(x_0)\right\} \leq \epsilon$. In altre parole, affinché $P\left\{\sup_{t\in[0,\infty)} ||y_t||^2 > \frac{1}{\epsilon}\phi(x_0)\right\} \leq \epsilon$ valga non è necessario che $\phi(x_0)$ sia una SSF. Per rimediare a tale inconveniente, un nuovo approccio basato sulla simulazione verrà illustrato nel capitolo successivo, mentre nel quarto capitolo verranno effettuati dei confronti tra i due metodi.

Capitolo 3

Un nuovo approccio basato sulla simulazione

3.1 Formulazione della riduzione di modello come problema di ottimizzazione chance-constrained

Dati un sistema ibrido stocastico S_1 e un modello approssimante S_2, il problema che affrontiamo in questo capitolo è quello di valutare in che misura S_2 approssima S_1 (Sezione 3.1.1) e di ottimizzare S_2 in modo da migliorare la qualità con cui esso approssima S_1 (Sezione 3.1.2). Faremo riferimento a tale scopo alla definizione di h-approssimante data nel Capitolo 1, che richiamiamo qui di seguito per completezza. In particolare ricordiamo che S_2 è detto h-approssimante di livello $1-\epsilon$ di S_1 se vale che:

$$P\left\{\sup_{t\in[0,T]} ||y_{S_1,t} - y_{S_2,t}||^2 \leq h\right\} \geq 1-\epsilon\ , \tag{3.1.1}$$

dove per semplicità abbiamo utilizzato il sup rispetto a t della norma della differenza delle realizzazioni.

L'idea che sta alla base della tecnica illustrata in questo capitolo è quella di utilizzare la (3.1.1) come un vincolo in probabilità e ottimizzare il valore di h, in modo da catturare il vero livello di somiglianza tra le traiettorie.

3.1.1 Valutazione della qualità di un modello approssimante

Il problema chance-constrained da risolvere è in questo caso il seguente:

$$\begin{aligned} &\min_{h} L[h(x_0)] \\ &\text{soggetto a } P\left\{ \sup_{t\in[0,T]} ||y_{S_1,t} - y_{S_2,t}||^2 \leq h(x_0) \right\} \geq 1-\epsilon \ , \end{aligned} \tag{3.1.2}$$

dove $L[h(x_0)]$ è un funzionale di costo definito sulla funzione $h(x_0)$ e $x_0 = \begin{bmatrix} x_{1,0} \\ x_{2,0} \end{bmatrix}$.

In (3.1.2) si è evidenziato che in generale quando le condizioni iniziali $x_{1,0}$ e $x_{2,0}$ di S_1 e S_2 sono soggette a variabilità, è bene che il bound h dipenda da $x_0 = \begin{bmatrix} x_{1,0} \\ x_{2,0} \end{bmatrix}$ in quanto inizializzazioni diverse portano a discrepanze tra le traiettorie e un bound unico può essere conservativo; si veda la discussione alla fine della Sezione 2.1 del Capitolo 2. In particolare si distinguono i due casi seguenti.

a.1) Condizione iniziale di S_1 deterministica
Condizione iniziale $x_{1,0}$ di S_1 fissata e condizione iniziale di S_2 $x_{2,0} = \rho(x_{1,0})$, anch'essa fissata (si ricorda che la mappa $\rho(\cdot)$ che ricava l'inizializzazione di S_2 dall'inizializzazione di S_1 fa parte dei dati della descrizione di S_2 ed è nota – si veda la Sezione 2.3 del Capitolo 2). Il bound h risulta quindi costante, condizionato dal particolare $x_0 = \begin{bmatrix} x_{1,0}^T & \rho(x_{1,0})^T \end{bmatrix}^T$ fissato. Il problema (3.1.2) assume di conseguenza la forma:

$$\begin{aligned} &\min_{h} h \\ &\text{soggetto a } P\left\{ \sup_{t\in[0,T]} ||y_{S_1,t} - y_{S_2,t}||^2 \leq h \,|\, x_0 \right\} \geq 1-\epsilon \ . \end{aligned} \tag{3.1.3}$$

Come già illustrato nella Sezione 2.1 del Capitolo 2 è possibile eliminare da (3.1.3) il condizionamento da x_0 integrando sul supporto della distribuzione di probabilità della condizione iniziale. Tale calcolo è in questo caso banale, dal momento che x_0 è deterministico e ha quindi distribuzione di probabilità

interamente concentrata in x_0, con valore 1. In definitiva in si ottiene:

$$\begin{aligned} &\min_{h} h \\ &\text{soggetto a } P\left\{ \sup_{t\in[0,T]} ||y_{S_1,t} - y_{S_2,t}||^2 \leq h \right\} \geq 1-\epsilon \ , \end{aligned} \tag{3.1.4}$$

dove in questo caso P è la probabilità sottostante la realizzazione degli ingressi stocastici w_t e p_t. Infatti nonostante la condizione iniziale dei due sistemi sia fissata, le variabili $y_{S_1,t}$ e $y_{S_2,t}$ (e di conseguenza anche $\sup_{t\in[0,T]} ||y_{S_1,t} - y_{S_2,t}||^2$) assumono valori casuali a causa dell'influenza dei processi stocastici w_t e p_t. La soluzione ottima h^* di questo problema è il quantile di ordine $1-\epsilon$ della distribuzione di probabilità della variabile aleatoria $\sup_{t\in[0,T]} ||y_{S_1,t} - y_{S_2,t}||^2$.

a.2) Condizione iniziale di S_1 stocastica

La condizione iniziale $x_{1,0}$ è una variabile casuale e di conseguenza lo è anche $x_{2,0} = \rho(x_{1,0})$. Il vincolo in probabilità di (3.1.2) avrà dunque forma:

$$P\left\{ \sup_{t\in[0,T]} ||y_{S_1,t} - y_{S_2,t}||^2 \leq h(x_{1,0},\theta) \right\} \geq 1-\epsilon \ ,$$

dove P si riferisce sia agli ingressi stocastici w_t e p_t, sia alla condizione iniziale x_0. Inoltre, affinché il problema possa essere risolto mediante metodi numerici, si parametrizzata $h(x_{1,0})$ mediante θ. Per quello che riguarda il funzionale di costo $L[h(x_0)]$, essendo $x_{1,0}$ casuale e per ottenere un risultato globalmente buono, si minimizza la media di $h(x_{1,0},\theta)$, rispetto ai possibili valori di $x_{1,0}$. In definitiva si ottiene quindi:

$$\begin{aligned} &\min_{\theta} E_{x_{1,0}}\left[h(x_{1,0},\theta)\right] \\ &\text{soggetto a } P\left\{ \sup_{t\in[0,T]} ||y_{S_1,t} - y_{S_2,t}||^2 \leq h(x_{1,0},\theta) \right\} \geq 1-\epsilon \ . \end{aligned} \tag{3.1.5}$$

3.1.2 Progetto del modello approssimante

Riscrivendo il problema di valutazione del livello di approssimazione come un problema di ottimizzazione, ci si accorge come lo stesso possa essere utilizzato anche per ottimizzare il modello approssimante al fine di migliorare la capacità di approssimazione (ovvero fare *design* del modello approssimante). Lo scopo di questo lavoro non è quello di fornire una completa

trattazione esaustiva rispetto a tutti i possibili gradi di libertà nella scelta del modello approssimante e ci si limiterà all'ottimizzazione della mappa $\rho(\cdot)$ tra le condizioni iniziali $x_{1,0}$ e $x_{2,0}$. Ancora una volta si distinguono due casi.

b.1) Condizione iniziale di S_1 deterministica
Condizione iniziale $x_{1,0}$ del sistema S_1 fissata. In questo caso ottimizzare la mappa $\rho(\cdot)$ equivale ad ottimizzare $x_{2,0}$. Il problema perciò diventa:

$$\begin{aligned} &\min_{x_{2,0},h} h \\ &\text{soggetto a } P\left\{\sup_{t\in[0,T]} ||y_{S_1,t} - y_{S_2,t}(x_{2,0})||^2 \leq h\right\} \geq 1-\epsilon\ , \end{aligned} \tag{3.1.6}$$

dove si è messo in evidenza il fatto che $y_{S_2,t}$ dipende dalla variabile di ottimizzazione $x_{2,0}$. Si noti che, essendo $x_{1,0}$ fissato, il bound $h(x_0)$ risulta costante, esattamente come nel caso a.1 illustrato precedentemente.

a.2) Condizione iniziale di S_1 stocastica
La condizione iniziale $x_{1,0}$ è stocastica e si vuole ottimizzare a tutti gli effetti la mappa $\rho(\cdot)$ qui supposta essere parametrica in λ, $\rho(\cdot) = \rho(\cdot, \lambda)$. A tal fine si ricerca la parametrizzazione λ che ottimizza le prestazioni come valutate nel caso a.2. Il problema risultante è:

$$\begin{aligned} &\min_{\theta,\lambda} E\left[h(x_{1,0},\theta)\right] \\ &\text{soggetto a } P\left\{\sup_{t\in[0,T]} ||y_{S_1,t} - y_{S_2,t}(\rho(x_{1,0},\lambda)||^2 \leq h(x_{1,0},\theta)\right\} \geq 1-\epsilon\ . \end{aligned}$$

Nuovamente, come nel caso a.2, la probabilità P è da intendersi rispetto ai valori dei processi stocastici e della condizione iniziale $x_{1,0}$.

3.2 L'approccio a scenario

Tutti i problemi introdotti nella precedente sezione sono del tipo:

$$\begin{aligned} &\min_{x} J(x) \\ &\text{soggetto a } P_\delta\left\{f(x,\delta) \leq 0\right\} \geq 1-\epsilon\ , \end{aligned} \tag{3.2.1}$$

dove x è la variabile di ottimizzazione mentre $\delta \in \Delta$ è invece il parametro stocastico su cui è definita la probabilità P_δ. Le variabili $\delta \in \Delta$, $x \in \mathbb{R}^d$ e

$f(x, \delta)$ assumono interpretazioni diverse a seconda del problema. In particolare δ corrisponde alle sole realizzazioni dei disturbi stocastici w_t e p_t per i problemi di tipo 1, e alle realizzazioni dei disturbi stocastici con aggiunta della condizione iniziale per i problemi di tipo 2, mentre:

- $x = h$ e $f(x, \delta) = \sup_{t \in [0,T]} ||y_{S_1,t} - y_{S_2,t}||^2 - h$ in a.1
- $x = \theta$ e $f(x, \delta) = \sup_{t \in [0,T]} ||y_{S_1,t} - y_{S_2,t}||^2 - h(x_0, \theta)$ in a.2
- $x = (x_{2,0}, h)$ e $f(x, \delta) = \sup_{t \in [0,T]} ||y_{S_1,t} - y_{S_2,t}(x_{2,0})||^2 - h$ in b.1
- $x = (\theta, \lambda)$ e $f(x, \delta) = \sup_{t \in [0,T]} ||y_{S_1,t} - y_{S_2,t}(x_{2,0}(\lambda))||^2 - h(x_0, \theta)$ in b.2.

I problemi di tipo (3.2.1) sono chiamati chance-constrained in quanto il vincolo $f(x, \delta) \leq 0$ non è richiesto essere soddisfatto sempre, ma con probabilità $1 - \epsilon$. Tipicamente, tali problemi sono di difficile risoluzione per due ragioni:

1. La distribuzione di probabilità di $f(x, \delta)$ può essere molto complicata, dal momento che è il risultato della propagazione della distribuzione di probabilità di δ attraverso sistemi generalmente non lineari.

2. Anche se $f(x, \delta) \leq 0$ è convesso in x $\forall \delta$, l'insieme delle x che soddisfa il vincolo $P_\delta \{f(x, \delta) \leq 0\} \geq 1 - \epsilon$ è in generale *non* convesso; si veda [3].

L'approccio a scenario (introdotto in [7]) è una recente metodologia che permette di trovare a basso costo computazionale una soluzione approssimata per problemi chance-constrained. Intuitivamente questa tecnica si basa sull'estrazione di N realizzazioni dell'incertezza δ e sulla risoluzione del problema campionato ottenuto considerando solo gli N vincoli corrispondenti alle estrazioni fatte; dagli N vincoli ne vengono poi eliminati una porzione $\lfloor \alpha N \rfloor$ con $0 \leq \alpha < \epsilon$, dove ϵ è la probabilità di violazione consentita nel problema chance-constrained originale, attraverso un algoritmo stabilito. La soluzione del problema con $N - \lfloor \alpha N \rfloor$ offre delle garanzie sul problema chance-constrained se N è scelto opportunamente. Viene di seguito spiegato nel dettaglio l'algoritmo dell'approccio a scenario.

Algoritmo a scenario

1. Si estraggano N istanze dell'incertezza $\delta^{(1)}, \delta^{(2)}, \ldots, \delta^{(N)}$ e si risolva il problema campionato:

$$\begin{aligned} &\min_x J(x) \\ &\text{soggetto a } f(x, \delta^{(i)}) \leq 0 \; \forall i = 1, \ldots, N \end{aligned} \tag{3.2.2}$$

   Si salvi la soluzione x^* trovata.

2. Si conti il numero di vincoli violati dalla soluzione x^*, ovvero il numero di volte in cui:

$$f(x^*, \delta^{(i)}) > 0 \; .$$

   Siano questi indici $j_1, \ldots, j_p$ (se $\{j_1, j_2, \ldots, j_p\} = \emptyset$, si prenda $p = 0$). Se $p = \lfloor \alpha N \rfloor$, con α parametro a scelta utente ($0 \leq \alpha < \epsilon$), allora si interrompa l'algoritmo e si restituisca la soluzione x^* salvata.

3. Si cerchino i vincoli attivi per la soluzione x^* salvata, ovvero gli indici i tali che $f(x^*, \delta^{(i)}) = 0 \quad i = 1, \ldots, N$. Siano questi indici $i_1, \ldots, i_m$.

4. Si calcoli

$$\mathcal{A}[i_1, \ldots, i_m] = i_s \; s \in \{1, \ldots, m\} \; ,$$

   dove $\mathcal{A}$ è l'algoritmo di eliminazione stabilito. Esso restituisce un particolare indice i_s tra gli m indici dei vincoli attivi in accordo a una determinata logica (due particolari tipi di algoritmi di eliminazione sono illustrati in seguito).

5. Si risolva il problema

$$\begin{aligned} &\min_x J(x) \\ &\text{soggetto a } f(x, \delta^{(i)}) \leq 0 \; i = \{1, \ldots, N\} \backslash \{i_s, j_1, \ldots, j_p\} \end{aligned}$$

   Se il valore della cifra di merito ottenuto è migliore di quello precedentemente salvato, si elimini la soluzione salvata e si aggiorni con l'ultima calcolata.

6. Si ritorni al punto 2.

L'algoritmo illustrato, se termina, restituisce una soluzione $x^*_{N,\lfloor \alpha N \rfloor}$ che non soddisfa il vincolo $f(x, \delta^{(i)}) \leq 0$ $\lfloor \alpha N \rfloor$ volte sulle N totali. L'algoritmo termina sicuramente se la rimozione di un vincolo attivo comporta un miglioramento della cifra di merito. Questa condizione è verificata quasi sempre, ad eccezione di casi patologici come l'estrazione di N realizzazioni dell'incertezza coincidenti $\delta^{(1)} = \cdots = \delta^{(N)}$.

I vincoli non soddisfatti sono rimossi uno alla volta in accordo ad un algoritmo $\mathcal{A}$ di eliminazione stabilito, come spiegato al punto 4. In questa trattazione si è fatto uso di due algoritmi: un algoritmo *greedy* e un algoritmo di eliminazione casuale.

- Algoritmo *greedy*. Si elimina tra gli m vincoli attivi quello che, una volta rimosso, genera il miglioramento massimo della cifra di merito:

  - *for* $k = 1, 2, \ldots, m$

    Si risolva il problema

    $$\begin{aligned} &\min_x J(x) \\ &\text{soggetto a } f(x, \delta^{(i)}) \leq 0 \; i \in \{1, \ldots, N\} \backslash \{i_k, j_1, \ldots, j_p\} \; , \end{aligned}$$

    dove $\{j_1, \ldots, j_p\}$ è l'insieme degli indici dei vincoli già violati, mentre i_k è l'indice del vincolo attivo considerato. Se il valore della cifra di merito ottenuto è migliore di quello precedentemente salvato, si elimini la soluzione salvata e si aggiorni con l'ultima calcolata.

  Per ogni eliminazione è quindi necessario risolvere m problemi di ottimizzazione, in modo da poter valutare quali di essi ha soluzione migliore.

- Algoritmo di eliminazione casuale. Si estrae casualmente un vincolo dagli m attivi e lo si elimina:

  - $\mathcal{A}[i_1, \ldots, i_m] = i_s$ con $s = \lfloor rand \cdot m \rfloor + 1$, dove $rand$ estrae un numero in $[0, 1]$ con probabilità uniforme.

L'idea di ottenere una soluzione del problema chance-constrained (3.2.1) estraendo N realizzazioni dell'incertezza, considerando solo i vincoli corri-

spondenti e eliminandone una determinata porzione $\lfloor \alpha N \rfloor$, potrebbe sembrare poco rigorosa, ma in realtà è fondata su solide basi matematiche. In particolare il seguente teorema garantisce un risultato sulla feasibility della soluzione $x^*_{N,\lfloor \alpha N \rfloor}$ ottenuta con l'algoritmo sopra illustrato per il problema originale chance-constrained (3.2.1). Prima di enunciare il teorema è necessario introdurre le seguenti due ipotesi.

1. Ogni problema di ottimizzazione ristretto a un sottoinsieme finito F di vincoli su Δ:

$$\begin{aligned} &\min_x J(x) \\ &\text{soggetto a } f(x,\delta) \leq 0 \ \delta \in F \subseteq \Delta \end{aligned}$$

   ammette una e una sola soluzione, feasible per il problema originale (3.2.1).

2. $J(x)$ è una funzione convessa di x. Inoltre, $\forall \delta \in \Delta$, $f(x,\delta)$ è convessa nella variabile x.

L'ipotesi 1 garantisce che il problema campionato (3.2.2) abbia sempre una e una sola soluzione feasible per il problema originale (3.2.1), a prescindere dal numero di vincoli che vengono poi eliminati. Tale ipotesi è verificata nella grande maggioranza dei casi di interesse. L'ipotesi 2 fa sì che i problemi di ottimizzazione campionati descritti nell'algoritmo siano convessi. Problemi di questo tipo possono essere agevolmente risolti attraverso l'uso di tools appositi come i già citati CVX [15] e YALMIP [10].

Viene ora enunciato il teorema su cui si fonda l'approccio a scenario per la risoluzione dei problemi chance-constrained, per la dimostrazione si rimanda a [13].

Teorema 9. Sia $\beta \in (0,1)$ un parametro di confidenza, $0 \leq \alpha < \epsilon$ un parametro a scelta dell'utente e d il numero di variabili di ottimizzazione. Se N è tale che:

$$\binom{\lfloor \alpha N \rfloor + d - 1}{\lfloor \alpha N \rfloor} \sum_{i=0}^{\lfloor \alpha N \rfloor + d - 1} \binom{N}{i} \epsilon^i (1-\epsilon)^{N-i} \leq \beta \tag{3.2.3}$$

allora, con confidenza non minore di $1-\beta$, la soluzione $x^*_{N,\lfloor \alpha N \rfloor}$ restituita dall'algoritmo a scenario è tale che:

$$P\left\{\delta : f(x^*_{N,\lfloor \alpha N \rfloor}, \delta) > 0\right\} \leq \epsilon \, .$$

Ovvero $x^*_{N,\lfloor \alpha N \rfloor}$ è feasible per il problema chance-constrained originale (3.2.1) con confidenza elevata $1 - \beta$.

Il teorema esprime un risultato sulla feasibility di $x^*_{N,\lfloor \alpha N \rfloor}$ per il problema originale chance-constrained: in particolare, essendo la soluzione $x^*_{N,\lfloor \alpha N \rfloor}$ ottenuta con l'approccio a scenario dipendente dalle N estrazioni casuali della variabile δ non è possibile affermare che la condizione (3.2.3) valga sempre, infatti potrebbe succedere che le N estrazioni fatte non siano abbastanza rappresentative. Questo è tuttavia un caso che può essere reso improbabile a piacimento, a patto di scegliere un N abbastanza grande. In particolare è possibile dimostrare che il più piccolo N che soddisfa (3.2.3) aumenta con il logaritmo di β; questo permette di scegliere valori molto piccoli per questo parametro, anche $\beta = 10^{-7}$ o $\beta = 10^{-10}$ (praticamente indistinguibili da 0), senza influenzare eccessivamente N. Ai fini pratici β svolge quindi un ruolo marginale, dal momento che la confidenza con cui è garantita la feasibility della soluzione a scenario per il problema originale può essere "praticamente" resa pari a 1.

3.2.1 Ruolo di α

Il parametro α quantifica la porzione di vincoli da eliminare rispetto agli N totali nell'algoritmo a scenario. Finora è stato definito genericamente come appartenente all'intervallo $[0, \epsilon)$ senza che sia stato specificato un criterio con cui sceglierlo. In generale α è un parametro a disposizione dell'utente con cui è possibile migliorare il grado di approssimazione di $x^*_{N,\lfloor \alpha N \rfloor}$ rispetto alla soluzione ottima del problema chance-constrained. In particolare all'avvicinarsi di α ad ϵ si ottiene una approssimazione empirica via via migliore al costo di un aumento del peso computazionale. Fissato β, infatti, incrementare α causa un incremento di N secondo il legame espresso da (3.2.3). Al limite si ha che per $\alpha \to \epsilon$ $N \to \infty$. Per valori di α vicini allo 0 si ha invece il caso opposto: minore peso computazionale ma scarsa accuratezza della soluzione. In particolare si ottengono soluzioni $x^*_{N,\lfloor \alpha N \rfloor}$ aventi probabilità di violazione dei vincoli ben più basse del livello ϵ consentito. Scegliere $\alpha = 0$ quindi si adatta bene alla soluzione di problemi chance-constrained nella forma (3.2.1) in cui il livello di violazione ϵ consentito è basso, o al limite pari a 0 (problemi worst-case). Come anticipato nella Definizione 3 del Capitolo 1 problemi di questo tipo sono tipici nel contesto dell'approssimazione

di sistemi ibridi stocastici, quando si vuole garantire che la distanza tra realizzazioni del sistema originale e del modello approssimante si mantenga limitata per una gran maggioranza delle possibili istanze (al limite tutte) dei processi stocastici. Per $\alpha = 0$ il Teorema 9 assume la forma seguente:

Teorema 10. In riferimento al problema (3.2.1), si estraggano N realizzazioni $\delta^{(1)}, \delta^{(2)}, ..., \delta^{(N)}$ di $\delta \in \Delta$ e si risolva il problema campionato

$$\begin{aligned} &\min_{x \in \mathbb{R}^d} J(x) \\ &\text{soggetto a } f(x, \delta^{(i)}) \leq 0 \; i = 1, \ldots, N \quad . \end{aligned}$$

Si chiami x_N^* la soluzione di questo problema. Scelti un parametro di violazione $\epsilon \in (0, 1)$ e un parametro di confidenza $\beta \in (0, 1)$, se N è tale che:

$$\sum_{i=0}^{d-1} \binom{N}{i} \epsilon^i (1 - \epsilon)^{N-i} \leq \beta \tag{3.2.4}$$

allora con confidenza non più piccola di $1 - \beta$, x_N^* soddisfa tutti i vincoli in Δ tranne al massimo una porzione di probabilità ϵ. Formalmente:

$$P(\delta : f(x_N^*, \delta) > 0) \leq \epsilon \; , \tag{3.2.5}$$

L'interpretazione del teorema è analoga a quella del teorema precedente: ancora una volta è presente un trade-off tra il livello di violazione garantito dalla soluzione trovata e il numero di estrazioni N. Al diminuire di ϵ si ha un incremento di N. Al limite, se si richiede che la soluzione x_N^* soddisfi *tutti* i vincoli, ovvero che sia $\epsilon = 0$, sarà necessario estrarre un numero infinito di realizzazioni di δ.

3.2.2 Illustrazione del teorema nel caso particolare $d = 1$

Per chiarire il risultato del Teorema 9 ci si può ricondurre al caso semplice in cui il numero di variabili di ottimizzazione d è pari a 1 e il vettore x in (3.2.1) risulta quindi uno scalare. Tra i problemi illustrati nella precedente sezione l'a.1 presenta una sola variabile di ottimizzazione dal momento che $x = h \in \mathbb{R}$ (si confronti la (3.1.4) con la riscrittura (3.2.1)); per questo motivo il Teorema 9 con $d = 1$ verrà illustrato in riferimento a questo caso.

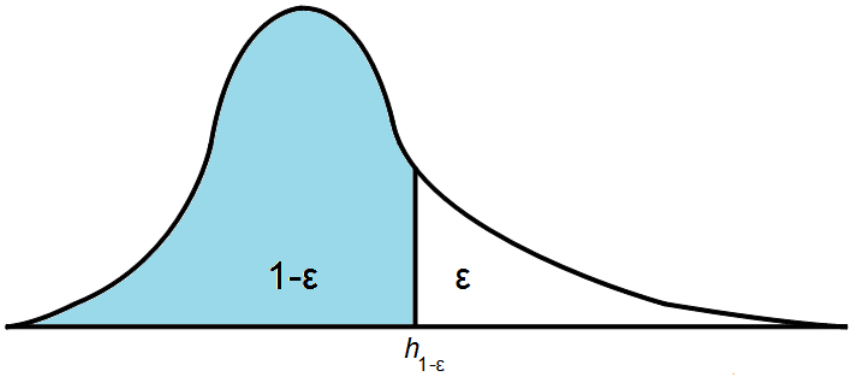


Figura 3.2.1: Quantile $h_{1-\epsilon}$ di ordine $1-\epsilon$ della distribuzione incognita di $\sup_{t\in[0,T]} ||y_{S_1,t} - y_{S_2,t}||^2$. La forma della densità di probabilità rappresentata ha solo valenza esemplificativa.

Ricordiamo la forma del problema a.1:

$$\begin{aligned} &\min_{h} h \\ &\text{soggetto a } P\left\{\sup_{t\in[0,T]} ||y_{S_1,t} - y_{S_2,t}||^2 \leq h\right\} \geq 1-\epsilon \end{aligned} \tag{3.2.6}$$

Questo problema consiste nella ricerca del quantile $h_{1-\epsilon}$ di ordine $1-\epsilon$ della distribuzione di probabilità incognita della variabile aleatoria $\sup_{t\in[0,T]} ||y_{S_1,t} - y_{S_2,t}||^2$. Si veda la Figura 3.2.1.

Per ogni valore $\tilde{h} > h_{1-\epsilon}$ si ha:

$$\int_0^{\tilde{h}} g(h) \cdot \mathrm{dh} \geq 1-\epsilon \ ,$$

dove con $g(\cdot)$ si è indicata la densità di probabilità della variabile aleatoria $\sup_{t\in[0,T]} ||y_{S_1,t} - y_{S_2,t}||^2$. Di conseguenza, ogni valore $\tilde{h}$ maggiore o uguale al quantile $h_{1-\epsilon}$ è feasibile per il problema (3.2.6).

L'algoritmo dell'approccio a scenario in questo caso risulta molto semplificato: è sufficiente infatti estrarre N realizzazioni di $\sup_{t\in[0,T]} ||y_{S_1,t} - y_{S_2,t}||^2$, disporle in ordine decrescente, e prendere il $\lfloor \alpha N \rfloor + 1$-esimo valore in ordine di grandezza $h^*_{N,\lfloor \alpha N \rfloor}$, con $0 \leq \alpha < \epsilon$. Come spiegato, tale valore ottenuto sarà feasible per il problema originale se:

$$h^*_{N,\lfloor \alpha N \rfloor} \geq h_{1-\epsilon} \ . \tag{3.2.7}$$

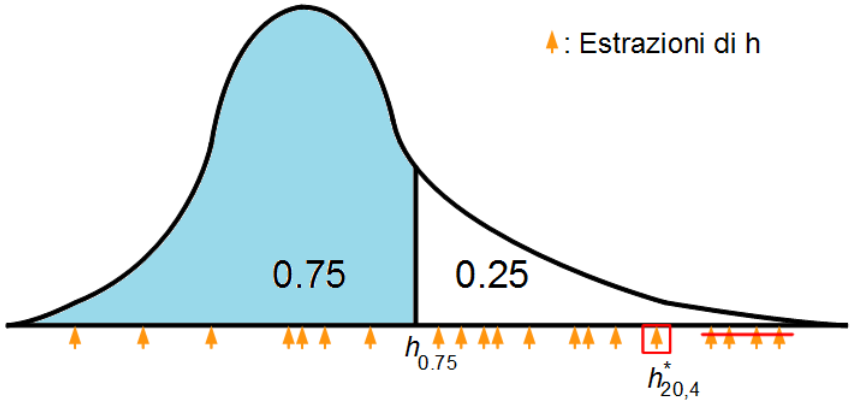


Figura 3.2.2: Sovrastima del quantile di ordine 0.75: eliminate le $4 = 0.20 \cdot 20$ estrazioni barrate, si sceglie un $h^*_{20,4}$ ben al di sopra di $h_{0.75}$. La stima ottenuta è quindi feasible per il problema originale.

Se introduciamo una variabile aleatoria Q che conta il numero di estrazioni inferiori al quantile $h_{1-\epsilon}$ sulle N totali – Q è variabile aleatoria con distribuzione binomiale di parametri N e $1-\epsilon$, $Q \sim \mathcal{B}(N, 1-\epsilon)$ – è possibile valutare quantitativamente la probabilità che la condizione (3.2.7) sia soddisfatta; ovvero è possibile valutare la probabilità che estraendo N realizzazioni di $\sup_{t\in[0,T]} ||y_{S_1,t} - y_{S_2,t}||^2$, ordinandole e prendendo la $\lfloor \alpha N \rfloor + 1$-esima in ordine di grandezza si ottenga una soluzione approssimata $h^*_{N,\lfloor \alpha N \rfloor}$ feasible per il problema originale (3.2.6). Si ragioni come segue: affinché $h^*_{N,\lfloor \alpha N \rfloor} \geq h_{1-\epsilon}$ è necessario avere più di $\lfloor \alpha N \rfloor$ estrazioni con valore superiore al quantile (o, equivalentemente avere $Q \leq \lfloor \alpha N \rfloor$), in modo che queste possano venir eliminate senza portare a stimare valori di $h^*_{N,\lfloor \alpha N \rfloor}$ inferiori a $h_{1-\epsilon}$. Un esempio di questa situazione è illustrato in Figura 3.2.2, in cui si è posto $N = 20$, $\epsilon = 0.25$, $\alpha = 0.20$.

Alla luce di queste considerazioni la probabilità che (3.2.7) sia soddisfatta è così calcolabile:

$$\begin{aligned} P\left\{h^*_{N,\lfloor \alpha N \rfloor} \geq h_{1-\epsilon}\right\} &= P\{Q \leq \lfloor \alpha N \rfloor\} = 1 - P\{Q > \lfloor \alpha N \rfloor\} = \\ &= 1 - (P\{Q = \lfloor \alpha N \rfloor + 1\} + P\{Q = \lfloor \alpha N \rfloor + 2\} + \dots \\ &\quad \cdots + P\{Q = N-1\} + P\{Q = N\}) = \qquad (3.2.8) \\ &= 1 - \sum_{i=0}^{\lfloor \alpha N \rfloor} \binom{N}{i} \epsilon^i (1-\epsilon)^{N-i} \qquad (3.2.9) \end{aligned}$$

Se vogliamo quindi che la soluzione a scenario $h^*_{N,\lfloor \alpha N \rfloor}$ sia feasible per il pro-

blema originale con un livello di confidenza $1-\beta$ alto a piacere, è sufficiente scegliere N tale che:

$$\sum_{i=0}^{\lfloor \alpha N \rfloor} \binom{N}{i} \epsilon^i (1-\epsilon)^{N-i} \leq \beta \; . \tag{3.2.10}$$

Si noti che il teorema con $d = 1$ è così dimostrato, dal momento che la (3.2.10) è proprio la condizione (3.2.3) del teorema nel caso particolare $d = 1$.

L'approccio a scenario nel caso di variabili di ottimizzazione scalari riprende sostanzialmente un risultato già noto nell'ambito delle statistiche d'ordine. Il risultato innovativo e sorprendente di questa teoria è l'estensione al caso multidimensionale, in cui la condizione su N continua a valere con qualche leggera modifica: si confronti a questo proposito la (3.2.10) con la (3.2.3). Per la dimostrazione del risultato generale con $d > 1$ si rimanda a [13].

3.3 Scelta della parametrizzazione del problema chance-constrained

L'algoritmo dell'approccio a scenario è stato illustrato nella precedente sezione in riferimento al generico problema chance-constrained:

$$\begin{aligned} &\min_x J(x) \\ &\text{soggetto a } P_\delta \{f(x,\delta) \leq 0\} \geq 1-\epsilon \end{aligned}$$

con $J(x)$ e $f(x,\delta)$ convesse in x, $\forall \delta \in \Delta$. Queste ipotesi si traducono in condizioni sui sistemi e sulle parametrizzazioni utilizzabili nella risoluzione dei problemi a.1 e a.2 (Sezione 3.1.1), e b.1 e b.2 (Sezione 3.1.2).

Problema a.1: il problema ha già cifra di merito e vincoli convessi (poiché lineari) nelle variabili di ottimizzazione. La tecnica dell'approccio a scenario è quindi applicabile senza ulteriori condizioni.

Problema a.2: è necessario utilizzare un bound $h(x_{1,0}, \theta)$ che sia convesso in θ per ottenere la convessità della cifra di merito $E_{x_{1,0}}[h(x_{1,0}, \theta)]$. Contemporaneamente è necessario che $h(x_{1,0}, \theta)$ sia concava in θ per ga-

rantire la convessità dei vincoli $\sup_{t\in[0,T]} ||y_{S_1,t} - y_{S_2,t}||^2 \leq h(x_{1,0},\theta)$. L'unica possibilità è dunque quella di scegliere $h(x_{1,0},\theta)$ linearmente parametrizzato in θ. Una possibile scelta simile a quella del bound ottenuto tramite la funzione stocastica di simulazione nel corollario 7 è la seguente:

$$h(x_{1,0},\theta) = \begin{bmatrix} x_{1,0}^T & 1 \end{bmatrix} M \begin{bmatrix} x_{1,0} \\ 1 \end{bmatrix} ,$$

dove M è una matrice quadrata di dimensione $n+1 \times n+1$ – n è la dimensionalità di $x_{1,0}$ – simmetrica e semidefinita positiva e θ sono gli elementi di questa matrice. Alla luce di questa scelta la cifra di merito $E_{x_{1,0}}[h(x_{1,0},\theta)]$ può essere riscritta nel seguente modo utilizzando semplici manipolazioni algebriche:

$$\begin{aligned} E_{1,0}[h(x_{1,0},\theta)] &= E_{x_{1,0}}\left[\begin{bmatrix} x_{1,0}^T & 1 \end{bmatrix} M \begin{bmatrix} x_{1,0} \\ 1 \end{bmatrix}\right] = \\ &= E_{x_{1,0}}\left[tr\left(\begin{bmatrix} x_{1,0}^T & 1 \end{bmatrix} M \begin{bmatrix} x_{1,0} \\ 1 \end{bmatrix}\right)\right] = \\ &= tr\left(E_{x_{1,0}}\left[M \begin{bmatrix} x_{1,0} \\ 1 \end{bmatrix} \begin{bmatrix} x_{1,0}^T & 1 \end{bmatrix}\right]\right) = \\ &= tr\left(M \cdot E_{x_{1,0}}\left[\begin{bmatrix} x_{1,0} \\ 1 \end{bmatrix} \begin{bmatrix} x_{1,0}^T & 1 \end{bmatrix}\right]\right) , \end{aligned}$$

dove $tr(\cdot)$ è la funzione traccia. Si noti che la matrice $E_{x_{1,0}}\left[\begin{bmatrix} x_{1,0} \\ 1 \end{bmatrix} \begin{bmatrix} x_{1,0}^T & 1 \end{bmatrix}\right]$ può essere calcolata analiticamente o stimata attraverso un metodo Monte-Carlo, nota la distribuzione di $x_{1,0}$. In definitiva il problema a.2 assume la seguente forma:

$$\begin{aligned} &\min_{M\succeq 0} tr\left(M \cdot E_{x_{1,0}}\left[\begin{bmatrix} x_{1,0} \\ 1 \end{bmatrix} \begin{bmatrix} x_{1,0}^T & 1 \end{bmatrix}\right]\right) \\ &\text{soggetto a } P\left\{\sup_{t\in[0,T]} ||y_{S_1,t} - y_{S_2,t}||^2 \leq \begin{bmatrix} x_{1,0}^T & 1 \end{bmatrix} M \begin{bmatrix} x_{1,0} \\ 1 \end{bmatrix}\right\} \geq 1-\epsilon . \end{aligned}$$

Non sono necessarie ipotesi sulla dinamica dei due sistemi S_1 e S_2.

Problema b.1: in questo caso la cifra di merito (lineare in h) non dà

problemi. Per quello che riguarda il vincolo, tuttavia, è necessario ipotizzare un legame lineare tra y_{s_2} e $x_{2,0}$, affinché il vincolo risulti convesso anche nella variabile $x_{2,0}$. In particolare tale ipotesi è sempre verificata nei JLSS analizzati, dal momento che la dinamica continua e la mappa dei reset sono lineari. Piu' precisamente, y_{S_2} si può esprimere nel modo seguente:

$$y_{S_2,t} = \Psi_t x_{2,0} \ , \tag{3.3.1}$$

dove Ψ_t è la matrice di transizione del sistema di dimensione $2 \times m$, con m dimensione dello stato $x_{2,0}$. L'espressione (3.3.1) può essere espansa evidenziando le componenti di $x_{2,0}$:

$$y_{S_2,t} = \sum_{i=1}^{m} \left[\Psi_t\right]_i \left[x_{2,0}\right]_i \ ,$$

dove $\left[x_{2,0}\right]_i$ è la componente i-esima di $x_{2,0}$ e $\left[\Psi_t\right]_i$ è la colonna i-esima di Ψ_t.

Se si sceglie quindi come stato iniziale un vettore avente tutte componenti nulle a eccezione di un 1 in posizione i ($i \in \{1, \ldots, m\}$), si otterrà in uscita il vettore $\left[\Psi_t\right]_i$, i-esima colonna della matrice di transizione Ψ_t. È possibile quindi ricavare l'intera matrice Ψ_t per colonne, accostando gli m vettori $y_{S_2,t}$ ottenuti simulando il sistema a partire dalle condizioni iniziali $\begin{bmatrix}1 & 0 & 0 & \ldots & 0\end{bmatrix}^T$, $\begin{bmatrix}0 & 1 & 0 & \ldots & 0\end{bmatrix}^T$, $\begin{bmatrix}0 & 0 & 1 & \ldots & 0\end{bmatrix}^T$, etc.

Problema b.2: per garantire la convessità della cifra di merito si può scegliere la stessa parametrizzazione θ del problema a.2. Per garantire la convessità dei vincoli è invece necessario introdurre l'ipotesi aggiuntiva di legame lineare tra $x_{2,0}$ e $x_{1,0}$: $x_{2,0} = \rho(x_{1,0}, \lambda) = Lx_{1,0}$, dove L è una matrice $n-2 \times n$ nel caso 1 (dinamica continua ridotta) e $n \times n$ nel caso 2 e 3 (eliminazione moto Browniano e eliminazione processo di Poisson) e λ è il vettore degli elementi di tale matrice. In questo modo si ha che $x_{2,0}$ dipende linearmente da L e grazie alla linearità dei JLSS si ha che $y_{S_2,t}$ dipende a sua volta linearmente da L. Adottando la stessa riscrittura della cifra di

merito vista per il problema a.2, il problema b.2 prende la seguente forma:

$$\min_{M\succeq 0, L} tr\left(M \cdot E_{x_{1,0}}\left[\begin{bmatrix} x_{1,0}^T & 1\end{bmatrix}^T \begin{bmatrix} x_{1,0} \\ 1\end{bmatrix}\right]\right)$$
$$\text{soggetto a } P\left\{\sup_{t\in[0,T]} ||y_{S_1,t} - y_{S_2,t}(Lx_{1,0})||^2 \leq \begin{bmatrix} x_{1,0}^T & 1\end{bmatrix}^T M \begin{bmatrix} x_{1,0} \\ 1\end{bmatrix}\right\} \geq 1-\epsilon\,.$$

3.4 Aspetti implementativi

3.4.1 Simulazione di equazioni differenziali stocastiche

Per implementare l'algoritmo a scenario descritto nella Sezione 3.3 è necessario estrarre N realizzazioni dell'incertezza δ che come visto, corrisponde alle realizzazioni del moto Browniano e del processo di Poisson per i problemi del tipo 1 e alle realizzazioni del moto Browniano e del processo di Poisson unite alla condizione iniziale casuale $x_{1,0}$ per i problemi di tipo 2. Ai fini del calcolo di $y_{S_1,t}$ e $y_{S_2,t}$ si procede quindi estraendo N realizzazioni dei processi stocastici w_t (moto Browniano) e p_t (processo di Poisson) e simulando i sistemi S_1 e S_2 alimentati da tali processi. In particolare per la simulazione si è fatto ricorso al metodo di *Eulero-Maruyama* che sostanzialmente rappresenta un'estensione del metodo di Eulero in avanti applicabile alle equazioni differenziali stocastiche.

Metodo di *Eulero-Maruyama*. Data una SDE nella forma:

$$dx_t = f(x_t)dt + g(x_t)dW_t, \quad X(0) = X_0, \quad 0 \leq t \leq T \tag{3.4.1}$$

si discretizzi $[0,T]$ in L intervalli di ampiezza $\Delta t = \frac{T}{L}$ e sia $\tau_j = j\Delta t$. Il metodo di Eulero-Maruyama assume la forma:

$$x_j = x_{j-1} + f(x_{j-1})\Delta t + g(x_{j-1})(W(\tau_j) - W(\tau_{j-1})), \quad j = 1, ..., L \tag{3.4.2}$$

Una giustificazione intuitiva della bontà di questo metodo si può ottenere confrontando le due equazioni e notando come ciascun addendo di (3.4.1) sia approssimato da uno di (3.4.2). I risultati sulla convergenza del metodo sono illustrati in [5]. Alcuni grafici risultanti dell'applicazione di questo metodo

sono già stati mostrati nelle simulazioni del capitolo 2 (si vedano le Figure 2.3.1, 2.3.2, 2.3.3).

3.4.2 Procedura euristica per il problema di ottimizzazione del modello approssimante

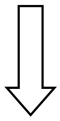
Nel caso di ottimizzazione del modello si puo' ridurre il carico computazionale utilizzando una procedura in due passi, che viene ora illustrata con riferimento al problema b.1. La procedura in due passi adottata consiste nell'approssimare il problema chance-constrained con un problema a scenario, con un numero limitato N_1 di vincoli estratti. Una volta calcolata la soluzione di questo problema, si fissa $x_{2,0}$ al valore (sub-ottimo) calcolato e si ottimizza solo il bound h sulla qualità dell'approssimante inizializzato con quella condizione iniziale $x_{2,0}$. La procedura è illustrata schematicamente nella pagina seguente.

Soluzione del problema ridotto con N_1 scenari e k_1 vincoli da eliminare

Si noti che anche se il problema è ridotto è comunque conveniente eliminare qualche vincolo per evitare risultati indesiderati dovuti a estrazioni sfortunate. Risolto il problema worst-case:

$$\begin{aligned} &\min_{x_{2,0},h} h \\ &\text{soggetto a } \sup_{t\in[0,T]} ||y^{(j)}_{S_1,t} - y^{(j)}_{S_2,t}(x_{2,0})||^2 \leq h \ \forall j = 1,\dots,N_1 \end{aligned}$$

si procede quindi all'eliminazione di k_1 vincoli attraverso un algoritmo greedy: $x^*_{2,0}$ è lo stato iniziale ottimo trovato.



Soluzione del problema analogo all'1.a con N scenari e $k = \lfloor \alpha N \rfloor$ vincoli da eliminare, con l'approssimante inizializzato in $x^*_{2,0}$

Dopo aver "parzialmente ottimizzato" lo stato iniziale si procede al miglioramento del bound h. Per farlo si risolve il problema a scenario:

$$\begin{aligned} &\min_{h} h \\ &\text{soggetto a } \sup_{t\in[0,T]} ||y^{(i)}_{S_1,t} - y^{(i)}_{S_2,t}(x^*_{2,0})||^2 \leq h \ \forall i = 1,\dots,N \end{aligned}$$

Con le estrazioni $w^{(i)}$ e $p^{(i)}$ indipendenti da quelle utilizzate per calcolare $x^*_{2,0}$. h^* è la soluzione del problema di ottimizzazione.

È importante osservare che per la soluzione h^* ottenuta al passo 2 si possono dare le garanzie dell'approccio a scenario pur di scegliere N in accordo a (3.2.10).

Capitolo 4

Risultati

In questo capitolo vengono illustrati i risultati ottenuti mediante l'approccio proposto nel Capitolo 3 per l'approssimazione di sistemi ibridi stocastici. Più precisamente si considera l'esempio numerico della Sezione 2.3 e si confrontano i risultati ottenuti con quelli dell'approccio basato sulla funzione stocastica di simulazione. La presentazione include dettagli sugli aspetti di carattere implementativo; il codice MATLAB è riportato in appendice.

4.1 Esempio numerico

Viene qui richiamato brevemente l'esempio numerico oggetto di studio.

Il sistema da approssimare è un JLSS descritto da:

$$S_1 : \begin{cases} dx_{1,t} = A_1 x_{1,t} dt + F_1 x_{1,t} dw_t + R_1 x_{1,t} dp_t \\ y_{S_1,t} = C_1 x_{1,t} \end{cases},$$

con

$$A_1 = diag\left(\begin{bmatrix} -1 & -10 \\ 10 & -1 \end{bmatrix}, \begin{bmatrix} -2 & -20 \\ 20 & -1 \end{bmatrix}, \begin{bmatrix} -2 & 0 \\ 0 & -2.5 \end{bmatrix} \right)$$

$$F_1 = 0.5 \cdot \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}, \; R_1 = 0.7 \cdot I_6$$

$$C_1 = \begin{bmatrix} 0.84 & -1.03 & 1.07 & -0.88 & 0.5 & 0 \\ -0.6 & -1.35 & -0.26 & -0.27 & 0 & -0.5 \end{bmatrix}.$$

I sistemi approssimanti considerati allo scopo di riprodurre l'andamento del segnale di interesse $y_{S_1,t}$ nell'orizzonte predittivo $[0, T] = [0, 12]$ sono 3:

1. Sistema S_2 con dinamica continua ridotta attraverso l'eliminazione delle ultime due componenti del vettore di stato x:

$$S_2 : \begin{cases} dx_{2,t} = A_2 x_{2,t} dt + F_2 x_{2t} dw_t + R_2 x_{2,t} dp_t \\ y_{S_2,t} = C_2 x_{2,t} \end{cases},$$

con

$$A_2 = diag\left(\begin{bmatrix} -1 & -10 \\ 10 & -1 \end{bmatrix}, \begin{bmatrix} -2 & -20 \\ 20 & -1 \end{bmatrix} \right),\ F_2 = 0.5 \cdot I_4$$
$$C_2 = \begin{bmatrix} 0.84 & -1.03 & 1.07 & -0.88 \\ -0.6 & -1.35 & -0.26 & -0.27 \end{bmatrix},\ R_2 = 0.7 \cdot I_4$$
$$x_2 \in \mathbb{R}^4 .$$

2. Sistema S_2 ottenuto eliminando il contributo del moto Browniano:

$$S_2 : \begin{cases} dx_{2,t} = A_1 x_{2,t} dt + R_1 x_{2,t} dp_t \\ y_{S_2,t} = C_1 x_t \end{cases}.$$

3. Sistema S_2 ottenuto eliminando il contributo del processo di Poisson:

$$S_2 : \begin{cases} dx_{2,t} = A_1 x_{2,t} dt + F_1 x_{2,t} dw_t \\ y_{S_2,t} = C_1 x_t \end{cases}.$$

4.2 Valutazione della qualità del sistema approssimante

Consideriamo il caso in cui la condizione iniziale $x_{1,0}$ del sistema S_1 da approssimare sia fissata in modo deterministico, così come la corrispondente condizione iniziale $x_{2,0}$ del sistema approssimante, ottenuta mediante una mappa nota: $x_{2,0} = \rho(x_{1,0})$. La mappa $\rho : X_2 \to X_2$ adottata è l'identità nel caso dei sistemi approssimanti che preservano l'ordine della dinamica continua ($X_2 = X_1 = \mathbb{R}^6$), mentre nel caso di dinamica continua ridotta ($X_2 = \mathbb{R}^4$) la condizione iniziale del sistema approssimante viene ottenuta

eliminando da $x_{1,0}$ le ultime due componenti. In formule: $x_{2,0} = Lx_{1,0}$ dove

$$L = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}.$$

Il problema chance-constrained da risolvere per valutare la qualità dell'approssimazione è:

$$\begin{aligned} &\min_h h \\ &\text{soggetto a } P\left\{ \sup_{t\in[0,T]} ||y_{S_1,t} - y_{S_2,t}||^2 \leq h \right\} \geq 1 - \epsilon \ , \end{aligned}$$

dove $\epsilon \in (0, 1)$ è fissato pari a 0.25.

Per risolvere questo problema secondo l'approccio a scenario descritto nel precedente capitolo, è sufficiente eseguire N simulazioni dei sistemi S_1 e del suo approssimante S_2 alimentati dalle stesse realizzazioni di p_t e w_t i stessi processi probabilistici, calcolare i valori di $\sup_{t\in[0,T]} ||y_{S_1,t} - y_{S_2,t}||^2$ corrispondenti, ordinarli in ordine decrescente, e prendere il $k+1$-esimo in ordine di grandezza, con $k = \lfloor \alpha N \rfloor$, dove $0 \leq \alpha < \epsilon$. La quantità ottenuta viene chiamata $h^*_{N,k}$ e soddisfa la condizione

$$P\left\{ \sup_{t\in[0,T]} ||y_{S_1,t} - y_{S_2,t}|| \leq h^*_{N,k} \right\} \geq 1 - \epsilon \ ,$$

con alta confidenza $1 - 10^{-10}$ (in pratica 1) se N soddisfa la (3.2.10) con $\beta = 10^{-10}$. Di seguito viene descritto per punti il procedimento di risoluzione del problema:

1. Fissare $\alpha \in [0, \epsilon)$, $\beta = 10^{-10}$, e calcolare il numero di estrazioni N corrispondente

2. Estrazione casuale di N realizzazioni indipendenti del processo di Poisson p_t e del moto Browniano w_t: $p_t^{(i)}$, $w_t^{(i)}$, $t \in [0, T]$, $i = 1, 2, \ldots, N$

3. Simulazione dei sistemi S_1 e S_2 alimentati dalla stessa coppia di realizzazioni di processi stocastici $w_t^{(i)}$ e $p_t^{(i)}$, con condizioni iniziali $x_{1,0}$ e $x_{2,0} = \rho(x_{1,0})$ rispettivamente e calcolo di $h^{(i)} = \sup_{t\in[0,T]} ||y^{(i)}_{S_1,t} - y^{(i)}_{S_2,t}||^2$, $i = 1...N$.

4. Ordinamento decrescente degli $h^{(i)}$: il primo valore in ordine di grandezza è h^*_N, soluzione del problema worst-case, mentre il $k+1$-esimo in ordine di grandezza, con $k = \lfloor \alpha N \rfloor$ è $h^*_{N,k}$, soluzione del problema chance-constrained

Il numero di variabili di ottimizzazione è $d = 1$. Posto $\alpha = 0.2$ si ricava che il valore corrispondente di N che soddisfa la condizione (3.2.10) con $\epsilon = 0.25$ e $\beta = 10^{-10}$ è $N = 2884$. L'upper bound ottenuto $h^*_{N,k}$, sul quadrato della norma della differenza tra $y_{S_1,t}$ e $y_{S_2,t}$ con $t \in [0, T]$, è mostrato nei seguenti grafici insieme a h^*_N, soluzione approssimata del problema worst-case, e all'upper bound al 75% ($\epsilon = 0.25$) ottenuto tramite l'approccio basato sulla funzione stocastica di simulazione illustrato nel Capitolo 2. I tre grafici riportano anche alcune realizzazioni di $||y_{S_1,t} - y_{S_2,t}||^2$ e si riferiscono ai tre sistemi approssimanti considerati. Gli stati iniziali $x_{1,0}$ e $x_{2,0}$ per sistema e approssimante sono gli stessi utilizzati nel Capitolo 2 e cioè $x_{1,0} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T$ e $x_{2,0} = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix}^T$ per il primo sistema approssimante, a dinamica continua ridotta, mentre $x_{1,0} = x_{2,0} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}^T$ per gli altri due sistemi approssimanti. La percentuale di realizzazioni di $||y_{S_1,t} - y_{S_2,t}||^2$ che violano il bound della soluzione chance-constrained dovrebbe essere pari a $100\epsilon\% = 25\%$.

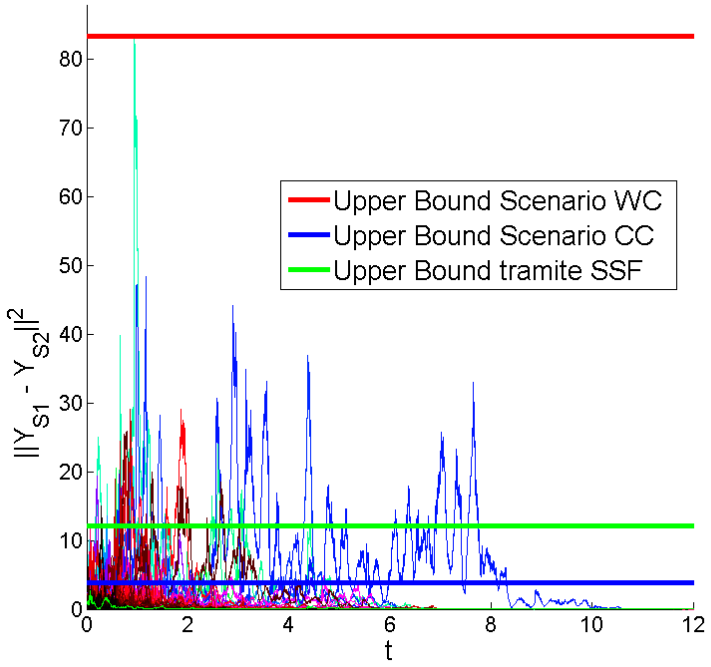


Figura 4.2.1: Caso 1: Dinamica continua ridotta – Qualità del sistema approssimante a condizione iniziale fissata e nota per $\epsilon = 0.25$.

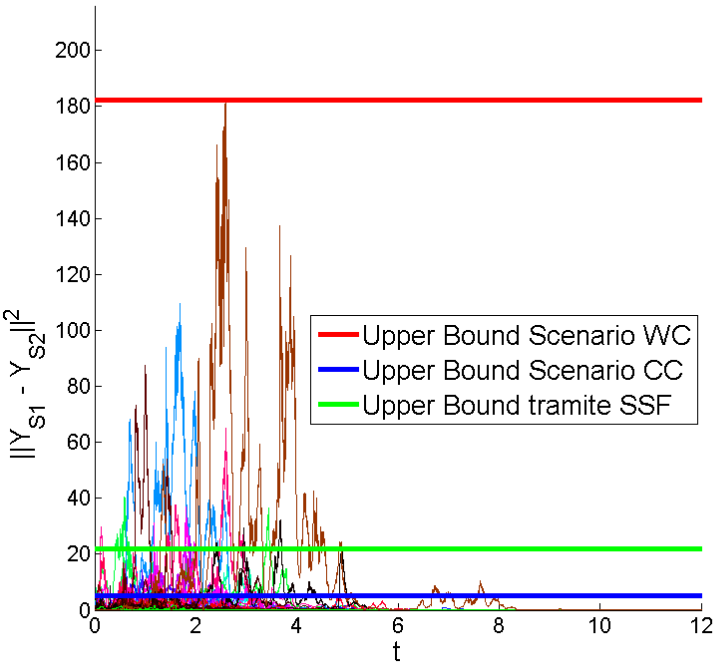


Figura 4.2.2: Caso 2: Eliminazione del moto Browniano – Qualità del sistema approssimante a condizione iniziale fissata e nota per $\epsilon = 0.25$.

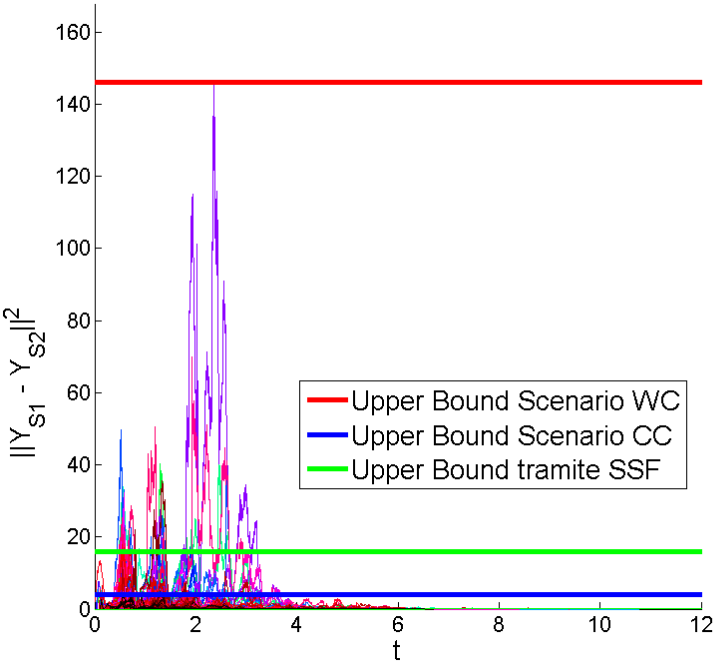


Figura 4.2.3: Caso 3: Eliminazione del processo di Poisson – Qualità del sistema approssimante a condizione iniziale fissata e nota per $\epsilon = 0.25$.

I risultati ottenuti sono molto buoni. Per ciascuno dei sottocasi si nota immediatamente che il bound ottenuto con l'approccio a scenario (chance-constrained) è nettamente più basso del bound ottenuto con l'approccio basato sulla SSF. Questi risultati sono stati poi testati su 3000 nuove realizzazioni indipendenti e si è valutato empiricamente il livello ϵ di violazione. I risultati complessivi con valori del bound ("bound") e stime di ϵ ("ϵ_{emp}") sono riportati nella seguente tabella riassuntiva:

	Scenario WC	Scenario CC	Approccio SSF
Caso 1	bound: 83.3	bound: 3.82	bound: 12.04
	$\epsilon_{emp} = 0.002$	$\epsilon_{emp} = 0.213$	$\epsilon_{emp} = 0.014$
Caso 2	bound: 182.4	bound: 5.25	bound: 21.8
	$\epsilon_{emp} = 0.005$	$\epsilon_{emp} = 0.204$	$\epsilon_{emp} = 0.027$
Caso 3	bound: 146.6	bound: 4.14	bound: 15.93
	$\epsilon_{emp} = 0.001$	$\epsilon_{emp} = 0.197$	$\epsilon_{emp} = 0.006$

Tabella 4.1: Tabella riassuntiva dei risultati della valutazione della qualità dell'approssimazione a condizione iniziale fissata e nota per i tre modelli approssimanti. Caso1 – dinamica continua ridotta, Caso 2 – Eliminazione del moto Browniano, Caso 3 – Eliminazione del processo di Poisson.

Come è lecito aspettarsi, la soluzione worst-case corrisponde a ϵ piccolo,

mentre la soluzione chance-constrained approssimata corrisponde a un valore di $\epsilon \approx 0.2$, coerentemente con la scelta $\alpha = 0.2$. L'approccio a scenario chance-constrained si dimostra quindi più efficiente dell'approccio basato sulla SSF. Per quest'ultimo infatti la probabilità di violazione effettiva è piccola, prossima a quella del caso worst-case. I tempi di calcolo per ottenere la soluzione chance-constrained approssimata mediante l'approccio a scenario sono stati di circa 6 minuti utilizzando un calcolatore con processore Intel Core Duo E7300 2.67Ghz, MATLAB 7.9.0, (R2009b), CVX versione 1.21. Se si ponesse $\alpha = 0.23$ ($N = 18591$) il tempo di calcolo sarebbe invece di circa 40 minuti.

Consideriamo ora il caso in cui la condizione iniziale del sistema S_1 da approssimare non sia fissata ma sia descritta da una variabile casuale con una certa distribuzione di probabilità. La corrispondente condizione iniziale del sistema approssimante è anch'essa una variabile casuale ottenuta attraverso la mappa $x_{2,0} = \rho(x_{1,0})$ definita prima. Si vuole valutare la qualità dei tre sistemi approssimanti risolvendo il problema chance-constrained:

$$\min_{M \succeq 0} E_{x_{1,0}} \left[\begin{bmatrix} x_{1,0}^T & 1 \end{bmatrix} M \begin{bmatrix} x_{1,0} \\ 1 \end{bmatrix} \right]$$

soggetto a

$$P \left\{ \sup_{t \in [0,T]} ||y_{S_1,t}(x_{1,0}) - y_{S_2,t}(\rho(x_{1,0}))||^2 \leq \begin{bmatrix} x_{1,0}^T & 1 \end{bmatrix} M \begin{bmatrix} x_{1,0} \\ 1 \end{bmatrix} \right\} \geq 1 - \epsilon \ . \tag{4.2.1}$$

Il problema è stato risolto per 2 diverse distribuzioni di $x_{1,0}$: $x_{1,0} \sim \mathcal{N}(0,1)$, gaussiana standard, e $x_{1,0} \sim \mathcal{U}(0,1)$, uniforme in [0,1]. Per quanto riguarda l'eliminazione dei vincoli, si è fatto ricorso a due algoritmi diversi: un algoritmo *greedy* e un algoritmo *random*.

La soluzione a scenario del problema chance-constrained (4.2.1) si basa sulla seguente procedura:

1. Definizione dei parametri ϵ, β, α, d, e calcolo del numero di estrazioni N

2. Estrazione casuale di N realizzazioni indipendenti del processo di Poisson p_t e del moto Browniano w_t: $p_t^{(i)}$, $w_t^{(i)}$, $t \in [0,T]$, $i = 1, 2, \ldots, N$

3. Simulazione del sistema S_1 (condizione iniziale $x_{1,0}^{(i)}$) e S_2 (condizione iniziale $x_{2,0}^{(i)} = \rho(x_{1,0}^{(i)})$) alimentati dalla stessa coppia di realizzazioni

di $w_t^{(i)}$ e $p_t^{(i)}$: calcolo di $h^{(i)} = \sup_{t\in[0,T]} ||y_{S_1,t}^{(i)}(x_{1,0}^{(i)}) - y_{S_2,t}^{(i)}(\rho(x_{1,0}^{(i)}))||^2$ $\forall i = 1...N$.

4. Soluzione del problema worst-case (ovvero con tutti i vincoli):

$$\begin{aligned} &\min_{M\succeq 0} E_{x_{1,0}}\left[\begin{bmatrix} x_{1,0}^T & 1\end{bmatrix} M \begin{bmatrix} x_{1,0} \\ 1\end{bmatrix}\right] \\ &\text{soggetto a} \quad h^{(i)} \leq \begin{bmatrix} x_{1,0}^{(i)T} & 1\end{bmatrix} M \begin{bmatrix} x_{1,0}^{(i)} \\ 1\end{bmatrix} \quad i = 1, ..., N\ . \end{aligned} \tag{4.2.2}$$

5. Eliminazione dei vincoli mediante uno dei due algoritmi greedy o random descritti nella Sezione 3.3.

6. La soluzione del problema con $N - k$ vincoli ottenuta è la soluzione (approssimata) del problema chance-constrained ricercata.

Il numero di variabili di ottimizzazione è $d = 28$, dato che M è una matrice 7×7 simmetrica. I parametri utilizzati per la risoluzione del problema mediante l'approccio a scenario sono stati:

$$\begin{aligned} \epsilon &= 0.25 \\ \beta &= 10^{-10} \\ \alpha &= 0.1\ . \end{aligned}$$

Si nota subito che il parametro α è stato fissato ad un valore decisamente più basso rispetto a quanto fatto precedentemente per il caso con condizione iniziale nota. Il motivo è che l'algoritmo *greedy* ha un carico computazionale elevato che può diventare eccessivo se si aumenta troppo il numero di vincoli da eliminare.

I risultati ottenuti sono stati confrontati con l'approccio basato sulla SSF: a questo proposito bisogna ricordare che le condizioni da imporre per ottenere una funzione stocastica di simulazione portavano alla formulazione di un problema LMI, di feasibility, senza cifra di merito da ottimizzare. Coerentemente con quanto fatto nel caso precedente con $x_{1,0}$ nota, si è scelto di ottimizzare il valore medio rispetto alla distribuzione di $x_{1,0}$ del bound sulla norma al quadrato della distanza. Dato che per ogni $x_0 = \begin{bmatrix} x_{1,0}^T & x_{2,0}^T\end{bmatrix}^T$ fissata si ha che

$$P\left\{\sup_{t\in[0,\infty)} ||y_{S_1,t} - y_{S_2,t}||^2 \leq \frac{1}{\epsilon}\phi(x_0)|x_0\right\} \geq 1 - \epsilon\ ,$$

con $\phi(x_0) = x_0^T M x_0$, allora si ottiene il seguente problema LMI:

$$\begin{aligned}
&\min_M E\left[\begin{bmatrix} x_{1,0}^T & \rho(x_{1,0})^T \end{bmatrix} M \begin{bmatrix} x_{1,0} \\ \rho(x_{1,0}) \end{bmatrix}\right] \\
&\text{soggetto a} \\
&M - C^T C \succeq 0 \\
&Q \preceq 0 \ ,
\end{aligned}$$

dove Q è la funzione lineare di M data in (2.2.2). La cifra di merito da minimizzare può essere espressa nel modo seguente:

$$tr\left(M \cdot E\left[\begin{bmatrix} x_{1,0} \\ \rho(x_{1,0}) \end{bmatrix}\begin{bmatrix} x_{1,0}^T & \rho(x_{1,0})^T \end{bmatrix}\right]\right) \ ,$$

dove la matrice $E\left[\begin{bmatrix} x_{1,0} \\ \rho(x_{1,0}) \end{bmatrix}\begin{bmatrix} x_{1,0}^T & \rho(x_{1,0})^T \end{bmatrix}\right]$ può essere calcolata analiticamente o attraverso il metodo Monte-Carlo.

Tabelle riassuntive e commenti

I risultati ottenuti per tutti i casi analizzati sono riportati nelle seguenti tabelle; per ciascun caso con "ottimo" si è indicato il valore atteso del bound sulla norma della differenza al quadrato delle uscite e con "ϵ_{emp}" si è indicato il livello di violazione empirico ottenuto su 3000 realizzazioni indipendenti.

1) Distribuzione Normale: confronto tra algoritmo greedy, random, e approccio basato su SSF.

$x_{1,0} \sim \mathcal{N}(0,1)$	Scenario: greedy	Scenario: random	Approccio SSF
Caso 1	ottimo: 3.49	ottimo: 3.88	ottimo: 10.13
	$\epsilon_{emp} = 0.109$	$\epsilon_{emp} = 0.092$	$\epsilon_{emp} = 0.025$
Caso 2	ottimo: 7.08	ottimo: 9.37	ottimo: 19.76
	$\epsilon_{emp} = 0.125$	$\epsilon_{emp} = 0.103$	$\epsilon_{emp} = 0.039$
Caso 3	ottimo: 8.13	ottimo: 9.57	ottimo: 15.63
	$\epsilon_{emp} = 0.11$	$\epsilon_{emp} = 0.105$	$\epsilon_{emp} = 0.054$

Tabella 4.2: Confronto tra scenario con algoritmo greedy, scenario con algoritmo random, approccio SSF quando la condizione iniziale è una gaussiana standard. Caso1 – dinamica continua ridotta, Caso 2 – Eliminazione del moto Browniano, Caso 3 – Eliminazione del processo di Poisson.

2) Distribuzione Uniforme: confronto tra algoritmo greedy, random, e approccio basato su SSF.

$x_{1,0} \sim \mathcal{U}(0,1)$	Scenario: greedy	Scenario: random	Approccio SSF
Caso 1	ottimo: 1.66	ottimo: 2.07	ottimo: 3.96
	$\epsilon_{emp} = 0.122$	$\epsilon_{emp} = 0.106$	$\epsilon_{emp} = 0.033$
Caso 2	ottimo: 3.40	ottimo: 4.22	ottimo: 7.11
	$\epsilon_{emp} = 0.098$	$\epsilon_{emp} = 0.102$	$\epsilon_{emp} = 0.039$
Caso 3	ottimo: 2.72	ottimo: 4.28	ottimo: 5.30
	$\epsilon_{emp} = 0.113$	$\epsilon_{emp} = 0.098$	$\epsilon_{emp} = 0.061$

Tabella 4.3: Confronto tra scenario con algoritmo greedy, scenario con algoritmo random, approccio SSF quando la condizione iniziale è uniformemente distribuita su $[0, 1]$. Caso1 – dinamica continua ridotta, Caso 2 – Eliminazione del moto Browniano, Caso 3 – Eliminazione del processo di Poisson.

Nella risoluzione con algoritmo *random* è stato possibile aumentare il valore di α, dal momento che il carico computazionale dell'algoritmo è notevolmente più basso. I parametri utilizzati in questo caso sono stati:

$$\epsilon = 0.25$$
$$\beta = 10^{-10}$$
$$\alpha = 0.15 \ .$$

I risultati ottenuti sono qui elencati:

	$x_{1,0} \sim \mathcal{N}(0,1)$	$x_{1,0} \sim \mathcal{U}(0,1)$
Caso 1	ottimo: 3.21	ottimo:1.51
	$\epsilon_{emp} = 0.158$	$\epsilon_{emp} = 0.163$
Caso 2	ottimo: 6.33	ottimo: 2.50
	$\epsilon_{emp} = 0.139$	$\epsilon_{emp} = 0.145$
Caso 3	ottimo: 5.31	ottimo: 2.16
	$\epsilon_{emp} = 0.146$	$\epsilon_{emp} = 0.135$

Tabella 4.4: Risultati dell'approccio a scenario con algoritmo di eliminazione random con $\alpha = 0.15$ quando la condizione iniziale è una gaussiana standard oppure è distribuita uniformemente in $[0, 1]$. Caso1 – dinamica continua ridotta, Caso 2 – Eliminazione del moto Browniano, Caso 3 – Eliminazione del processo di Poisson.

Si possono fare le seguenti considerazioni:

1. I livelli di violazione stimati empiricamente sono in tutti i casi coerenti con i parametri α ed ϵ scelti. Nelle tabelle 4.2 e 4.3 si ha $\epsilon_{emp} \approx 0.1$ coerentemente con la scelta $\alpha = 0.1$, mentre nella tabella 4.4 si ha $\epsilon_{emp} \approx 0.15$ coerentemente con la scelta $\alpha = 0.15$.

2. A parità di α l'algoritmo greedy funziona meglio di quello random, anche se il greedy è subottimo. Si vedano a proposito le tabelle 4.2 e 4.3. Nel caso in cui α venga aumentato l'approccio random dà risultati migliori perché corrispondenti ad una violazione empirica maggiore (Tabella 4.4).

3. Il confronto con il metodo basato sulla funzione stocastica di simulazione rivela ancora una volta come questa tecnica porti a risultati troppo conservativi. In tutti i casi si ha infatti $\epsilon_{emp} \ll \epsilon_{desiderato} = 0.25$.

Capitolo 5

Conclusioni e sviluppi futuri

Conclusioni e sviluppi futuri In questa tesi è stato proposto un nuovo approccio per affrontare il problema dell'approssimazione di sistemi ibridi stocastici. L'obiettivo è quello di ottenere un modello semplificato di un sistema dato che riproduca con una certa accuratezza l'andamento di alcune variabili di interesse, e che possa essere quindi utilizzato per la verifica di proprietà di raggiungibilità legate all'evoluzione nel tempo di tali variabili. La qualità dell'approssimazione, misurata tramite la distanza tra i segnali di interesse e quelli riprodotti dal modello semplificato, viene ottimizzata su un insieme di realizzazioni delle sorgenti di incertezza agenti sul sistema (ingressi stocastici e stato iniziale) di probabilità prefissata $1-\epsilon$, con $\epsilon \in (0,1)$ parametro di progetto. Il problema di ottimizzazione chance-constrained risultante è difficile da risolvere in modo esatto,e nella tesi è stata adottata una soluzione approssimata ottenuta mediante il cosiddetto "approccio a scenario". Da un punto di vista algoritmico, questo comporta estrarre un numero finito di realizzazioni e risolvere il problema di ottimizzazione chance-constrained con riferimento ai soli scenari estratti, eliminandone una frazione di circa l'$\epsilon 100\%$, in modo da avere una violazione empirica pari a quella teorica desiderata ϵ. Gli scenari da eliminare vengono scelti uno dopo l'altro, secondo un procedimento "greedy", in modo da ottimizzare l'accuratezza del modello approssimante. Sotto certe ipotesi di convessità, se il numero di realizzazioni estratte viene scelto in modo opportuno allora si possono fornire garanzie sulla qualità della soluzione randomizzata ottenuta al problema chance-constrained di partenza. Un vantaggio importante dell'approccio proposto è che non vengono fatte ipotesi restrittive sul sistema da approssimare, e, in linea di principio, non è nemmeno necessario avere a disposizione un mo-

dello per esso. Bisogna però poter eseguire sul sistema vari esperimenti ed avere accesso tramite misura alle sorgenti di incertezza agenti su di esso. La complessità del progetto di un modello approssimante mediante l'approccio proposto è indipendente dalla complessità del sistema da approssimare e dipende invece da quella della classe di modelli adottata. Più precisamente, la parametrizzazione del modello deve essere scelta in modo da garantire la convessità dell'indice attraverso il quale si valuta la sua accuratezza. In assenza della proprietà di convessità verrebbero infatti meno le garanzie date dalla teoria a scenario sulla soluzione randomizzata al problema di ottimizzazione chance-constrained, ed il problema di ottimizzazione con un numero di scenari finito sarebbe difficile da risolvere perché non convesso. Nella tesi si è scelto di illustrare l'approccio con riferimento ad un sistema appartenente alla classe dei Jump Linear Stochastic System (JLSS). Ciò ha reso possibile il confronto con un metodo alternativo per l'approssimazione di sistemi ibridi stocastici, apparso di recente in letteratura, che utilizza la nozione di Funzione di Simulazione Stocastica (SSF) per derivare i risultati teorici sulla qualità dell'approssimante, e che si riduce alla soluzione di un problema LMI per quanto riguarda gli aspetti algoritmici. I risultati ottenuti hanno rivelato la superiorità dell'approccio proposto e la conservatività di quello basato sulla SSF. A differenza di quest'ultimo, l'approccio proposto può inoltre essere utilizzato per il progetto del modello approssimante e non solo per la valutazione della qualità di un modello dato. Questo è stato evidenziato nel lavoro di tesi con riferimento al problema dell'ottimizzazione della condizione iniziale del modello, sempre con riferimento alla classe dei JLSS.

Per ovviare alla difficoltà di scelta di una parametrizzazione del modello che garantisca la convessità si potrebbe pensare ad una procedura in due stadi in cui dapprima si adotta un metodo – anche basato su di un'euristica – per la costruzione del modello semplificato del sistema ibrido stocastico di interesse, e poi si valuta la sua qualità come approssimante tramite l'approccio basato sulla teoria a scenario proposto in questa tesi. Di particolare interesse sembra a tale riguardo il procedimento di astrazione per sistemi ibridi introdotto in [6]. Da un punto di vista del carico computazionale, si è osservato come il procedimento greedy di eliminazione degli scenari possa essere particolarmente oneroso soprattutto nel caso di ottimizzazione del modello approssimante. Per questo motivo nella tesi sono state proposte

alcune soluzioni sub-ottime per rendere il problema trattabile. Opzioni alternative da investigare sono l'utilizzo di strumenti per il calcolo parallelo, incluso il GPU computing (l'utilizzo di una unità di elaborazione grafica per l'esecuzione di calcoli), nella procedura greedy, quando si valuta quale sia lo scenario che, se eliminato, migliora maggiormente la misura dell'accuratezza del modello approssimante, e l'utilizzo diretto dei risolutori di problemi di ottimizzazione convessa quali SeDuMi senza avvalersi delle interfacce YALMIP/CVX.

Appendice A

Codice MATLAB

A.1 Problema a.1

```
clear all
close all


test=menu('Utilizzo?','Solo calcolo','Calcolo+test');
scelta=menu('Tecnica a scenario:','Continuo ridotto'...
    ,'No browniano','No Poisson');
sceltagraf=menu('Visualizzare grafico?','Sì','No');



A=[-1 -10   0    0    0    0;
    10  -1   0    0    0    0;
    0   0   -2  -20    0    0;
    0   0   20   -1    0    0;
    0   0    0    0   -2    0;
    0   0    0    0    0 -2.5];
F=0.5*[1 0 0 0 1 1;
       0 1 0 0 0 0;
       0 0 1 0 1 1;
       0 0 0 1 0 0;
       1 0 0 0 0 1;
       0 0 1 0 1 0];
C=[0.84 -1.03 1.07  -0.88 0.5    0;
   -0.6 -1.35 -0.26 -0.27  0   -0.5];
R=0.7*eye(6);
n=size(A,1);



conf=0.75;
alph=0.20;
eps=1-conf;
beta=1e-10;
d=1;
fprintf('Calcolo nr con eps=%g%% , alpha=%g%%',...
    eps*100,alph*100)
nr=findN_new(eps,alph,beta,d);

if test==2
nr2=input('Inserire numero realizzazioni di test: ');
```

```
end

T=8; N=T*1000; dt=T/N;
mult = 4; Dt = mult*dt;
L = N/mult;


randn('state',100)
rand('state',100)


lambda=0.5;
p=zeros(nr,10*T);

for jj=1:nr
    p(jj,1)=-log(rand(1))./lambda;
    k=1;
    while p(jj,k)<=T
        p(jj,k+1)=p(jj,k)-log(rand(1))./lambda;
        k=k+1;
    end
end
p=Dt*round(p/Dt);

tic

if scelta==1

    Arid=A(1:4,1:4);
    Frid=0.5*eye(4);
    Crid=C(1:2,1:4);
    Rrid=0.7*eye(4);

    AA=[A,zeros(6,4);
        zeros(4,6),Arid];
    FF=[F,zeros(6,4);
        zeros(4,6),Frid];
    CC=[C -Crid];
    RR=0.7*eye(10);


    XXem = zeros(10,L);
    YYem = zeros(2,L);
    diff=zeros(nr,L+1);

    X10=ones(n,1);
    X20=[1.05;0.71;1.08;0.931];
    XXzero=[X10;X20];

    wait=...
      waitbar(0,'Attendere: Simulazione in corso ...');

    for jj=1:nr
        ii=1;
        XXtemp = XXzero;
        dW = sqrt(dt)*randn(1,N);
        for j = 1:L
            if j*Dt == p(jj,ii)
                ii=ii+1;
                XXem(:,j) = (eye(10)+RR)*XXtemp;
            else
                Winc = sum(dW(mult*(j-1)+1:mult*j));
                XXem(:,j) = XXtemp + AA*XXtemp*Dt ...
                    + FF*XXtemp*Winc;
            end
            XXtemp = XXem(:,j);
        end
        YYem(:,:)=CC*XXem;
```

```
        YYzero=CC*XXzero;
        diff(jj,:)=[YYzero(1)^2+YYzero(2)^2 ...
            YYem(1,:).^2+YYem(2,:).^2];
        waitbar(jj/nr);
    end
    close(wait);
    h=max(diff,[],2);
    h=choppatore(h,4);

toc
end


if scelta==2

    AA=[A,zeros(6,6);
        zeros(6,6),A];
    FF=[F,zeros(6,6);
        zeros(6,6),zeros(6,6)];
    CC=[C -C];
    RR=0.7*eye(12);

    XXem = zeros(2*n,L);
    YYem = zeros(2,L);
    diff=zeros(nr,L+1);

    X10=ones(n,1);
    X20=[0.89;1.15;1.02;0.76;2.02;1.34];
    XXzero=[X10;X20];

    wait=...
      waitbar(0,'Attendere: Simulazione in corso ...');

    for jj=1:nr
        ii=1;
        XXtemp = XXzero;
        dW = sqrt(dt)*randn(1,N);
        for j = 1:L
            if j*Dt == p(jj,ii)
                ii=ii+1;
                XXem(:,j) = (eye(12)+RR)*XXtemp;
            else
                Winc = sum(dW(mult*(j-1)+1:mult*j));
                XXem(1:6,j) = XXtemp(1:6) + ...
                    AA(1:6,1:6)*XXtemp(1:6)*Dt +...
                    FF(1:6,1:6)*XXtemp(1:6)*Winc;
                XXem(7:12,j) = XXtemp(7:12) + ...
                    AA(7:12,7:12)*XXtemp(7:12)*Dt;
            end
            XXtemp = XXem(:,j);
        end
        YYem(:,:)=CC*XXem;
        YYzero=CC*XXzero;
        diff(jj,:)=[YYzero(1)^2+YYzero(2)^2 ...
            YYem(1,:).^2+YYem(2,:).^2];
        waitbar(jj/nr);
    end
    h=max(diff,[],2);
    h=choppatore(h,4);
    close(wait);

toc
end

if scelta==3

    AA=[A,zeros(6,6);
        zeros(6,6),A];
    FF=[F,zeros(6,6);
        zeros(6,6),F];
```

```
    RR=[R zeros(6,6)
        zeros(6,6), zeros(6,6)];
    CC=[C -C];

    XXem = zeros(2*n,L);
    YYem = zeros(2,L);
    diff=zeros(nr,L+1);

    X10=[1 1 1 1 1 1]';
    X20=[1.594;1.773;1.285;1.195;2.29;-0.48];
    XXzero=[X10;X20];
    wait=...
      waitbar(0,'Attendere: Simulazione in corso ...');

    for jj=1:nr
        ii=1;
        XXtemp = XXzero;
        dW = sqrt(dt)*randn(1,N);
        for j = 1:L
            if j*Dt == p(jj,ii)
                ii=ii+1;
                Winc = sum(dW(mult*(j-1)+1:mult*j));
                XXem(1:6,j) = (eye(6)+R)*XXtemp(1:6);

                XXem(7:12,j)= XXtemp(7:12) + ...
                    AA(7:12,7:12)*XXtemp(7:12)*Dt + ...
                    FF(7:12,7:12)*XXtemp(7:12)*Winc;
            else
                Winc = sum(dW(mult*(j-1)+1:mult*j));
                XXem(:,j) = XXtemp + AA*XXtemp*Dt + ...
                    FF*XXtemp*Winc;
            end
            XXtemp = XXem(:,j);
        end
        YYem(:,:)=CC*XXem;
        YYzero=CC*XXzero;
        diff(jj,:)=[YYzero(1)^2+YYzero(2)^2 ...
            YYem(1,:).^2+YYem(2,:).^2];
        waitbar(jj/nr);
    end
    h=max(diff,[],2);
    h=choppatore(h,4);
    close(wait);

toc
end


nn=size(AA,1);
XXzerotr=XXzero';
cvx_begin sdp
cvx_quiet(true)
variable M_pap(nn,nn) symmetric
Q=M_pap*(AA+lambda*RR)+(AA+lambda*RR)'*M_pap+...
    FF'*M_pap*FF+lambda*RR'*M_pap*RR;
minimize XXzerotr*M_pap*XXzero
M_pap-CC'*CC>=0;
Q<=0;
cvx_end
fi0=cvx_optval;
UBpap=1/(1-conf)*fi0;


if sceltagraf==1


    h_iniz=h;
    ColorSet = varycolor(nr/10);
    set(gca,'ColorOrder',ColorSet);
    hold all
    for jj=1:10:nr
```

```
        grafico=plot([0:Dt:T],(diff(jj,:))');
    end


    axis([0 T 0 6])
    hold on
    set(get(get(grafico,'Annotation'),...
        'LegendInformation'),'IconDisplayStyle','off');
    h=sort(h,'descend');
    k=ceil(alph*nr);



    graf(1)=plot([0 T],[h(15) h(15)],'r');
    graf(2)=plot([0 T],[h(k-1) h(k-1)],'b');
    graf(3)=plot([0 T],[UBpap UBpap]);

    axis([0 T 0 h(12)+0.05*h(12)])


  legend(graf,'Upper bound Scenario WC',...
  'Upper Bound Scenario CCP','Upper Bound tramite SFF')

    xlabel('t','FontSize',20)
    ylabel('||Y_S_1 - Y_S_2||^2','FontSize',20,...
        'Rotation',90,'HorizontalAlignment','right')
end


if test==1
k=ceil(alph*nr);
nr_badP=size(find(UBpap<h),1);
h=sort(h,'descend');

fprintf...
('\nScenario: h* senza eliminazione vincoli = %f',h(1));
fprintf...
('\nScenario: h* con eliminazione vincoli = %f', h(k-1));
fprintf...
('\nPappas: bounda il %g%% delle realizzazioni\n\n',...
(nr-nr_badP)/nr*100);
end

if test==2

lambda=0.5;
p=zeros(nr2,10*T);
for jj=1:nr2
    p(jj,1)=-log(rand(1))./lambda;
    k=1;
    while p(jj,k)<=T
        p(jj,k+1)=p(jj,k)-log(rand(1))./lambda;
        k=k+1;
    end
end
p=Dt*round(p/Dt);

if scelta==1
    tic


    XXem = zeros(10,L);
    YYem = zeros(2,L);
    diff=zeros(nr2,L+1);

    X10=ones(n,1);
    X20=X10(1:4);
    XXzero=[X10;X20];
```

```
    wait=...
     waitbar(0,'Attendere: Simulazione in corso ...');

    for jj=1:nr2
        ii=1;
        XXtemp = XXzero;
        dW = sqrt(dt)*randn(1,N);
        for j = 1:L
            if j*Dt == p(jj,ii)
                ii=ii+1;
                XXem(:,j) = (eye(10)+RR)*XXtemp;
            else
                Winc = sum(dW(mult*(j-1)+1:mult*j));
                XXem(:,j) = XXtemp + AA*XXtemp*Dt + ...
                    FF*XXtemp*Winc;
            end
            XXtemp = XXem(:,j);
        end
        YYem(:,:)=CC*XXem;
        YYzero=CC*XXzero;
        diff(jj,:)=[YYzero(1)^2+YYzero(2)^2 ...
            YYem(1,:).^2+YYem(2,:).^2];
        waitbar(jj/nr2);
    end
    close(wait);
    htest=max(diff,[],2);
    htest=choppatore(htest,4);
toc
end

if scelta==2

    tic

    XXem = zeros(2*n,L);
    YYem = zeros(2,L);
    diff=zeros(nr2,L+1);

    X10=ones(n,1);
    X20=X10;
    XXzero=[X10;X20];

    wait=waitbar(0,'Attendere: Simulazione in corso ...');

    for jj=1:nr2
        ii=1;
        XXtemp = XXzero;
        dW = sqrt(dt)*randn(1,N);
        for j = 1:L
            if j*Dt == p(jj,ii)
                ii=ii+1;
                XXem(:,j) = (eye(12)+RR)*XXtemp;
            else
                Winc = sum(dW(mult*(j-1)+1:mult*j));
                XXem(1:6,j) = XXtemp(1:6) + ...
                    AA(1:6,1:6)*XXtemp(1:6)*Dt + ...
                    FF(1:6,1:6)*XXtemp(1:6)*Winc;
                XXem(7:12,j) = XXtemp(7:12) + ...
                    AA(7:12,7:12)*XXtemp(7:12)*Dt;
            end
            XXtemp = XXem(:,j);
        end
        YYem(:,:)=CC*XXem;
        YYzero=CC*XXzero;
        diff(jj,:)=[YYzero(1)^2+YYzero(2)^2 ...
            YYem(1,:).^2+YYem(2,:).^2];
        waitbar(jj/nr2);
    end
    close(wait);
    htest=max(diff,[],2);
```

```
    htest=choppatore(htest,4);

toc
end

if scelta==3

    tic

    XXem = zeros(2*n,L);
    YYem = zeros(2,L);
    diff=zeros(nr2,L+1);

    X10=randn(n,1);
    X20=X10;
    XXzero=[X10;X20];
    wait=waitbar(0,'Attendere: Simulazione in corso ...');

    for jj=1:nr2
        ii=1;
        XXtemp = XXzero;
        dW = sqrt(dt)*randn(1,N);
        for j = 1:L
            if j*Dt == p(jj,ii)
                ii=ii+1;
                Winc = sum(dW(mult*(j-1)+1:mult*j));
                XXem(1:6,j) = (eye(6)+R)*XXtemp(1:6);

                XXem(7:12,j)= XXtemp(7:12) + ...
                    AA(7:12,7:12)*XXtemp(7:12)*Dt + ...
                    FF(7:12,7:12)*XXtemp(7:12)*Winc;
            else
                Winc = sum(dW(mult*(j-1)+1:mult*j));
                XXem(:,j) = XXtemp + AA*XXtemp*Dt + ...
                    FF*XXtemp*Winc;
            end
            XXtemp = XXem(:,j);
        end
        YYem(:,:)=CC*XXem;
        YYzero=CC*XXzero;
        diff(jj,:)=[YYzero(1)^2+YYzero(2)^2 ...
            YYem(1,:).^2+YYem(2,:).^2];
        waitbar(jj/nr2);
    end
    close(wait);
    htest=max(diff,[],2);
    htest=choppatore(htest,4);

toc
end

h=sort(h,'descend');
k=ceil(alph*nr);
nr_badS1=size(find(h(1)<htest),1);
nr_badS2=size(find(h(k-1)<htest),1);
nr_badP=size(find(UBpap<h),1);
fprintf('\n------- Scenario -------')
fprintf...
('\nh(1) con eps=%g%% e alph=%g%% bounda il %g%%')
fprintf('delle realizzazioni',eps*100,alph*100,...
(nr2-nr_badS1)/nr2*100);
fprintf('\nh* con eps=%g%% e alph=%g%% bounda il')
fprintf('%g%% delle realizzazioni',eps*100,alph*100,...
(nr2-nr_badS2)/nr2*100);
fprintf('\n------- Pappas -------')
fprintf('\nBounda il %g%% delle realizzazioni\n',...
    (nr-nr_badP)/nr*100);

end
```

A.2 Problema a.2

```
clear all
close all


greedy=menu('Utilizzo?','Completo (con greedy)',...
    'Solo simulatore per verifica (senza greedy)');
scelta=menu('Tecnica a scenario:',...
    'Continuo ridotto','No browniano','NoPoisson');
distrib=menu('Distribuzione x10?','Normale','Uniforme');
if greedy==1
sceltarandom=menu('Calcolo valore atteso?',...
    'Inserimento manuale','MonteCarlo');
end
risposta=menu('Greedy vero o Eliminazione casuale?',...
    'Greedy vero', 'Eliminazione casuale');


A=[-1 -10   0    0    0    0;
10 -1   0    0    0    0;
0   0   -2 -20    0    0;
0   0   20   -1    0    0;
0   0    0    0   -2    0;
0   0    0    0    0 -2.5];
F=0.5*[1 0 0 0 1 1;
0 1 0 0 0 0;
0 0 1 0 1 1;
0 0 0 1 0 0;
1 0 0 0 0 1;
0 0 1 0 1 0];
C=[0.84 -1.03 1.07  -0.88 0.5     0;
-0.6 -1.35 -0.26 -0.27  0   -0.5];
R=0.7*eye(6);
n=size(A,1);
if scelta==1
nn=2*n-2;
else
nn=2*n;
end


conf=0.75;
alph=0.10;
eps=1-conf;
beta=1e-10;
d=n*(n+1)/2+n+1;
if greedy==1
disp('Utilizzo completo. Calcolo nr con findN');
nr=findN_new(eps,alph,beta,d)
end
if greedy==2
fprintf('\n**********************************\n')
nr=input('Inserire numero realizzazioni: ');
end

T=12; N=12000; dt=T/N;
mult = 1; Dt = mult*dt;
L = N/mult;

randn('state',sum(100.*clock))
rand('state',sum(100.*clock))
```

```
if scelta==1
f=inline('x(1:end-2,:)','x');
else
f=inline('x','x');
end


if greedy==1
if sceltarandom==1
fprintf('Inserire␣E[x10*x10]␣(%u␣x␣%u):␣',n+1,n+1);
valore_attesoX10_all=input('');
fprintf('Inserire␣E[xx0*xx0]␣(%u␣x␣%u):␣',nn,nn);
valore_attesoXX0=input('');
end
if sceltarandom==2
if distrib==1
auxx1=load('valori_attesi.mat',...
    'valore_attesoX10_N_all');
valore_attesoX10_all=auxx1.valore_attesoX10_N_all;
if scelta==1
    auxx2=load('valori_attesi.mat',...
        'valore_attesoXX0_N_10x10');
    valore_attesoXX0=auxx2.valore_attesoXX0_N_10x10;
else
    auxx2=load('valori_attesi.mat',...
        'valore_attesoXX0_N_12x12');
    valore_attesoXX0=auxx2.valore_attesoXX0_N_12x12;
end
end
if distrib==2
auxx1=load('valori_attesi.mat','valore_attesoX10_U_all');
valore_attesoX10_all=auxx1.valore_attesoX10_U_all;
if scelta==1
    auxx2=load('valori_attesi.mat',...
        'valore_attesoXX0_U_10x10');
    valore_attesoXX0=auxx2.valore_attesoXX0_U_10x10;
else
    auxx2=load('valori_attesi.mat',...
        'valore_attesoXX0_U_12x12');
    valore_attesoXX0=auxx2.valore_attesoXX0_U_12x12;
end
end
end
end



lambda=0.5;
p=zeros(nr,10*T);
for jj=1:nr
p(jj,1)=-log(rand(1))./lambda;
k=1;
while p(jj,k)<=T
p(jj,k+1)=p(jj,k)-log(rand(1))./lambda;
k=k+1;
end
end
p=choppatore(p,3);


if scelta==1
tic

Arid=A(1:4,1:4);
Frid=0.5*eye(4);
Crid=C(1:2,1:4);
Rrid=0.7*eye(4);

AA=[A,zeros(6,4);
```

```
zeros(4,6),Arid];
FF=[F,zeros(6,4);
zeros(4,6),Frid];
CC=[C -Crid];
RR=0.7*eye(10);

if distrib==1
X10_matrix=randn(n,nr);
end
if distrib==2
X10_matrix=rand(n,nr);
end
X10_matrix=choppatore(X10_matrix,4);

XXem = zeros(nn,L);
YYem = zeros(2,L);
diff=zeros(nr,L+1);
wait=waitbar(0,'Attendere: Simulazione in corso ...');

for jj=1:nr
ii=1;
X1zero=X10_matrix(:,jj);
X2zero=f(X1zero);
XXzero=[X1zero;X2zero];
XXtemp = XXzero;
dW = sqrt(dt)*randn(1,N);
for j = 1:L
if j*Dt == p(jj,ii)
    ii=ii+1;
    XXem(:,j) = (eye(nn)+RR)*XXtemp;
else
    Winc = sum(dW(mult*(j-1)+1:mult*j));
    XXem(:,j) = XXtemp + AA*XXtemp*Dt + FF*XXtemp*Winc;

end
XXtemp = XXem(:,j);
end
YYem(:,:)=CC*XXem;
YYzero=CC*XXzero;
diff(jj,:)=[YYzero(1)^2+YYzero(2)^2 ...
    YYem(1,:).^2+YYem(2,:).^2];
waitbar(jj/nr);
end
h=max(diff,[],2);
h=choppatore(h,4);
close(wait);

toc

h_iniz=h;

clear diff
clear XXem
clear YY
clear YYem
clear W
clear dW
end

if scelta==2
tic

AA=[A,zeros(6,6);
zeros(6,6),A];
FF=[F,zeros(6,6);
zeros(6,6),zeros(6,6)];
CC=[C -C];
RR=0.7*eye(12);

if distrib==1
X10_matrix=randn(n,nr);
```

```
end
if distrib==2
X10_matrix=rand(n,nr);
end
X10_matrix=choppatore(X10_matrix,4);

XXem = zeros(nn,L);
YYem = zeros(2,L);
diff=zeros(nr,L+1);
wait=waitbar(0,'Attendere: Simulazione in corso ...');

for jj=1:nr
ii=1;
X1zero=X10_matrix(:,jj);
X2zero=f(X1zero);
XXzero=[X1zero;X2zero];
XXtemp = XXzero;
dW = sqrt(dt)*randn(1,N);
for j = 1:L
if j*Dt == p(jj,ii)
    ii=ii+1;
    XXem(:,j) = (eye(nn)+RR)*XXtemp;
else
    Winc = sum(dW(mult*(j-1)+1:mult*j));
    XXem(1:6,j) = XXtemp(1:6) + ...
        AA(1:6,1:6)*XXtemp(1:6)*Dt + ...
        FF(1:6,1:6)*XXtemp(1:6)*Winc;
    XXem(7:12,j) = XXtemp(7:12) + ...
        AA(7:12,7:12)*XXtemp(7:12)*Dt;
end
XXtemp = XXem(:,j);
end
YYem(:,:)=CC*XXem;
YYzero=CC*XXzero;
diff(jj,:)=[YYzero(1)^2+YYzero(2)^2 ...
    YYem(1,:).^2+YYem(2,:).^2];
waitbar(jj/nr);
end
h=max(diff,[],2);
h=choppatore(h,4);
close(wait);

toc

h_iniz=h;

clear diff
clear XXem
clear YY
clear YYem
clear dW
end


if scelta==3
tic

AA=[A,zeros(6,6);
zeros(6,6),A];
FF=[F,zeros(6,6);
zeros(6,6),F];
RR=[R zeros(6,6)
zeros(6,6), zeros(6,6)];
CC=[C -C];

if distrib==1
X10_matrix=randn(n,nr);
end
if distrib==2
X10_matrix=rand(n,nr);
end
```

```
X10_matrix=choppatore(X10_matrix,4);

XXem = zeros(nn,L);
YYem = zeros(2,L);
diff=zeros(nr,L+1);
wait=waitbar(0,'Attendere: Simulazione in corso ...');

for jj=1:nr
ii=1;
X1zero=X10_matrix(:,jj);
X2zero=f(X1zero);
XXzero=[X1zero;X2zero];
XXtemp = XXzero;
dW = sqrt(dt)*randn(1,N);
for j = 1:L
if j*Dt == p(jj,ii)
    ii=ii+1;
    Winc = sum(dW(mult*(j-1)+1:mult*j));
    XXem(1:6,j) = (eye(n)+R)*XXtemp(1:6);
    XXem(7:12,j)= XXtemp(7:12) + ...
        AA(7:12,7:12)*XXtemp(7:12)*Dt + ...
        FF(7:12,7:12)*XXtemp(7:12)*Winc;
else
    Winc = sum(dW(mult*(j-1)+1:mult*j));
    XXem(:,j) = XXtemp + AA*XXtemp*Dt + FF*XXtemp*Winc;
end
XXtemp = XXem(:,j);
end
YYem(:,:)=CC*XXem;
YYzero=CC*XXzero;
diff(jj,:)=[YYzero(1)^2+YYzero(2)^2 ...
    YYem(1,:).^2+YYem(2,:).^2];
waitbar(jj/nr);
end
h=max(diff,[],2);
h=choppatore(h,4);
close(wait);

toc

h_iniz=h;

clear diff
clear XXem
clear YY
clear YYem
clear dW
end


if greedy==1

k=ceil(alph*nr);
num_elim=0;
num_viol_elim=0;
numtot_viol_elim=0;
ottimo=1000*ones(1,d+5);
fi=zeros(nr,d+5);
h_iniz_riord=h_iniz;
X10_matrix=[X10_matrix; ones(1,nr)];
X10_iniz=X10_matrix;
pos_fin=zeros(1,ceil(alph*nr));
pos_finM=zeros(30,ceil(alph*nr));

tic

cvx_begin sdp
cvx_quiet(true)
variable M_all(n+1,n+1) symmetric
minimize trace(M_all*valore_attesoX10_all)
M_all>=0;
```

```
for jj = 1:nr
X10_matrix(:,jj)'*M_all*X10_matrix(:,jj) >= h(jj);
end
cvx_end

M_all_wc1=M_all;
ottimoScenario_allWC=cvx_optval;

fi(:,1)=diag(X10_matrix'*M_all*X10_matrix);
fi=choppatore(fi,4);
pos_att=find(fi(:,1)==h);
a=cvx_optval;


ciclo=0;
flag=1;
while flag
ciclo=ciclo+1;
while (num_elim<ceil(alph*nr)&&ciclo==1)||...
        (num_viol_elim==0&&ciclo>1)
tic
if risposta==1
    for i=1:numel(pos_att)
        X10_rid=X10_matrix;
        h_rid=h;
        X10_rid(:,pos_att(i))=zeros(n+1,1);
        h_rid(pos_att(i))=0;

        cvx_begin sdp
        cvx_quiet(true)
        variable M_all(n+1,n+1) symmetric
        minimize trace(M_all*valore_attesoX10_all)
        M_all>=0;
        for jj = 1:nr-num_elim-numtot_viol_elim
            X10_rid(:,jj)'*M_all*X10_rid(:,jj) ...
                >= h_rid(jj)
        end
        cvx_end
        fi(:,i)=diag(X10_rid'*M_all*X10_rid);
        ottimo(i)=cvx_optval;
    end
    toc
    [a b]=min(ottimo);
    X10_matrix=[X10_matrix(:,1:pos_att(b)-1),...
        X10_matrix(:,pos_att(b)+1:nr),zeros(n+1,1)];
    h=[h(1:pos_att(b)-1);h(pos_att(b)+1:nr);0];
    fi(:,b)=[fi(1:pos_att(b)-1,b);...
        fi(pos_att(b)+1:nr,b);0];
    fi(:,b)=choppatore(fi(:,b),4);
    if ciclo==1
        num_elim=num_elim+1;
        pos_fin(num_elim)=pos_att(b);
    else
        num_viol_elim=num_viol_elim+1;
        pos_fin=pos_att(b);
    end
end
if risposta==2
    b=randi(numel(pos_att));
    X10_matrix=[X10_matrix(:,1:pos_att(b)-1),...
        X10_matrix(:,pos_att(b)+1:nr),zeros(n+1,1)];
    h=[h(1:pos_att(b)-1);h(pos_att(b)+1:nr);0];
    if ciclo==1
        num_elim=num_elim+1;
        pos_fin(num_elim)=pos_att(b);
    else
        num_viol_elim=num_viol_elim+1;
        pos_fin=pos_att(b);
    end
    cvx_begin sdp
    cvx_quiet(true)
```

```
    variable M_all(n+1,n+1) symmetric
    minimize trace(M_all*valore_attesoX10_all)
    M_all>=0;
    for jj = 1:nr-num_elim-numtot_viol_elim
    X10_matrix(:,jj)'*M_all*X10_matrix(:,jj) >= h(jj);
    end
    cvx_end
    a=cvx_optval;
    fi(:,b)=diag(X10_matrix'*M_all*X10_matrix);
    fi(:,b)=choppatore(fi(:,b),4);
end
toc

if ciclo==1
    pos_att=find(fi(:,b)==h&h~=0&fi(:,b)~=0);

else
    numtot_viol_elim=numtot_viol_elim+1;

end
end

flag=0;

for s=1:numel(pos_fin)
h_iniz_riord=[h_iniz_riord(1:pos_fin(s)-1);...
h_iniz_riord(pos_fin(s)+1:end);h_iniz_riord(pos_fin(s))];
end
pos_finM(ciclo,1:size(pos_fin,2))=pos_fin;

clear pos_fin

if sum(fi(:,b)<h_iniz_riord)~=ceil(alph*nr)+...
        num_viol_elim
fprintf('\nATTENZIONE:␣Ci␣sono␣state␣%u␣violazioni\n',...
    sum(fi(:,b)<h_iniz_riord)-ceil(alph*nr)...
    -num_viol_elim)
pos_att=find(fi(:,b)<h_iniz_riord & fi(:,b)~=0);
fprintf('Posizione␣violazioni:␣%s␣\n', ...
    num2str(pos_att'))

num_viol_elim=0;
flag=1;
end
end

fprintf('Tempo␣totale␣greedy:␣%g\n', toc);
ottimoScenario_all=a;


cvx_begin sdp
cvx_quiet(true)
variable M_pap(nn,nn) symmetric
minimize trace(M_pap*valore_attesoXX0)
Q=M_pap*(AA+lambda*RR)+(AA+lambda*RR)'*M_pap+...
    FF'*M_pap*FF+lambda*RR'*M_pap*RR;
M_pap-CC'*CC>=0;
Q<=0;
cvx_end

ottimoPappas=1/eps*cvx_optval;

end


if greedy==2


if risposta==1
if scelta==1&&distrib==1
z=load('2a_cr_e25_a15_N_gree.mat','scelta','eps',...
'alph','M_all','ottimoScenario_all',...
```

```
'ottimoPappas','M_pap');

end
if scelta==1&&distrib==2
z=load('2a_cr_e25_a10_U_gree.mat','scelta','eps',...
'alph','M_all','ottimoScenario_all',...
'ottimoPappas','M_pap');

end
if scelta==2&&distrib==1
z=load('2a_nb_e25_a10_N_gree.mat','scelta','eps',...
'alph','M_all','ottimoScenario_all',...
'ottimoPappas','M_pap');

end
if scelta==2&&distrib==2
z=load('2a_nb_e25_a10_U_gree.mat','scelta','eps',...
'alph','M_all','ottimoScenario_all',...
'ottimoPappas','M_pap');

end
if scelta==3&&distrib==1
z=load('2a_np_e25_a10_N_gree.mat','scelta','eps',...
'alph','M_all','ottimoScenario_all',...
'ottimoPappas','M_pap');

end
if scelta==3&&distrib==2
z=load('2a_np_e25_a10_U_gree.mat','scelta','eps',...
'alph','M_all','ottimoScenario_all',...
'ottimoPappas','M_pap');

end
end

if risposta==2
if scelta==1&&distrib==1
z=load('2a_cr_e25_a15_N.mat','scelta','eps',...
'alph','M_all','ottimoScenario_all',...
'ottimoPappas','M_pap');

end
if scelta==1&&distrib==2
z=load('2a_cr_e25_a15_U.mat','scelta','eps',...
'alph','M_all','ottimoScenario_all',...
'ottimoPappas','M_pap');

end
if scelta==2&&distrib==1
z=load('2a_nb_e25_a15_N.mat','scelta','eps',...
'alph','M_all','ottimoScenario_all',...
'ottimoPappas','M_pap');

end
if scelta==2&&distrib==2
z=load('2a_nb_e25_a15_U.mat','scelta','eps',...
'alph','M_all','ottimoScenario_all',...
'ottimoPappas','M_pap');

end
if scelta==3&&distrib==1
z=load('2a_np_e25_a15_N.mat','scelta','eps',...
'alph','M_all','ottimoScenario_all',...
'ottimoPappas','M_pap');

end
if scelta==3&&distrib==2
z=load('2a_np_e25_a15_U.mat','scelta','eps',...
'alph','M_all','ottimoScenario_all',...
'ottimoPappas','M_pap');

end
end
```

```
fi_all=diag([X10_matrix; ones(1,nr)]'*z.M_all*...
    [X10_matrix; ones(1,nr)]);
nr_badS=size(find(fi_all<h_iniz),1);
fiPappas=1/(z.eps)*diag([X10_matrix;f(X10_matrix)]'...
    *z.M_pap*[X10_matrix;f(X10_matrix)]);
nr_badP1=size(find(fiPappas<h_iniz),1);
fprintf(z.ottimoScenario_all,(nr-nr_badS)/nr*100);

fprintf(z.ottimoPappas,(nr-nr_badP1)/nr*100);

fprintf('**************************************\n\n')
end
```

Bibliografia

[1] A. Abate, J.P. Katoen, J. Lygeros, M. Prandini, "Approximate model checking of stochastic hybrid systems", *European Journal of Control*, vol. 16(6): 624-641, 2010.

[2] A. Agung Julius, G. Pappas, "Approximations of Stochastic Hybrid Systems", *IEEE Transactions on Automatic Control*, vol.54, 6:1193-1203, 2009.

[3] A. Prèkopa, "Stochastic programming", Kluwer, Boston, MT, USA, 1995.

[4] C. G. Cassandras, J. Lygeros, "Stochastic Hybrid Systems", *Taylor & Francis Group,* 2007.

[5] D. J. Higham, "An Algorithmic Introduction to Numerical Simulation of Stochastic Differential Equations", *SIAM review*, vol.43, No. 3, pp. 525-546

[6] E. Mazzi, A. SangiovanniVincentelli, A. Balluchi, A. Bicchi, "Hybrid system model reduction", In Proc. 47th IEEE Conf. on Decision and Control, pages 227–232, Cancun, Mexico, 2008.

[7] G. Calafiore, M. C. Campi, "The scenario approach to robust control design", *IEEE Transactions on Automatic Control*, 51(5):742-753, 2006.

[8] H. A. P. Blom, J. Lygeros, "Stochastic Hybrid Systems", *Springer,* 2006.

[9] "Handbook of hybrid systems control theory, tools, applications", editor J. Lunze, F. Lamnabhi-Lagarrigue

[10] J. Löfberg, "YALMIP: A toolbox for modeling and optimization in MATLAB". Disponibile: http://users.isy.liu.se/johanl/yalmip/

[11] J. Lygeros, M. Prandini, "Stochastic Hybrid Systems: A Powerful Framework for Complex, Large Scale Applications*", European Journal of Control*, vol. 16(6):583-594, 2010.

[12] M. C. Campi, S. Garatti, "Variable Robustness Control: Principles and Algorithms", *19th International Symposium on Mathematical Theory of Networks and Systems (MTNS)*, Budapest, Hungary, 2010.

[13] M. C. Campi, S. Garatti, "A Sampling-and-Discarding Approach to Chance-Constrained Optimization: Feasibility and Optimality", *Journal of Optimization Theory and Applications* 148(2):257-280, 2011.

[14] M. C. Campi, S. Garatti, M.Prandini, "The Scenario Approach for Systems and Control Design", *Annual Reviews in Control* 33(2):149-157, 2009.

[15] S. Boyd, M. C. Grant, CVX–MATLAB Software for Disciplined Convex Programming. Disponibile: http://cvxr.com/cvx/

[16] S. Boyd, L. El Ghaoui, E. Feron, V. Balakrishnan, "Linear Matrix Inequalities in System and Control Theory", SIAM, 1994.