POLITECNICO DI MILANO

FACOLTA' DI INGEGNERIA DELL'INFORMAZIONE

Corso di Laurea in Ingegneria dell'Automazione

LFT-Based Nonlinear MPC

Relatore: Prof. Gianni Ferretti

Correlatore: Ing. Alessandro Della Bona

Tesi di laurea di:

Bossi Marco

Matr. 800899

Anno Accademico 2013 - 2014

Ai miei genitori

Sommario

In questi ultimi anni, le tecniche Model Predictive Control (MPC), originariamente nate per effettuare controllo multivariabile di impianti chimici, sono sempre più utilizzate per applicazioni di ogni tipo, anche in condizione di vincoli real-time molto stringenti. Di conseguenza si richiedono, per la banda del controllore, specifiche di frequenza sempre più stringenti. L'obiettivo di questo elaborato è lo sviluppo e l'applicazione ad un problema reale di un nuovo metodo di controllo MPC non lineare, utilizzando un approccio basato su una modellazione Linear Fractional Transform (LFT).
Nel primo capitolo si presenta lo Stato dell'arte del controllo MPC non lineare e della rappresentazione LFT; in seguito si mostra la forma LFT per sistemi non lineari e come essa possa essere utilizzata per problemi di identificazione parametrica. Nel terzo capitolo si presenta il metodo utilizzato in seguito per eseguire un'ottimizzazione non lineare, essenziale per la ricerca del minimo della cifra di merito nel controllo ottimo; successivamente si ricava la forma LFT che può essere utilizzata nel problema di controllo ottimo sviluppato, e si implementa il controllo MPC a partire dagli stadi di simulazione LFT ricavati. Infine, si presentano rapidamente, a livello teorico, i metodi utilizzati per rendere più efficiente il calcolo delle matrici necessarie al controllo, si estende il lavoro al caso MIMO e si applica il controllo MPC non lineare così ottenuto ad un problema di controllo di traiettoria per un veicolo.

Parole chiave: Linear Fractional Transformation; LFT; Ottimizzazione; Controllo; OCP; Simulazione; Model Predictive Control; MPC

Abstract

Nowadays the Model Predictive Control (MPC) techniques, usually designed for multi-variable control on chemical plants, has been more and more used into all purpose applications, even requiring faster and faster computational time. Therefore it is required always to narrow frequencies' specifications. This work takes as goal to develop and apply to a real problem a new nonlinear MPC control method, using a Linear Fractional Transform (LFT) modeling based approach.
In the first chapter, State of the Art referring to both nonlinear MPC control and LFT modeling is presented; then LFT modeling for nonlinear systems is introduced and it's shown how it can be used for parametric identification problems. In the third chapter we aim to present the method for nonlinear optimization we use later in this work. Nonlinear optimization is indeed necessary to find the minimum of the merit function in optimal control. Then LFT shape to be used in developed optimal control is obtained and nonlinear MPC control is developed, starting from LFT simulation shapes obtained. Finally, we rapidly present, from theoretical point of view, methods used for making MPC matrix calculations more efficient, we extend this work to MIMO cases and we apply nonlinear MPC previously designed to a optimal trajectory control for a vehicle.

Keywords: Linear Fractional Transformation; LFT; Optimization; Control; OCP; Simulation; Model Predictive Control; MPC

Indice

Capitolo 1

Introduzione

In questo primo capitolo si introduce il problema che si intende risolvere in questo elaborato. Si introducono quindi il controllo MPC, l'ottimizzazione non lineare e il problema MPC non lineare.

Negli ultimi anni il controllo MPC, inizialmente sviluppato per impianti chimici ed industriali di grosso calibro ma a ridotta frequenza di campionamento, ha trovato via via applicazione nel controllo di sistemi con dinamiche sempre più veloci. La conseguenza di questa tendenza è la crescente richiesta di una velocità di risoluzione dei problemi di ottimo annessi tale da consentire applicazioni di tipo "hard real-time".
Per risolvere questo problema, nel tempo, si sono utilizzati vari approcci. Questo elaborato si pone come obiettivo quello di costruire un nuovo strumento di controllo MPC non lineare che, sfruttando le potenzialità rappresentative della forma LFT, consenta di ottenere una velocità di calcolo tale da permettere l'esecuzione in tempo reale anche su sistemi dove attualmente il controllo MPC fatica ad affermarsi.

In questo elaborato si affronta il problema da un punto di vista per lo più teorico, analizzando la forma LFT, studiandone le caratteristiche e le proprietà, per poi sfruttarle nell'implementazione di metodi e strumenti di calcolo efficaci. Il fine ultimo è quello di, dato un sistema rappresentato in forma LFT, ottenere vantaggi significativi nell'esecuzione del controllo MPC.
Si inizia fornendo una panoramica dello Stato dell'Arte sulla velocizzazione delle tecniche preesistenti per problemi ad alta frequenza e presentando i metodi di ottimizzazione non lineare annessi. Si prosegue con la presentazione di metodi di simulazione di sistemi non lineari avvicinando via via il lettore alla forma LFT generica di sistemi non lineari; si spiegano quali vantaggi porti questa rappresentazione nell'ambito dell'identificazione di parametri incerti. Di seguito si introducono una serie di soluzioni per il calcolo e l'ottimizzazione del controllo ottimo, basati sulle proprietà della forma LFT e ispirate al procedimento di identificazione analizzato. Nel capitolo seguente si scende più nel dettaglio, derivando dalla forma LFT i termini necessari a risolvere i problemi di ottimo. Successiva-

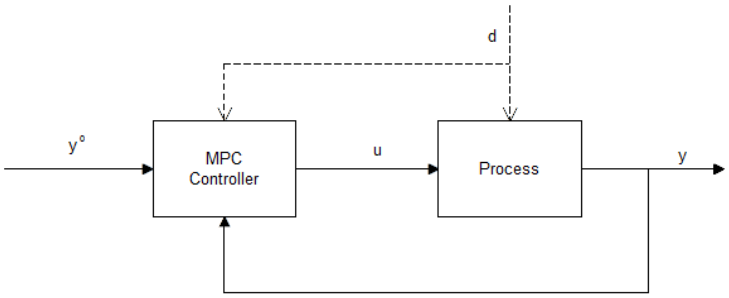

Figura 1.1: Schema di base MPC

mente si presentano alcuni metodi utilizzati per l'aumento dell'efficienza (come l'adozione dei termini comuni, il calcolo e l'utilizzo delle matrici di sparsità del sistema ecc.) e si esegue un esperimento a titolo di test dimostrativo della fattibilità del metodo. Come caso specifico si sceglie un modello di dinamica laterale di un veicolo, chiamato ad inseguire una traiettoria pre-determinata. I risultati ottenuti in questo esperimento mostrano come l'approccio sia concreto e ponga le basi per lo sviluppo di un toolbox in questo campo.

1.1 Il controllo MPC

Il controllo MPC - Model Predictive Control -, secondo una recente analisi che si può trovare in [7], è considerato di uso frequente per oltre l'80% dei processi industriali. Si inizia dallo schema di controllo illustrato in Figura 1.1, dal quale si può vedere come il controllore MPC riceva in ingresso dal sistema e dall'ambiente:

- l'uscita obiettivo di regolazione (o da inseguire) y^o
- l'uscita misurata del sistema sotto controllo y
- l'eventuale disturbo d (se misurato)

Il controllo MPC è il risultato dell'applicazione iterativa del problema del controllo ottimo (OCP - Optimal Control Problem). Più nel dettaglio, ad ogni istante di tempo si fissa un orizzonte (detto Orizzonte di Predizione), e al suo interno si determina la legge di controllo a spesa minima tale da minimizzare la distanza tra y ed y^o. I vincoli contrastanti su spesa di controllo e inseguimento del set-point danno origine ad un problema convesso di cui l'ottimo è la soluzione solitamente ricercata (da qui Optimal Control Problem). Una volta risolto l'OCP, si attua il controllo applicando il primo campione di u, si trasla in avanti la finestra di ottimizzazione del tempo relativo al campione applicato, e si ripete la risoluzione del nuovo OCP così ottenuto.

Il controllo ottimo (OCP) prevede la risoluzione ad ogni istante del problema così definito:

$$\begin{aligned} \min_{u} \quad & \mathrm{J^o}(y, y^o, u, u^o) \\ \text{subject to} \quad & \dot{x} = f(x, p, u) \\ & y = g(x, p, u) \\ & u \in U, \\ & x \in X \end{aligned}$$

dove $u \in U$ sono gli ingressi, $x \in X$ sono i valori dello stato del sistema, p i parametri del sistema ed y e y^o sono le uscite misurate e quelle desiderate. Le funzioni f e g, dipendenti dallo stato del sistema, dagli ingressi e dai parametri, rappresentano il sistema sotto controllo. La funzione J^o è detta *cifra di merito* ed è di solito quadratica, con espressione:

$$J^o = \int_{\tau=0}^{\tau=T_p} \left(\mathbf{y}(\tau) - \mathbf{y^o}(\tau)\right)^T Q \left(\mathbf{y}(\tau) - \mathbf{y^o}(\tau)\right) + \mathbf{u}(\tau)^T R \mathbf{u}(\tau) d\tau$$

Ad ogni ciclo di OCP, il controllore MPC esegue una simulazione del sistema sotto controllo, basata su un modello del sistema stesso, tramite il quale ottiene i valori di predizione dell'uscita all'interno dell'Orizzonte di Predizione, da cui è possibile calcolare gli ingressi necessari per la minimizzazione rispetto all'uscita desiderata. Nel caso di sistema sotto controllo sia lineare, orizzonte infinito e cifra di merito quadratica (controllo ottimo LQ_∞), la soluzione del problema si ottiene risolvendo l'equazione stazionaria di Riccati.

1.2 Il controllo MPC non lineare

Nel caso in cui il sistema sotto controllo sia non lineare, cioè il caso che si vuole esaminare in questo elaborato, in generale non esiste la soluzione in forma chiusa per l'integrazione del sistema (e quindi per la predizione dell'uscita). Da ciò ne consegue anche l'impossibilità di trovare il minimo della funzione J^o per via analitica. Per il calcolo della simulazione e dell'ottimo, si devono dunque utilizzare strumenti numerici per la simulazione e l'ottimizzazione di sistemi non lineari. Invece di effettuare semplicemente la valutazione della formula esplicita dell'ottimo come nel caso lineare, è necessario, per il controllo MPC non lineare (abbreviato anche in NMPC - Nonlinear Model Predictive Control), l'utilizzo di metodi di ricerca che, naturalmente, riducono drasticamente l'efficienza computazionale del controllore.

1.3 L'ottimizzazione non lineare

Si definisce problema di ottimizzazione non lineare il seguente problema matematico:

$$\begin{aligned} \min_{x} \quad & \mathrm{f}(x) \\ \text{subject to} \quad & g(x) = 0, \end{aligned}$$

In cui si minimizza $f(x)$ non lineare rispetto ad x, sotto condizioni di vincolo (anch'esse non lineari) $g(x) = 0$. L'ottimizzazione non lineare è fondamentale per l'esecuzione corretta ed efficiente di qualsiasi controllo ottimo, e specificatamente è vitale per l'esecuzione del controllo NMPC. Nello Stato dell'Arte (si veda la sezione 2.2) si mostrano alcuni metodi costruiti per l'esecuzione dell'ottimizzazione non lineare, utilizzabili o utilizzati per il controllo NMPC. Nel Capitolo 4 si mostra il metodo utilizzato per la costruzione dell'algoritmo di ottimizzazione non lineare utilizzato per costruire il controllore.

Capitolo 2

Stato dell'Arte

In questo capitolo si presenta lo Stato dell'Arte del controllo MPC, con l'obiettivo di dare una panoramica dei metodi correntemente utilizzati per eseguire il controllo MPC non lineare, e relative varianti per migliorare l'efficienza computazionale del controllore. Si mostrano inoltre gli algoritmi esistenti per la ricerca dell'ottimo, alcuni dei quali si utilizzano nel corso di questo elaborato. Infine si affronta brevemente il problema della discretizzazione e si mostrano alcuni metodi per l'integrazione numerica che possono essere utilizzati.

2.1 Metodi di controllo MPC non lineare

Prima di tutto si presentano i metodi esistenti per il controllo MPC non lineare, con l'obiettivo di mostrare quali strategie sono utilizzate per ridurre il tempo computazionale del classico controllo MPC.

2.1.1 Il controllo MPC

Il controllo MPC è attualmente il metodo più avanzato, a livello concettuale, per effettuare il controllo di un sistema mediante delle variabili manipolabili e una misura in retroazione.
Si configura essenzialmente come la ripetizione di un problema di controllo ottimo nel tempo. In particolare:

- si decide un orizzonte temporale, detto Prediction Horizon, su cui eseguire il controllo ottimo ad ogni passo;
- si esegue il controllo ottimo dall'istante corrente all'istante di termine del Prediction Horizon;
- si ottiene la sequenza ottima da applicare nell'orizzonte di predizione;

- si applica il primo elemento del vettore di variabili di controllo ottenute e si sposta di un istante la finestra del Prediction Horizon; quindi si ricomincia il ciclo.

Solitamente, il controllo MPC lineare configura un problema di controllo ottimo con una cifra di merito quadratica, che quindi possiede, all'interno della zona di ammissibilità delle soluzioni, una soluzione unica e facilmente calcolabile con una formula esatta.
Nel caso non lineare, il controllo MPC manifesta il problema aggiuntivo che l'esecuzione dell'ottimizzazione richiesta dal controllo ottimo deve essere eseguita su un sistema non lineare, con una cifra di merito che non è più strettamente quadratica che può avere più soluzioni di ottimo locale. Tale soluzione deve essere trovata, pertanto, con metodi di ottimizzazione non lineare vincolata, che trattiamo nel paragrafo seguente. Inoltre, la simulazione del sistema per ottenere i valori di predizione non è più ottenibile con un integrazione informa chiusa ma deve essere ottenuta per via numerica.

2.1.2 Metodi MPC non lineari

Per risolvere questo problema si sono utilizzati nel tempo svariati metodi. Di seguito presentiamo quelli più significativi, prima con un approccio generale, ed in seguito con applicazioni mirate all'esempio di cui al capitolo 7. Il problema generico di MPC non lineare può essere rappresentato dalle equazioni:

$$\begin{aligned} \min_{\overline{u}(\cdot)} \quad & \mathrm{J}\left(x(t),\, \overline{u}(\cdot)\right) \\ \text{subject to} \quad & \dot{\overline{x}}(\tau) = f\left(\overline{x}(\tau), \overline{u}(\tau)\right), \\ & \overline{u}(\tau) = \overline{u}(t+T_c), \forall \tau \in [t+T_c, t+T_p],, \\ & \overline{u}(\tau) \in U, \forall \tau \in [t, t+T_c], \\ & \overline{x}(\tau) \in X, \forall \tau \in [t, t+T_p]. \end{aligned} \tag{2.1}$$

Con $J\left(x(t), \overline{u}(\cdot)\right)$ funzione obiettivo di genere quadratico definita come:

$$J\left(x(t), \overline{u}(\cdot)\right) \triangleq \int_t^{t+T_p} (x - x^o)^T Q (x - x^o) + (u - u^o)^T R (u - u^o) \tag{2.2}$$

Si tratta, come già detto, di un problema di ottimizzazione non lineare su orizzonte temporale mobile. La sostituzione di soluzioni analitiche con metodi numerici permette:

- L'uso diretto di modelli di predizione non-lineari
- L'introduzione di vincoli espliciti sulle variabili di stato, di uscita e di controllo

- La risoluzione on-line il problema di controllo ottimo all'interno dell'orizzonte

Un'introduzione al controllo MPC non lineare può essere trovata in [4], con l'introduzione matematica al controllo NMPC ed alcune tecniche di controllo MPC che riprendiamo nella sottosezione 2.1.2.

Metodi NMPC in genere

I metodi utilizzati per risolvere il problema MPC non lineare sono suddivisibili nelle classi:

- basati sull'uso di penalty function
- basati su linearizzazione di sottoproblemi e controllo robusto
- basati su minimizzazione non completa
- basati su suddivisione online e offline del problema
- basati su ricerca lineare dell'ottimo

Tutti i metodi presentati hanno come caratteristica quella di ridurre la complessità del problema, che non sarebbe altrimenti risolvibile in tempo reale (soprattutto al crescere del numero di variabili di stato).
I metodi con penalty function, come ad esempio [9] aggiungono un termine

$$\overline{x}(t + T_P) = 0 \tag{2.3}$$

all'insieme dei vincoli per forzare il controllo ad avere sicura stabilità in retroazione.

I metodi basati sulla linearizzazione di sottoproblemi per controllo robusto, come presentati, ad esempio, in[19], utilizzano lo schema standard di MPC, ma hanno la caratteristica di valutare la funzione obiettivo non utilizzando i valori nominali di stato e ingresso, ma utilizzando valori perturbati che collocano la funzione obiettivo all'interno di una regione D. Di conseguenza si risolve online un problema non più di minimizzazione, bensì di minimo/massimo, con la funzione obiettivo:

$$\underset{\overline{u}(\cdot)}{min} \underset{\Delta \in D}{max} \int_t^{t+T_p} F\left(\overline{x}(\tau), \overline{u}(\tau)\right) d\tau \tag{2.4}$$

Estremizzando questo approccio si può anche ottenere (in casi in cui le nonlinearità siano sufficientemente lievi) un risultato molto importante: quello cioè di poter utilizzare un modello nominale lineare, a patto di ampliare sufficientemente il campo di incertezza.
Sempre del gruppo con linearizzazione fa parte il metodo presentato in [22]. In questo caso il problema di ottimizzazione viene mantenuto convesso e quadratico

utilizzando istante per istante una linearizzazione del sistema sotto controllo, ed effettuando un'ottimizzazione non vincolata su una cifra di merito modificata con l'aggiunta di una penalty function in grado di introdurre i vincoli. Si utilizza anche il vincolo 2.1.2 per ottenere un sistema di controllo MPC in grado di eseguire un controllo di buona qualità nella risoluzione di vincoli anolonomi.

Un discorso a parte lo meritano gli algoritmi MPC che attuano una minimizzazione non completa. L'idea di fondo è di velocizzare il controllo MPC riducendo il tempo impiegato per l'ottimizzazione non lineare. Tramite un'analisi dei dati ci si accorge che la maggior parte delle volte il primo passo di ottimizzazione è già sufficiente ad ottenere una sensibile riduzione del valore della cifra di merito. Di conseguenza molti metodi NMPC si limitano ad eseguire un passo solo di ottimizzazione, considerandolo già più che sufficiente per il conseguimento dell'obiettivo di controllo. Nonostante non ci siano garanzie in tal senso, è comunque una pratica molto comune in letteratura, che consente di velocizzare moltissimo l'esecuzione del controllo MPC.

Un algoritmo particolare per l'esecuzione di NMPC è costruito in [3]. In questo algoritmo si divide il problema di controllo in due parti. Il problema di controllo "in avanti" viene risolto ad ogni istante in modo online. Tale problema comporta uno sforzo computazionale piuttosto ridotto, e quindi permette di essere risolto direttamente online. Il problema di controllo in retroazione, invece, è costruito per mantenere la deviazione causata dal controllo in avanti all'interno di una predeterminata tolleranza, e viene risolto offline per costruire una legge di controllo stabilizzante. Ciò consente di rilassare i vincoli sullo stato iniziale del problema. L'algoritmo così ottenuto è robusto e stabilizzante.

Un altro algoritmo che divide la soluzione in online ed offline si trova in [31]. Qui l'autore effettua una minimizzazione non completa del problema di controllo ottimo, calcolando soltanto il primo dei valori di controllo nell'orizzonte di controllo (cioé l'unico valore che viene effettivamente applicato). Si costruisce (offline) una serie di campi in cui la variabile di controllo può trovarsi, partendo dai valori nominali. Infine (online) si ottimizza con approssimanti lineari del sistema all'interno dei campi pre-determinati. In questo modo l'ottimizzazione è un problema LQ, risolvibile tramite formula esplicita.

Infine si trovano metodi NMPC con una ricerca lineare dell'ottimo. Un esempio si può trovare in [18]. In questo caso, gli autori partono da una iterazione Newton-Rapson applicata ad un controllo NMPC, e la modificano per applicarla alle variabili primali del problema di ottimizzazione. Così facendo, garantiscono la risolvibilità del problema di ottimizzazione e l'uso del metodo di Newton-Rapson che consente una rapida convergenza della soluzione ad un ottimo ammissibile.

Metodi NMPC per controllo di traiettoria

In questo sottoparagrafo si mostrano alcuni metodi NMPC costruiti per essere applicati direttamente al problema nel nostro esempio (controllo di traiettoria di

un veicolo su ruote). Una review dei possibili controlli su sistemi anolonomi si può trovare in [28].

I metodi analizzati in questo lavoro si possono suddividere nelle categorie seguenti:

- utilizzo di un sistema Lineare Tempo Variante
- MPC basato su ottimizzazione "Gradient Descent"
- eliminazione del tempo nella funzione di costo
- uso di variabili di "slack" per rilasciare i vincoli sul controllo

Un metodo MPC applicato al controllo di robot mobili con vincoli anolonomi[1] è presentato in [15], dove si risolve il problema di controllo in anello aperto, ad ogni istante di esecuzione dell'algoritmo. Il sistema utilizzato permette di trattare i vincoli anolonomi, sviluppando una funzione di Lyapunov adatta e ottenendo quindi un metodo robusto di controllo non lineare. Per migliorare il tempo computazionale si utilizza una inizializzazione di tipo "hot start", che consente di utilizzare valori di primo tentativo per la risoluzione dei vincoli che siano già vicini al risultato ottimo che si vuole ottenere.

L'utilizzo di sistemi Lineari Tempo Varianti per MPC è tipico dei problemi di controllo di traiettoria, che sono intrinsecamente non-lineari ma possono essere visti come lineari tempo varianti, a patto di ridurre a sufficienza il passo temporale di integrazione della soluzione. Molti algoritmi MPC per controllo di varie parti del veicolo, dal controllo del carico laterale (si veda [17]), al controllo laterale dello slip (si veda [23]), alla pianificazione di traiettoria (si veda [21]), utilizzano questo approccio combinato con varie tecniche di ottimizzazione, che riprendiamo nel paragrafo successivo.

Il controllo MPC basato su ottimizzazione "Gradient Descent" utilizza come ottimizzatore il solo metodo del Gradiente, come è descritto in 4.1.2. Questo metodo è sfruttato in molti algoritmi, specialmente in quelli ad ottimizzazione non completa visti nel paragrafo 2.1.2.

Alcuni metodi di controllo NMPC lavorano ignorando i vincoli sul tempo della traiettoria. In particolare questi algoritmi sono utilizzati per risolvere problemi di parcheggio, dove la velocità di spostamento non è prioritaria, mentre è molto più importante l'inseguimento corretto della traiettoria del sistema. Un esempio si può trovare in [27]. Eliminando il vincolo temporale il problema di controllo viene semplificato sostanzialmente e si permette dunque di ottenere una soluzione più rapida e precisa rispetto alla traiettoria da seguire, che è il principale obiettivo dell'algoritmo di parcheggio.

[1] Si dice *vincolo anolonomo* un vincolo che non sia riconducibile ad una condizione sulla posizione x, y, z del corpo

Alcuni sistemi di controllo MPC utilizzano delle variabili di *slack* [2] per rilasciare vincoli sulle variabili di controllo o sulle variabili di stato del problema. Un esempio relativo ai veicoli può essere trovato in [16], dove si esegue un controllo di un motore diesel tramite MPC con slack variables.

2.2 Metodi di ottimizzazione non lineare

Il questo paragrafo si mostrano alcuni dei metodi di ottimizzazione non lineare che sono usati per applicazioni nel campo del controllo predittivo MPC non lineare. Inoltre, si dà una rassegna non esaustiva di metodi di ottimizzazione non lineare utilizzati nei problemi di ottimizzazione.

2.2.1 Il problema di ottimizzazione non lineare

Ogni problema di ottimizzazione non lineare può essere scritto come:

$$\begin{aligned} \min_{x \in D} \quad & \mathrm{J}\left(x(\cdot)\right) \\ \text{subject to} \quad & g(x) \geq 0, \\ & x \in D. \end{aligned} \tag{2.5}$$

dove $J(x(\cdot))$ è la funzione obiettivo non lineare in x, D è il campo di ammissibilità della soluzione x e $g(x)$ è la funzione di vincolo sulla variabile da ottimizzare.

2.2.2 Metodi classici

Si comincia questa rassegna con alcuni metodi classici di ottimizzazione non lineare. Si presentano in seguito metodi che possono essere tutti trovati in [20].

In generale, lo schema di un qualsiasi algoritmo di soluzione di un problema di ottimizzazione non lineare può essere riassunto nella seguente Figura 2.1

[2] Sono dette variabili di slack le variabili ausiliarie utilizzate per trasformare disuguaglianze in uguaglianze nei vincoli

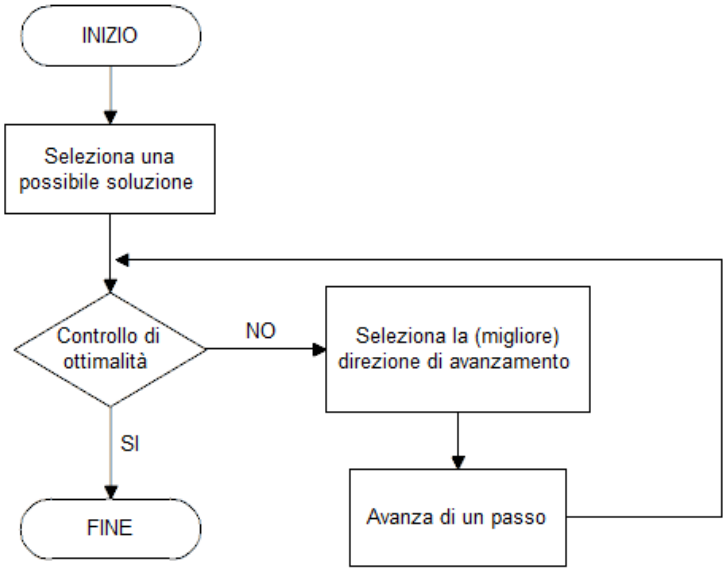

Figura 2.1: Algoritmo per Ottimizzazione Nonlineare

Gli algoritmi basilari di ottimizzazione non vincolata che andiamo a trattare in questo capitolo possono essere divisi nelle categorie seguenti:

- ricerca lineare senza uso di derivate
- ricerca lineare con l'uso di derivate

Algoritmi di ottimo non lineare senza l'uso di derivate

Tra i metodi di ricerca lineare senza l'uso di derivate il più semplice è sicuramente il metodo detto *a ricerca dicotomica*. Tale metodo segue un concetto molto semplice per minimizzare una funzione $\vartheta(x) : \mathbb{R} \to \mathbb{R}$ su un intervallo $[a_1, b_1]$. I passi utilizzati sono:

- Al passo k si valuta la funzione in un intorno del punto medio dell'intervallo, ottenendo i valori $f(m+)$ ed $f(m-)$.
- se $f(m-) > f(m+)$ allora il nuovo intervallo sarà $[m-, b]$; altrimenti sarà $[a, m+]$.

Di conseguenza il metodo di ricerca dicotomica trova un minimo all'interno di un problema convesso o quasi-convesso. La convergenza è però lenta.
In Figura 2.2 si è rappresentato il funzionamento della ricerca dicotomica nei due casi. $f1$ rappresenta una funzione per cui, nell'intervallo $[a, b]$, si effettua la scelta $[m-, b]$; $f2$ rappresenta la scelta opposta.

Un'evoluzione del metodo di ricerca dicotomica che permette una convergenza più rapida ed un passo di convergenza costante è il metodo *a sezione aurea*. In

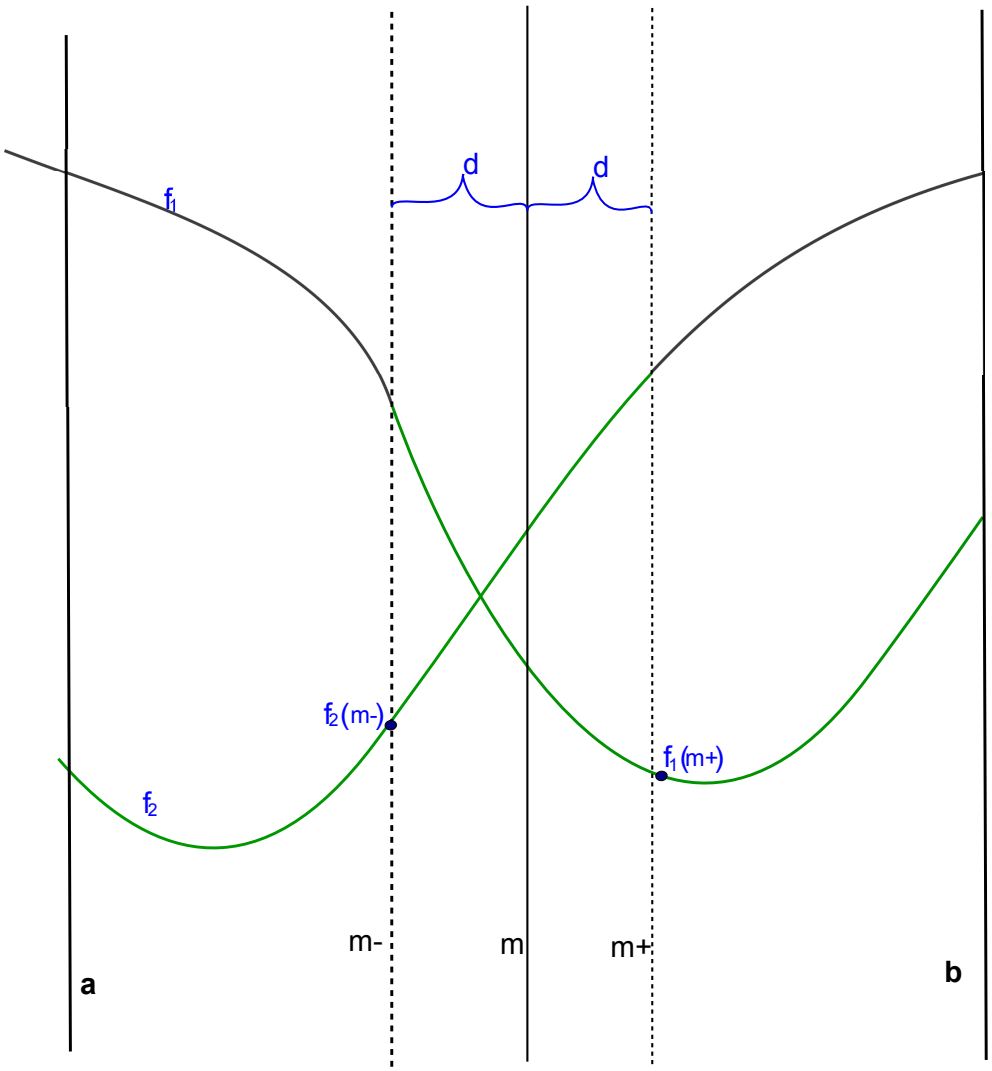

Figura 2.2: Esempio di scelte nella ricerca dicotomica

questo metodo l'ampiezza dell'intervallo viene sempre ridotta di un fattore φ, da cui il nome del metodo (ogni intervallo è la sezione aurea del precedente).
Sempre analogo è il metodo detto *di Fibonacci*, che basa la riduzione dell'ampiezza dell'intervallo sui fattori componenti l'omonima serie. Seguendo tale metodo l'ampiezza dell'intervallo di incertezza della soluzione al passo k è predeterminata e calcolabile tramite $d_0/F(k)$, dove F è la famosa serie di Fibonacci e d_0 l'ampiezza dell'intervallo iniziale.

Tutti i metodi precedenti sono costruiti per effettuare l'ottimizzazione ad una sola dimensione (cioè su una sola variabile). Per problemi multivariabile (la gran maggioranza dei problemi di ottimo) si utilizzano metodi multivariabile.
Un metodo multivariabile che fa parte dei metodi di ricerca senza l'uso di derivate è il *Metodo di Rosenbrock*. Questo metodo di ricerca utilizza un vettore di direzioni di dimensione n, pari alla dimensione del problema, ed esegue ad ogni passo un'ortogonalizzazione di Gram-Schmidt per ottenere il nuovo vettore ortonormale delle direzioni. L'algoritmo di funzionamento è il seguente:

Algorithm *Metodo di Rosenbrock*()
1. Scegli un punto di partenza x_1, $y_1 = x_1$, $k = j = 1$
2. Ad ogni iterazione:
3. λ_j sia la soluzione ottima della minimizzazione di $f(y_j + \lambda d_j)$
4. $y_{j+1} = y_j + \lambda_j d_j$
5. se $j < n$ $j = j + 1$
6. altrimenti $x_{k+1} = y_{n+1}$
7. se $\|x_{k+1} - x_k\| < \varepsilon$ stop
8. altrimenti $y_1 = x_{k+1}$, $k = k + 1$, $j = 1$ e crea un nuovo set di d con un'opportuna ortogonalizzazione di Gram-Schmidt

Il metodo di Rosenbrock esegue, dunque, una minimizzazione lineare in ogni direzione ortonormale di Gram-Schmidt ad ogni iterazione del ciclo di ottimizzazione.

Algoritmi di ottimo non lineare con l'uso di derivate

Tra gli algoritmi di ottimo che usano le derivate, il più semplice è il metodo *di bisezione*. Questo metodo di ottimizzazione, da non confondere con il metodo di bisezione per trovare lo zero di una funzione, inizia scegliendo il numero di passi che compie, per rispettare una determinata precisione. Scelta la precisione obiettivo $l > 0$, la funzione da ottimizzare $\vartheta(x)$ e dato l'intervallo iniziale $[a1, b1]$, si ha che il numero di iterazioni del metodo di bisezione è:

$$n \geq \log_{1/2}(\frac{l}{b_1 - a_1}) \tag{2.6}$$

Ad ogni iterazione k, si sceglie un λ_k secondo la regola:

$$\lambda_k = \frac{1}{2}(a_k + b_k) \tag{2.7}$$

Se $\frac{\partial \vartheta}{\partial \lambda}(\lambda_k) = 0$ allora l'algoritmo si ferma perché λ è ottimo.
Se invece $\frac{\partial \vartheta}{\partial \lambda}(\lambda_k) > 0$, si ripete l'iterazione con il nuovo intervallo $[a_k, \lambda_k]$.
Altrimenti, si ripete l'iterazione con l'intervallo $[\lambda_k, b_k]$.
L'algoritmo si ferma in ogni caso se $k = n$. In tal caso si sa che l'ottimo è all'interno di $[a_k, b_k]$.

Il metodo *di Newton*, universalmente noto, è storicamente il primo metodo a derivata prima per trovare lo zero di una funzione.
Il metodo di *Gauss* applica il metodo di Newton alla funzione derivata, per ottenere lo zero della stessa, cioè il minimo della funzione originaria da ottimizzare. Il metodo di Gauss utilizza dunque derivata prima e derivata seconda. La formula di Gauss-Newton è:

$$\lambda_{k+1} = \lambda_k - \left(\frac{\partial^2 \vartheta}{\partial \lambda^2}(\lambda_k)\right)^{-1} \left(\frac{\partial \vartheta}{\partial \lambda}(\lambda_k)\right) \tag{2.8}$$

Questo metodo è di provata e rapida convergenza per problemi convessi, ed entrerà a far parte dell'algoritmo di ricerca del minimo che usiamo (si veda il capitolo 4) insieme ad altre strategie quali:

- Cambi di base
- Controllo di convessità
- Steepest-descent

Un altro metodo per ottimizzazione che utilizza soltanto la derivata prima è il metodo di *steepest descent* che si utilizza per la nostra ottimizzazione in condizioni quasi-convesse. Il metodo segue un principio molto semplice, ideato da Cauchy nel 1847. Ad ogni istante si sceglie $d_k = -\nabla f(x_k)$ e si esegue un passo di lunghezza λ_k in direzione d_k.
Si itera finché $\|\nabla f(x_k)\| < \varepsilon$.

Si presenta inoltre un metodo che, pur utilizzando le derivate prime e seconde della funzione obiettivo, ma in modo diverso rispetto all'algoritmo di Gauss-Newton. Si tratta del metodo *Quadratic Fit Line Search*, che adatta il metodo di Newton per ottenere una convergenza globale, sotto l'ipotesi di una funzione obiettivo ϑ continua e quasi-convessa. Il metodo può essere riassunto nell'algoritmo seguente:

Algorithm *Metodo Quadratic Fit Line Search*()
1. Si sceglie una terna λ_1, λ_2, λ_3 tali che, definiti $\forall j \theta_j \triangleq \vartheta(\lambda_j)$, si abbia che $\vartheta_2 \leq \vartheta_1$ e $\vartheta_2 \leq \vartheta_3$.
2. Si esegue il fitting di una curva quadratica che passi per i punti (λ_1, ϑ_1), (λ_2, ϑ_2), (λ_3, ϑ_3) e la si chiama $q(\lambda)$.
3. Si cerca il minimo $\overline{\lambda}$ di $q(\lambda)$ e si calcola il corrispondente $\overline{\vartheta}$.
4. In base al valore di $\overline{\lambda} - \lambda_2$ si aggiorna la terna

5. Si esce dal ciclo se $\overline{\lambda} - \lambda_2 = 0$ e $\lambda_3 - \lambda_1 < \varepsilon$.

In Figura 2.3 si rappresenta la situazione in cui si sceglierà come nuova terna $(\lambda_1, \lambda_2, \overline{\lambda})$. I punti in blu 1, 2, 3 rappresentano la terna di λ, mentre il punto in nero è il minimo della funzione quadratica in verde.

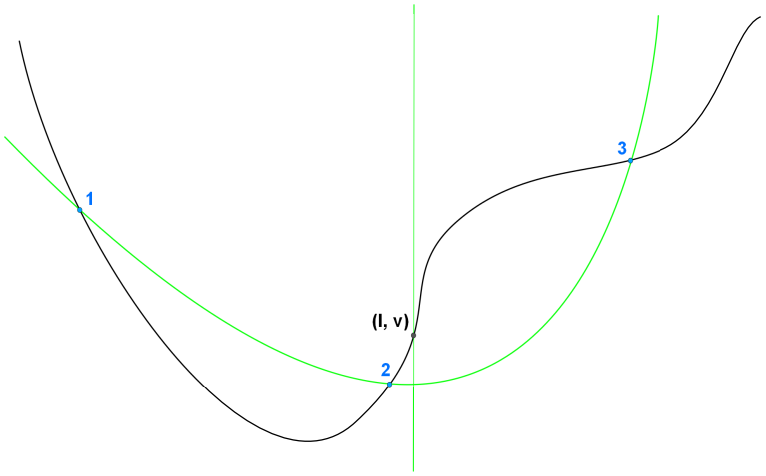

Figura 2.3: Quadratic Fit Method

La ricerca del passo: regola di Armijo

La ricerca di Armijo è un popolare metodo di ricerca del passo nella soluzione di problemi di ottimo non lineare non vincolati. Gli algoritmi di questo tipo sono detti di *backtracking line search*. Si tratta di metodi di ricerca lineare, per determinare il massimo passo da compiere lungo una direzione scelta di ottimizzazione. La direzione può essere scelta mediante il metodo di Gauss-Newton, o con il metodo del gradiente, o con altri più complessi metodi di ottimizzazione non lineare. Il metodo di Armijo è descritto nell'algoritmo sottostante:

Algorithm *Ricerca di Armijo*()
1. Inizia con una lunghezza di step candidata $\alpha_0 > 0$
2. Imposta $t = -\gamma \nabla f(x_0)'$ e un contatore di iterazione $j = 0$
3. Finchè non accade che $f(x) - f(x + \alpha_j p) \geq \alpha_j t$, incrementa j, aggiorna t e imposta sempre $\alpha_j = t\alpha_{j-1}$.
4. Restituisci α_j come soluzione.

L'algoritmo, di conseguenza, riduce α_0 di un fattore τ ad ogni istante finchè la condizione non è soddisfatta.

2.2.3 Metodi avanzati

In base ai metodi analizzati in questo elaborato, si possono suddividere gli algoritmi di ottimizzazione non lineare di ultima generazione nelle seguenti sottoclassi:

- basati su Lagrangiano per rilassamento dei vincoli
- basati su algoritmi di programmazione lineare
- algoritmo a piani di taglio sequenziali

I metodi basati su Lagrangiano per rilassamento dei vincoli utilizzano una funzione Lagrangiana che ha l'obiettivo di integrare nella cifra di merito una penalizzazione che risponda ai vincoli sul sistema. In particolare si costruisce una nuova funzione obiettivo che inglobi i vincoli, nel seguente modo:

$$\begin{aligned} \min_{x \in D} \quad & \mathrm{J}\left(x(\cdot)\right) - \lambda g_d(x) \\ \text{subject to} \quad & g_e(x) \geq 0, \\ & x \in D. \end{aligned} \tag{2.9}$$

dove con g_e si sono indicate le dinamiche della funzione di vincolo che sono facilmente risolvibili; con g_d le dinamiche che, al contrario, creano problemi alla risoluzione del sistema. Utilizzando λ, detta anche *moltiplicatore di Lagrange*, si ha che per ogni $\lambda > 0$ la soluzione ottima costituisce un lower bound sull'ottimo del problema originario. Questo procedimento permette dunque di inserire i vincoli "problematici" direttamente all'interno della cifra di merito del problema. Un esempio di metodo di questo tipo applicato a problemi non lineari può essere trovato in [24].

I metodi di ricerca dell'ottimo basati su algoritmi di programmazione lineare sono molto utili per valutare ed eseguire rapidamente controllo MPC in problemi di tipo lineare a vincoli lineari. Di questi è possibile trovare un esempio applicato direttamente in MPC in [13], dove si sviluppa un algoritmo di applicazione su MPC di un metodo di ottimizzazione lineare *Interior Point*. Un algoritmo sempre di tipo Interior Point viene utilizzato per problemi di tipo non lineare in [30]. Per l'utilizzo in programmazione non lineare si sono sviluppati nel tempo algoritmi detti *a piani di taglio sequenziali*. In [25] se ne può trovare un esempio costruito per ottimizzazione non lineare. Il metodo presentato si applica a problemi di ottimizzazione non lineare dove il Gradiente e l'Hessiano della funzione da minimizzare non siano noti a priori. Il principio di funzionamento è il seguente:

Algorithm *Sequential Cutting NLP*()
1. Si inizializza l'Hessiano e la soluzione
2. Generare il problema LP (cutting plane)
3. Risolverlo per trovare la direzione di passo
4. Eseguire una ricerca in linea per minimizzare la Lagrangiana nella direzione corrente
5. Eseguire il passo e aggiornare le stime di Hessiano e Gradiente
6. Ripetere fino a convergenza

L'algoritmo appena presentato è testato, sempre in [25], su problemi non lineari convessi e quasi-convessi. Una evoluzione di questo algoritmo, adatta a risolvere problemi di programmazione non lineare differenziabili con continuità si può trovare in [26].

2.3 Discretizzazione di problemi di controllo ottimo

Nel corso di questo elaborato si eseguono discretizzazioni con vari metodi di simulazione a partire da un problema di controllo ottimo. Di seguito mostriamo alcuni casi in letteratura in cui si sono utilizzati metodi simili per risolvere problemi di controllo ottimo, come si possono trovare in [29], in [10] e in [14]. L'obiettivo di questo paragrafo è solamente di citare alcuni esempi in vari campi che siano in grado mostrare la validità, a livello di letteratura, dell'approccio utilizzato; come mostrano i numerosi esempi, dei quali i tre precedenti sono soltanto una piccola parte, non vi è in generale perdita di proprietà a causa della discretizzazione, nell'esecuzione di un controllo ottimo a tempo continuo, purché si abbia l'accortezza di utilizzare metodi di simulazione all'interno del loro campo di stabilità e si imponga un'accuratezza sufficientemente elevata.

2.4 Alcuni metodi di simulazione

Infine, ci si pone l'obiettivo di introdurre alcuni metodi di simulazione non elementari che, sebbene non implementati nel nostro controllo ottimo, possono essere inseriti come sviluppo ulteriore, con l'obiettivo di migliorare l'accuratezza e conseguentemente di coprire un orizzonte di predizione più ampio a pari numero di passi. Per la trattazione dettagliata dei concetti di accuratezza, ordine nonché stiffness e A-stabilità[3] dei vari metodi si rimanda a [5]. I metodi che si trattano in questa sezione sono i seguenti:

- Metodo di Eulero all'indietro
- Metodo del trapezio
- Metodi di Runge-Kutta
- Metodi di Gauss
- Metodi BDF

[3]L'accuratezza e l'ordine sono una misura di come la soluzione approssimata fornita dal metodo numerico si avvicini alla soluzione esatta

2.4.1 Metodo di Eulero all'indietro

Il metodo di discretizzazione di Eulero all'indietro rappresenta il più semplice tra i cosiddetti "metodi impliciti". Il metodo consiste nell'eseguire una risoluzione implicita dell'equazione differenziale che rappresenta il sistema, ad ogni passo di integrazione. Il sistema non lineare viene discretizzato nel seguente modo:

$$\begin{cases} x(k+1) & = x(k) + h\, f\,(x(k+1)\, u(k+1)) \\ y(k+1) & = g\,(x(k+1)\, u(k+1))) \\ x(0) & = x\,(t_0) \end{cases} \tag{2.10}$$

Si noti come, nel sistema (2.10), i valori all'istante $k+1$ appaiano sia a sinistra che a destra del segno di uguaglianza. I metodi di simulazione numerica che possiedono tale caratteristica sono detti *metodi impliciti*. La soluzione del sistema implicito deve essere ricercata per via numerica anch'essa. Il sistema 2.10 é risolto ad ogni passo di integrazione per trovare il punto $(x(k+1)\, y(k+1))$.
I vantaggi di questo procedimento sono molteplici:

- diventa possibile allungare il passo di integrazione a qualsiasi livello, mantenendo stabile la convergenza del metodo (ma perdendo in accuratezza)
- le equazioni di vincolo del sistema vengono risolte per ogni punto trovato, garantendone il rispetto

Questi vantaggi danno come costo un maggiore tempo computazionale, dovuto proprio alla risoluzione implicita dei vincoli.

2.4.2 Metodo del Trapezio

Il metodo del trapezio è il metodo più semplice trattato in questa sezione. Si tratta di una combinazione dei due metodi più elementari in assoluto: Eulero in avanti ed Eulero all'indietro. Partendo da un punto x_k, infatti, ogni iterazione del metodo del Trapezio esegue la derivata della funzione $f(x)$ nel punto x_k e la derivata nel punto x_{k+1}. Il punto x_{k+1} è scelto tramite il passo temporale h_{k+1} risolvendo la seguente equazione:

$$x_{k+1} = x_k + \frac{h_{k+1}}{2}\,(f_k + f_{k+1}) \tag{2.11}$$

dove con f_k si indica la derivata di f in x_k ($\frac{\partial f}{\partial x}\big|_k$). Il metodo esegue dunque una media tra le derivate all'inizio ed alla fine del passo, ottenendo un risultato migliore sia di Eulero in avanti (che si limita alla derivata nel punto iniziale) sia di Eulero all'indietro (che esegue la derivata nel punto finale del passo). In Figura 2.4 si presenta graficamente l'esecuzione di un passo del metodo e la differenza con Eulero. Il metodo del Trapezio risulta dunque essere più accurato dei metodi di Eulero, pur mantenendo la A-stabilità.

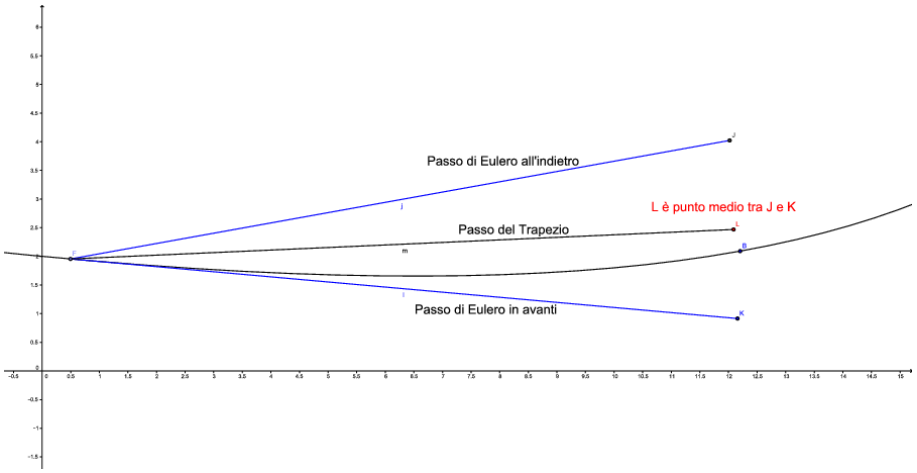

Figura 2.4: Il metodo del trapezio

2.4.3 Metodi di Runge-Kutta

I metodi di Runge-Kutta sono un'evoluzione del metodo del Trapezio. Mentre il metodo del Trapezio esegue la media delle derivate tra il punto x_k ed il punto x_{k+1}, il più popolare dei metodi di Runge-Kutta (di ordine 4) esegue la valutazione della derivata anche a metà del passo e la utilizza per effettuare una media pesata, per migliorare l'accuratezza della simulazione. In questo modo, l'esecuzione del passo avviene tramite la formula:

$$X_1 = x_k \tag{2.12}$$

$$X_2 = x_k + \frac{h_{k+1}}{2} f\left(t_k, X_1\right) \tag{2.13}$$

$$X_3 = x_k + \frac{h_{k+1}}{2} f\left(t_{k+\frac{1}{2}}, X_2\right) \tag{2.14}$$

$$X_4 = x_k + h_{k+1} f\left(t_{k+\frac{1}{2}}, X_3\right) \tag{2.15}$$

$$x_{k+1} = x_k + \frac{h_{k+1}}{6}\left(f(t_k, X_1) + 2f(t_{k+\frac{1}{2}}, X_2) + 2f(t_{k+\frac{1}{2}}, X_3) + f(t_{k+1}, X_4)\right) \tag{2.16}$$

dove le equazioni (2.12, 2.13, 2.14, 2.15) sono dette *stadi di integrazione*, mentre la (2.16) è detta di *quadratura*. Il metodo, sebbene esplicito, risulta pesante computazionalmente, ma a pari lunghezza del passo fornisce una soluzione più accurata del metodo del trapezio. Il problema principale di questo metodo è la difficoltà di calcolo dei coefficienti per la media pesata, che viene eseguito off-line.

2.4.4 Metodi di Gauss

I metodi di Gauss impliciti consentono di raggiungere un ordine di accuratezza uguale al corrispondente metodo di Runge-Kutta, con lo stesso principio, ma permettono di usare un numero di stadi più basso (e quindi coefficienti ricavabili off-line in maniera più semplice) al prezzo di una risoluzione implicita di ogni stadio. A titolo di esempio si riporta il metodo del punto medio implicito (a due stadi). L'esecuzione del passo avviene tramite il sistema di equazioni implicite:

$$X_1 = x_k + \frac{h_{k+1}}{2} f\left(t_{k+\frac{1}{2}}, X_1\right) \tag{2.17}$$

$$x_{k+1} = x_k + k_{k+1} f\left(t_{k+\frac{1}{2}}, X_1\right) \tag{2.18}$$

Si valuta dunque la derivata di f una sola volta, e con due stadi (entrambi a risoluzione implicita) si ottiene un ordine di accuratezza pari a 4, il massimo ottenibile.

2.4.5 Metodi BDF

Tutti i metodi visti precedentemente eseguono ogni passo indipendentemente dal precedente. I metodi BDF, invece, fanno parte di una categoria di metodi (detti *a passo multiplo*) in quanto utilizzano gli ultimi n passi di integrazione già compiuti per ricostruire un'approssimazione di ordine elevato dell'espansione in serie di Taylor della soluzione esatta. L'approssimazione è tanto migliore quanti più passi pregressi vengono utilizzati per tale ricostruzione (ordine del metodo BDF). Ciò comporta che nel tratto iniziale di simulazione, non disponendo di un numero di passi pregressi necessari ad ottenere l'ordine n del metodo, si dovranno utilizzare metodi BDF di ordine crescente fino al raggiungimento dell'ordine n. In questa sede a questa breve introduzione per motivi di spazio, rimandando alla letteratura [5].

Capitolo 3

La forma LFT per problemi non lineari

In questo capitolo si introduce la forma LFT.
LFT sta per "Linear Fractional Transform". Si tratta di un metodo per rappresentare efficacemente e simulare rapidamente sistemi non lineari. Il sistema non lineare viene infatti rappresentato come composto da blocchi lineari e blocchi non lineari in retroazione.

Dato un sistema non lineare tempo invariante definito dal seguente sistema DAE (Differential Algebraic Equations) di indice 1:

$$\left\{ F\left(x, \dot{x}, p\right) = 0 \right. \tag{3.1}$$

dove x, $\dot{x}$, incognite del problema, siano le variabili di stato, e p i parametri del modello.

Si assume che il sistema DAE (3) possa essere trasformato in una forma LFT, seguendo il procedimento mostrato in [12].

La rappresentazione grafica del sistema LFT ottenuto dalla trasformazione può essere trovata in Figura 3.1. Il sistema LFT può essere coerentemente rappresentato con le equazioni seguenti:

$$\begin{aligned} \dot{x} &= Ax + B_1 w + B_2 \zeta + B_3 u \\ z &= C_1 x + D_{11} w + D_{12} \zeta + D_{13} u \\ \omega &= C_2 x + D_{21} w + D_{22} \zeta + D_{23} u \\ y &= C_3 x + D_{31} w + D_{32} \zeta + D_{33} u \\ w &= diag\left\{\delta_1 I_{r1}, \ldots, \delta_q I_{rq}\right\} z \triangleq \Delta z \\ \zeta &= \Theta\left(\omega\right) \end{aligned} \tag{3.2}$$

Nelle equazioni LFT si evidenziano le seguenti sezioni, corrispondenti a blocchi in Figura 3.1:

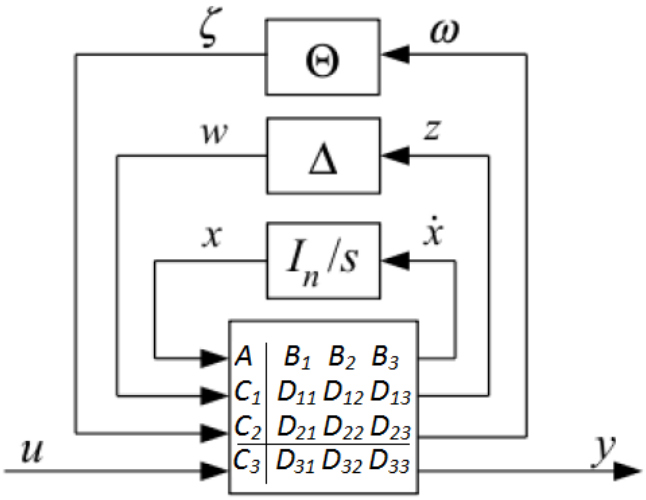

Figura 3.1: Sistema LFT

- un blocco LTI (Lineare Tempo Invariante)
- un blocco Δ che contiene i parametri affetti da incertezza
- un blocco Θ che contiene le funzioni non lineari note
- un blocco $\frac{I_n}{s}$ che integra $\dot{x}$ ottenendo lo stato del sistema

Il passaggio da sistema di equazioni DAE a sistema LFT, in molti casi, non può essere effettuato manualmente. In [8] si mostra un possibile procedimento per eseguire il passaggio da DAE ad LFT utilizzando l'ambiente di simulazione Modelica.

3.1 Identificazione parametrica basata su modelli LFT

Il modello LFT appena introdotto viene utilizzato da [12] per eseguire l'identificazione di parametri presenti nel blocco Δ. Il problema di identificazione a partire da struttura LFT è stato oggetto di ricerca per più di 10 anni. In particolare, in [2] si estende il metodo di stima dei parametri del modello alle funzioni non lineari. Di seguito si mostra il procedimento eseguito, utile per comprendere il procedimento utilizzato nei capitoli successivi per la derivazione delle equazioni di controllo ottimo.

Definita con $\hat{y}$ l'uscita del sistema LFT, si ottiene

$$e(\delta) = y_{mis} - \hat{y}(\delta) \tag{3.3}$$

dove con y_{mis} si indicano le variabili di uscita misurate per l'identificazione, con $\hat{y}(\delta)$ le medesime variabili di uscita stimate sul modello a partire dal valore corrente dei parametri δ. I parametri incogniti δ possono essere stimati tramite minimizzazione della cifra di merito:

$$J_p = \frac{1}{2}\langle e(\delta), e(\delta)\rangle \tag{3.4}$$

Approssimiamo J_p con la sua approssimante quadratica locale (cioè con il suo sviluppo di Taylor arrestato ai termini di secondo grado)

$$J_p(\delta) \approx V_p(\delta) \triangleq J_p(\delta^r) + \left(\left.\frac{\partial J_p}{\partial \delta}\right|_{\delta^r}\right)^T (\delta - \delta^r) + \\ + \frac{1}{2}(\delta - \delta^r)^T \left.\frac{\partial^2 J_p}{\partial \delta^2}\right|_{\delta^r} (\delta - \delta^r) \tag{3.5}$$

Da cui, derivando rispetto a δ, si ha:

$$\frac{\partial V_p}{\partial \delta} = \left(\left.\frac{\partial J_p}{\partial \delta}\right|_{\delta^r}\right)^T + (\delta - \delta^r)^T \left(\left.\frac{\partial^2 J_p}{\partial \delta^2}\right|_{\delta^r}\right) \tag{3.6}$$

e, uguagliando a zero, si ottiene:

$$\delta = \delta^r - \left(\left.\frac{\partial^2 J_p}{\partial \delta^2}\right|_{\delta^r}\right)^{-1} \left(\left.\frac{\partial J_p}{\partial \delta}\right|_{\delta^r}\right)^T \tag{3.7}$$

Eseguendo ad ogni iterazione r dell'algoritmo di identificazione la (3.7), si rende più accurata la stima di δ, cioè dei parametri incerti nel modello LFT.
Di conseguenza, per trovare la minimizzazione della cifra di merito J_p, é necessario calcolare ∇J_p e $\mathcal{H}(J_p)$.

$$\nabla J_p = \left.\frac{\partial J_p}{\partial \delta}\right|_{\delta^r} = \frac{1}{N}\sum_{t=1}^{T_p} e(t,\delta)^T \left.\frac{\partial e}{\partial \delta}\right|_{\delta^{(r)}} = \\ = -\frac{1}{N}\sum_{t=1}^{T_p} e(t,\delta)^T \left.\frac{\partial y}{\partial \delta}\right|_{\delta^{(r)}} = \\ = -\left\langle e(t,\delta), \frac{\partial y(t,\delta)}{\partial \delta}\right\rangle\Bigg|_{\delta^{(r)}} \tag{3.8}$$

$$\mathcal{H}(J_p) = \left.\frac{\partial^2 J_p}{\partial \delta^2}\right| = \frac{1}{N}\sum_{t=1}^{T_p}\left(\left.\frac{\partial e}{\partial \delta}\right|_{\delta^r}\right)^T \left.\frac{\partial e}{\partial \delta}\right|_{\delta^r} + \frac{1}{N}\sum_{t=1}^{T_p} e(t,\delta)^T \left.\frac{\partial^2 e}{\partial \delta^2}\right|_{\delta^{(r)}} \tag{3.9}$$

Trascurando ora i termini di second'ordine rispetto ad e nella (3.9) si ha:

$$\mathcal{H}(J_p) \approx \left\langle \frac{\partial e(t,\delta)}{\partial \delta}, \frac{\partial e(t,\delta)}{\partial \delta}\right\rangle\Bigg|_{\delta^{(r)}} = \left\langle \frac{\partial y(t,\delta)}{\partial \delta}, \frac{\partial y(t,\delta)}{\partial \delta}\right\rangle\Bigg|_{\delta^{(r)}} \tag{3.10}$$

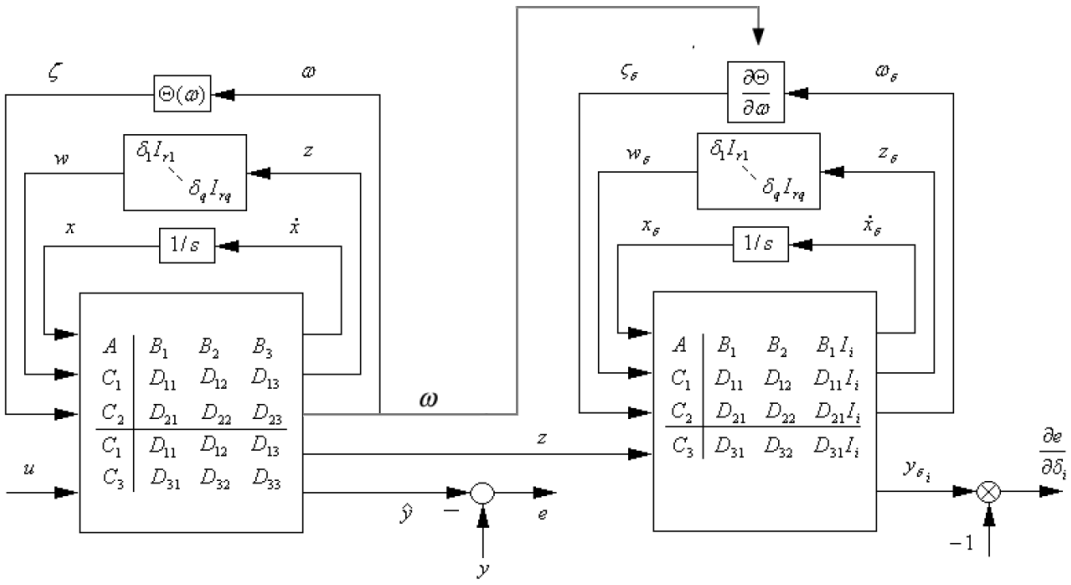

Figura 3.2: Doppio filtro LFT (Si ringrazia [2] per la gentile concessione)

Dalle (3.8, 3.9, 3.10) si nota come, per calcolare tutto ciò che è necessario all'identificazione, siano sufficienti:

- la matrice delle derivate di $y(t)$ (quindi della storia temporale di y) in $\delta_1, \delta_2, \ldots, \delta_q$
- il vettore degli $e(t)$, e quindi di $y(t)$, calcolato nell'iterazione attuale

Ognuno di questi due valori può essere calcolato, secondo l'idea esposta in [2, Paragrafo 3.2], tramite un apposito filtro LFT a due stadi, rappresentato in Figura 3.2.
Si esegue infatti una prima simulazione del filtro LFT (primo stadio), da cui si ricavano i valori di $\dot{x}$, $\hat{y}$, z ed ω in dipendenza dai valori degli ingressi u e della stima corrente dei parametri identificati δ. Con i valori di uscita di z e di ω, si esegue una simulazione del filtro LFT (secondo stadio), derivato rispetto al singolo parametro da identificare. L'esecuzione del filtro LFT di secondo stadio, detto anche di sensitività, viene eseguita tante volte quanti sono i parametri presenti nel vettore δ. Dal confronto tra $\hat{y}$ e y_{mis} si ottiene l'errore di stima

In Figura 3.2 si vede anche come il sistema LFT scelto per effettuare l'identificazione di Δ permetta di ottenere dai due filtri LFT in cascata tutti i valori richiesti dal metodo di identificazione parametrica. In particolare, il primo stadio viene simulato una sola volta per ottenere il valore di $\hat{y}$, mentre il secondo stadio (detto di sensitività) viene simulato q volte, con q pari al numero di parametri

da identificare presenti in δ.
Il primo stadio contiene il filtro LFT classico, rappresentato dalle equazioni (3.2). Il secondo stadio, invece, è la derivazione del filtro LFT classico rispetto a δ. Le equazioni risultanti sono:

$$\begin{aligned}
\dot{x}_\delta &= Ax_\delta + B_1\Delta z_\delta + B_2 \left.\frac{\partial\Theta(\omega)}{\partial\omega}\right|_\omega \omega_\delta + B_1 I_i z \\
z_\delta &= C_1 x_\delta + D_{11}\Delta z_\delta + D_{12} \left.\frac{\partial\Theta(\omega)}{\partial\omega}\right|_\omega \omega_\delta + D_{13} I_i z \\
\omega_\delta &= C_2 x_\delta + D_{21}\Delta z_\delta + D_{22} \left.\frac{\partial\Theta(\omega)}{\partial\omega}\right|_\omega \omega_\delta + D_{23} I_i z \\
y_\delta &= C_3 x_\delta + D_{31}\Delta z_\delta + D_{32} \left.\frac{\partial\Theta(\omega)}{\partial\omega}\right|_\omega \omega_\delta + D_{33} I_i z
\end{aligned} \tag{3.11}$$

Si noti dunque che il secondo stadio del filtro LFT è lineare tempo-variante, ed ha come ingressi z ed ω provenienti dal primo.

3.2 Soluzioni per l'aumento dell'efficienza computazionale

Data la linearità del secondo stadio del filtro, si può esplicitare:

$$\begin{bmatrix} z_\delta \\ \omega_\delta \end{bmatrix} = \Gamma_1(\omega) \left| \begin{matrix} x_\delta \\ I_i z \end{matrix} \right| \tag{3.12}$$

con

$$\Gamma_1(\omega) = \begin{bmatrix} I_z - D_{11}\Delta & -D_{12} \left.\frac{\partial\Theta(\omega)}{\partial\omega}\right|_\omega \\ -D_{21}\Delta & I_\omega - D_{22} \left.\frac{\partial\Theta(\omega)}{\partial\omega}\right|_\omega \end{bmatrix}^{-1} \begin{bmatrix} C_1 & D_{11} \\ C_2 & D_{21} \end{bmatrix} \tag{3.13}$$

Dall'equazione precedente, sostituendo nell'equazione di stato del sistema:

$$\dot{x}_\delta = \Gamma_2(\omega) \left| \begin{matrix} x_\delta \\ I_i z \end{matrix} \right| \tag{3.14}$$

con

$$\Gamma_2(\omega) = \left[\begin{matrix} A & B_1 \end{matrix} \right] + \left[\begin{matrix} B_1\Delta & B_2 \left.\frac{\partial\Theta(\omega)}{\partial\omega}\right|_\omega \end{matrix} \right] \Gamma_1(\omega) \tag{3.15}$$

Analizzando l'uso di $\Gamma_1(\omega)$ e $\Gamma_2(\omega)$, si nota come i loro componenti possano essere salvati per velocizzare molto l'esecuzione dello stadio di sensitività; calcolati durante il primo stadio, vengono salvati negli istanti di simulazione eseguiti, per poi essere interpolati e valutati in ognuno degli istanti di simulazione del secondo stadio. Si usano inoltre informazioni di sparsità per limitare il numero di interpolazioni.

Capitolo 4

Ottimizzazione Non Lineare basata sulla forma LFT

In questo capitolo si introduce un metodo largamente utilizzato per l'implementazione del controllo MPC che, come abbiamo visto, è intrinsecamente soggetto a vincoli sulla variabile di controllo. La possibilità di gestire nativamente i vincoli sia sulle variabili di controllo che su stato ed uscita è, come già detto, una delle caratteristiche che rendono il controllo MPC particolarmente utile e popolare, ma anche un motivo per il quale l'ottimizzazione vincolata che sta alla base dell'algoritmo di controllo MPC rimane lenta, specialmente nel caso non lineare.

Si consideri il problema di ottimizzazione vincolato:

$$\min J = \min \frac{1}{N_p} \sum_{j=k+1}^{k+N_p} \frac{1}{2} \left((\hat{y}(t_j) - y^o(t_j))^T Q \ (\hat{y}(t_j) - y^o(t_j)) + u(t_j)^T R \, u(t_j) \right) \tag{4.1}$$

$$subject\ to\ u_{min} < u(t_j) < u_{MAX} \tag{4.2}$$

dove si indicano con $\hat{y}$ l'uscita predetta dal modello, con y^o il riferimento e con u l'azione di controllo negli istanti t_j. Il problema 4.1 soggetto ai vincoli 4.2 può essere trasformato nel problema di ottimizzazione non vincolato descritto da:

$$min\ \tilde{J} = \\ \min \frac{1}{N_p} \sum_{j=k+1}^{k+N_p} \frac{1}{2} \left[\left(\left(\hat{y}_j - y_j^o \right)^T Q \ \left(\hat{y}_j - y_j^o \right) + u_j^T R \, u_j \right) + \mu B \left(u_j, \hat{y}_j, y_j^o \right) \right] \tag{4.3}$$

Dove $B(u, \hat{y})$ é detta *Barrier function* e viene usata per integrare i vincoli sulla variabile di ottimizzazione all'interno del funzionale di costo del problema di ottimizzazione non vincolato equivalente. Per ragioni di spazio si è inoltre utilizzata la notazione u_j, y_j dove si intende $u_j = u(t_j)$. Il moltiplicatore $\mu \in \mathbb{R}$

è un peso che varia in base all'importanza da attribuire al rispetto del vincolo. Questo metodo, alternativo al metodo dei moltiplicatori di Lagrange, permette inoltre di ottenere un affinamento iterativo della soluzione del problema di ottimo considerato.

Utilizzando infatti una Barrier function di tipo logaritmico, detta di Frisch (si veda [20, pag. 487]), si ha che per il nostro caso (con vincolo bilatero) otteniamo:

$$B(u(t_j)) = -\ln\left(1-\left(\frac{u(t_j)-u_{min}}{u_{MAX}-u_{min}}\right)^N\right)-\ln\left(1-\left(\frac{u_{MAX}-u(t_j)}{u_{MAX}-u_{min}}\right)^N\right) \quad (4.4)$$

per la Barrier function relativa ai valori consentiti alle variabili di controllo (per semplicità di notazione si trascura qui la dipendenza dal tempo e la discretizzazione).
Per le Barrier function relative alle variabili di uscita si ottiene un'espressione analoga:

$$B(\hat{y}(t_j)) = -\ln\left(1-\left(\frac{\hat{y}(t_j)-y_{min}}{y_{MAX}-y_{min}}\right)^N\right)-\ln\left(1-\left(\frac{y_{MAX}-\hat{y}(t_j)}{y_{MAX}-y_{min}}\right)^N\right) \quad (4.5)$$

Senza perdere di generalità, la Barrier function $B\left(u(t_j), \hat{y}(t_j), y^o(t_j)\right)$ può essere scritta come combinazione delle (4.4) e (4.5). La Barrier function scelta, al variare di u e di y all'interno dell'intervallo (in figura simmetrico di ampiezza 3), assume la forma indicata in Figura 4.1. Si noti che la funzione diminuisce rapidamente al centro al crescere di N, mentre richiede un N molto elevato per "schiacciarsi" agli estremi dell'intervallo di ammissibilità. Queste due caratteristiche permettono di ottenere una convergenza rapida della soluzione ad un ottimo interno alla zona ammissibile.

4.1 Costruzione di un algoritmo di ottimizzazione non lineare

Si presenta ora il metodo utilizzato per la costruzione di un algoritmo di ottimizzazione non lineare semplice, che però sia in grado di trovare soluzioni all'interno di un campo in cui la funzione obiettivo sia quasi-convessa. Per la definizione di quasi-convessità si rimanda a [20].

Il metodo utilizzato in questo elaborato combina il metodo detto di Gauss-Newton, che si dimostra essere convergente ad un ottimo all'interno di una regione convessa con funzione obiettivo convessa, ed il metodo detto di "steepest descent", che lavora facendo evolvere la ricerca dell'ottimo lungo la linea di massima pendenza. Si è così ottenuto un algoritmo che trova il minimo della funzione obiettivo dove esso esista e sia raggiungibile in condizioni di quasi-convessità. Nelle sottosezioni 4.1.1 e 4.1.2 si mostrano i due metodi sopra citati ed il loro funzionamento.

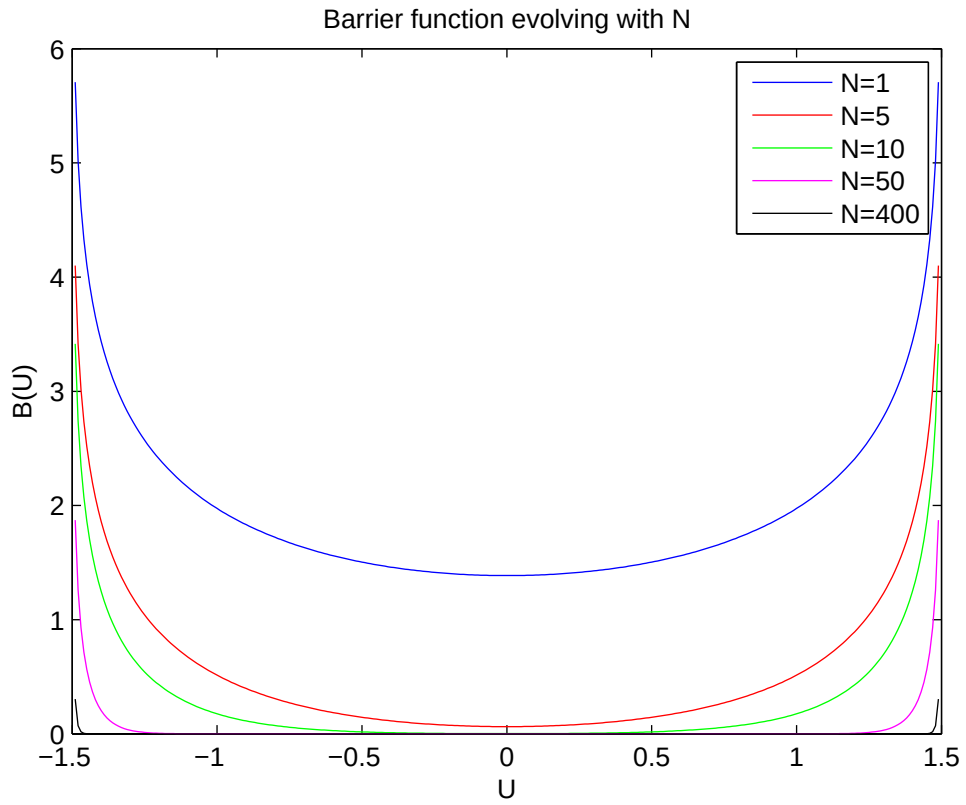

Figura 4.1: Barrier function al variare di N sull'intervallo $U_{min} : U_{MAX}$

4.1.1 Metodo di Gauss-Newton

Il principio di base del metodo di Gauss-Newton deriva direttamente dall'esistenza dell'approssimante in serie di Taylor al secondo ordine di una funzione non lineare. Eseguendo infatti l'approssimazione in serie di Taylor di una $f : \mathbb{R}^n \to \mathbb{R}$ si ha:

$$f(x) = f(x_o) + \nabla(x_o)(x - x_o)^T + \frac{1}{2}(x - x_o)^T H(x_o)(x - x_o) + o\left((x - x_o)^T(x - x_o)\right) \tag{4.6}$$

Dove, se f è una funzione doppiamente differenziabile in x_o, ∇ e H sono ben definiti. L'idea del metodo di Gauss-Newton è di trovare il punto dove la funzione derivata di $f(x)$ (cioè il gradiente) diventa nulla mediante il metodo delle tangenti (da cui il nome di Newton) e raggiungere tali punti con passi tali che l'Hessiano sia sempre definito positivo (quindi con una concavità rivolta verso l'alto).
Il metodo di Gauss-Newton cerca un ottimo con la formula $x_{k+1} = x_k - H(x_k)^{-1}\nabla(x_k)$.
Si supponga di trovarsi in condizioni per cui il gradiente nel punto di ottimo sia

nullo (cioè un vettore nullo). Sotto queste condizioni, il metodo di Gauss-Newton è ben definito e convergente ad un ottimo locale di f.
Vale infatti la seguente proposizione, che può essere trovata in [20]:

Proposizione 4.1.1
Siano $D \subseteq \mathbb{R}$ un insieme aperto convesso ed $f : \mathbb{R}^n \to \mathbb{R}$ una funzione che sia due volte continuamente differenziabile su D e definita come:

$$f(x) = \frac{1}{2} \sum_{i=1}^{m} \left(r_i(x)^2 \right) \tag{4.7}$$

Si supponga che valgano le seguenti condizioni:

- *esiste un $x^* \in D$ tale che la matrice Jacobiana $J(x)$ ed il residuo $r(x)$ sono tali che $J(x^*)^T r(x^*) = 0$;*
- *la matrice Jacobiana $J(x)$ è Lipschitz-continua su D;*
- *esiste un $\sigma \geq 0$ tale che*

$$\left| \left(J(x) - J(x^*) \right)^T r(x^*) \right| \leq \sigma \|x - x^*\| \tag{4.8}$$

  per ogni $x \in D$

Sia $\lambda \geq 0$ il più piccolo autovalore di $J(x^*)^T J(x)$. Se $\lambda > \sigma$, allora esiste una sfera aperta $B(x^*; \varepsilon)$ tale che per ogni $x_o \in B(x^*; \varepsilon)$ la sequenza generata con il metodo di Gauss-Newton

$$x_{k+1} = x_k - \left(J(x_k)^T J(x_k) \right)^{-1} J(x_k)^T r(x_k) \tag{4.9}$$

è ben definita, rimane in $B(x^*; \varepsilon)$ e converge ad x^*.

Il metodo di Gauss è quindi convergente e funzionale per funzioni di tipo quadratico, come ad esempio $f(x) = x^2 + y^2$. Non trova invece minimi corretti ove la funzione sia più che quadratica; i termini di secondo grado ignorati dall'approssimante di Taylor diventano infatti rilevanti.

4.1.2 Metodo steepest descent

È però possible estendere il metodo di Gauss-Newton se accoppiato con un altro metodo che gestisca l'ottimizzazione della cifra di merito nei casi in cui questa non sia semplicemente quadratica a residuo nullo, ma anche laddove ci si trovi in una condizione ibrida, ossia dove l'Hessiano della cifra di merito non sia definito positivo, ma esista una direzione di diminuzione della funzione obiettivo. In questi casi l'analisi dell'Hessiano non consente, di per sé, di determinare la direzione di movimento (la direzione determinata dal metodo di Gauss-Newton potrebbe

portare ad una crescita della cifra di merito!).
Il metodo detto "steepest descent" si basa su un principio semplice quanto efficace: la definizione di gradiente di una funzione. Dato che il gradiente ∇ è definito come la direzione locale di massima crescita della funzione, basta muoversi lungo la direzione del gradiente, ma con verso opposto ad esso, per andare nella direzione locale di massima decrescita della funzione obiettivo. La formula utilizzata di conseguenza è $x_{k+1} = x_k - \nabla(x_k)$. Questo metodo, ancorché semplice, permette di convergere all'interno di una condizione di funzione quasi-convessa su insieme convesso. Associandolo al metodo di Gauss-Newton all'interno della zona in cui la funzione è perfettamente convessa, i due metodi combinati permettono di ottenere una rapida convergenza nel caso quasi-convesso.

4.1.3 Combinazione dei due metodi

Si utilizza dunque una combinazione dei due metodi: in base al modulo del gradiente, si sceglie se usare il metodo "steepest descent" o il metodo di Gauss-Newton. Si segue "steepest descent" per ogni spostamento tra punti critici della funzione (tranne ove si possa applicare Gauss-Newton), e nei punti critici ove il gradiente sia nullo, si sceglie la direzione di movimento lungo l'autovalore più negativo dell'Hessiano, e ci si muove in direzione di decrescita. In tale modo si è in grado di gestire un'ottimizzazione non lineare per trovare un ottimo locale all'interno di un insieme convesso su cui la funzione sia quasi-convessa. Il metodo così ottenuto gestisce autonomamente la presenza di selle e di convessità non minime, conducendo la soluzione dell'ottimizzazione non lineare verso un ottimo locale che rispetti le condizioni di Karush-Kuhn Tucker (KKT). Per le definizioni delle condizioni KKT e per alcuni metodi avanzati di ottimizzazione, si rimanda a [20].

Capitolo 5

Utilizzo della formulazione LFT per problemi di controllo

5.1 Formulazione LFT per ottimizzazione

La rappresentazione LFT, introdotta nelle (3.2), viene qui riportata per semplicità:

$$\dot{x} = Ax + B_1 w + B_2 \zeta + B_3 u \tag{5.1}$$
$$z = C_1 x + D_{11} w + D_{12} \zeta + D_{13} u \tag{5.2}$$
$$\omega = C_2 x + D_{21} w + D_{22} \zeta + D_{23} u \tag{5.3}$$
$$y = C_3 x + D_{31} w + D_{23} \zeta + D_{33} u \tag{5.4}$$
$$w = diag\left\{\delta_1 I_{r1}, \ldots, \delta_q I_{rq}\right\} z = \Delta z \tag{5.5}$$
$$\zeta = \Theta(\omega) \tag{5.6}$$

L'errore di controllo sul modello è definito come:

$$e\left(t, u\right) = y^o(t, u) - \hat{y}(t, u) \tag{5.7}$$

dove con u si indica l'ingresso del sistema a tempo continuo, e con y^o e $\hat{y}$ rispettivamente il riferimento e il valore di predizione delle uscite.
L'obiettivo del controllo ottimo è di trovare un controllo $u(t)$ su un orizzonte $t \in [t_0, t_0 + T_c]$ tale che sia minimizzata la cifra di merito

$$J_0 = \min_{u(t), t \in [t_0, t_0+T_c]} J(u(t)) = \min_{u(t)} \frac{1}{2} \frac{1}{T_c} \int_{t_o}^{t_0+T_c} u(t)^T R u(t) + \left(y(t) - y^o(t)\right)^T Q \left(y(t) - y^o(t)\right) dt \tag{5.8}$$

Per i sistemi non lineari come (5.1) si ricorre generalmente ad una soluzione a tempo discreto. Per questo motivo si definiscono u_k, x_k e y_k come segnali

negli istanti t_k individuati sull'intervallo $[t_0, t_0 + T_c]$ da passi di campionamento h_k. Il problema si risolve a tempo discreto scegliendo un opportuno metodo di discretizzazione.
La cifra di merito del controllo ottimo (5.8) può quindi essere scritta, a tempo discreto, come:

$$J_o = \frac{1}{N_p} \sum_{j=k+1}^{k+N_p} \frac{1}{2} \left(e(t_k, u_k)^T Q\, e(t_k, u_k) + u_k^T R u_k \right) h_k \tag{5.9}$$

In questo capitolo si analizza un problema di controllo ottimo non vincolato. L'estensione con l'introduzione dei vincoli si può trovare nella sezione 6.2. Effettuando il medesimo procedimento svolto per l'identificazione si ha:

$$J_o(u_k) \approx V_o(u_k) = J_o(u_k^r) + \left(\left. \frac{\partial J_o}{\partial u_k} \right|_{u_k^r} \right)^T (u_k - u_k^r) + \frac{1}{2}(u_k - u_k^r)^T \left(\left. \frac{\partial^2 J_o}{\partial u_k^2} \right|_{u_k^r} \right) (u_k - u_k^r) \tag{5.10}$$

per cui:

$$\frac{\partial V_o}{\partial u_k} = \left(\left. \frac{\partial J_o}{\partial u_k} \right|_{u_k^r} \right)^T + (u_k - u_k^r)^T \left(\left. \frac{\partial^2 J_o}{\partial u_k^2} \right|_{u_k^r} \right) \equiv 0 \tag{5.11}$$

Risolvendo ancora rispetto ad u_k si trova la formula ricorsiva per l'ottimizzazione:

$$u_k^{r+1} = u_k^r - \left(\left. \frac{\partial^2 J_o}{\partial u_k^2} \right|_{u_k^r} \right)^{-1} \left(\left. \frac{\partial J_o}{\partial u_k} \right|_{u_k^r} \right)^T \tag{5.12}$$

Calcoliamo ora Gradiente ed Hessiano della cifra di merito:

$$\left. \frac{\partial J_o}{\partial u_k} \right|_{u_k^r} = \frac{1}{N_p} \sum_{j=k+1}^{k+N_p} h_k \left(u_k^T R + e_k^T Q \left(\left. \frac{\partial e_k}{\partial u_k} \right|_{u_k^r} \right) \right) \tag{5.13}$$

$$\left. \frac{\partial^2 J_o}{\partial u^2} \right|_{u_k^r} = \frac{1}{N_p} \sum_{j=k+1}^{k+N_p} h_k \left(R + \left(\left. \frac{\partial e_k}{\partial u_k} \right|_{u_k^r} \right)^T Q \left(\left. \frac{\partial e_k}{\partial u_k} \right|_{u_k^r} \right) + e_k^T Q \left(\left. \frac{\partial^2 e_k}{\partial u_k^2} \right|_{u_k^r} \right) \right) \tag{5.14}$$

Trascurando i termini di secondo ordine sull'errore e tenendo presente la (5.7), si

ottiene:

$$\nabla J_o = \left.\frac{\partial J_o}{\partial u_k}\right|_{u_k^r} = \frac{1}{N_p}\sum_{j=k+1}^{k+N_p} h_k \left(u_k^T R + \left(\hat{y}_k - y_k^o\right)^T Q \left(\left.\frac{\partial \hat{y}_k}{\partial u_k}\right|_{u_k^r} \right) \right) \tag{5.15}$$

$$\mathcal{H}\left(J_o\right) = \left.\frac{\partial^2 J_o}{\partial u_k^2}\right|_{u_k^r} = \frac{1}{N_p}\sum_{j=k+1}^{k+N_p} h_k \left(R + \left(\left.\frac{\partial \hat{y}_k}{\partial u_k}\right|_{u_k^r} \right)^T Q \left(\left.\frac{\partial \hat{y}_k}{\partial u_k}\right|_{u_k^r} \right) \right) \tag{5.16}$$

È quindi possibile con la sola derivata di $\hat{y}$ ottenere la formula ricorsiva di ottimizzazione.
Tale derivata deve qui essere calcolata non più in relazione a dei parametri costanti, bensì relativamente all'ingresso nel sistema, vale a dire relativamente alle variabili manipolabili, che variano nel tempo all'interno dell'orizzonte di controllo.
Di conseguenza, abbiamo che, con un orizzonte di controllo generico di n istanti, si può ottenere:

$$\frac{\partial \hat{y}\left(j\right)}{\partial u\left(i\right)} = \begin{bmatrix} \frac{\partial \hat{y}(1)}{\partial u(1)} & 0 & 0 & 0 & \cdots & \cdots & 0 \\ \frac{\partial \hat{y}(2)}{\partial u(1)} & \frac{\partial \hat{y}(2)}{\partial u(2)} & 0 & 0 & \cdots & \cdots & 0 \\ \frac{\partial \hat{y}(3)}{\partial u(1)} & \frac{\partial \hat{y}(3)}{\partial u(2)} & \frac{\partial \hat{y}(3)}{\partial u(3)} & 0 & \cdots & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & 0 & \cdots & 0 \\ \frac{\partial \hat{y}(n-1)}{\partial u(1)} & \frac{\partial \hat{y}(n-1)}{\partial u(2)} & \frac{\partial \hat{y}(n-1)}{\partial u(3)} & \frac{\partial \hat{y}(n-1)}{\partial u(4)} & \cdots & \frac{\partial \hat{y}(n-1)}{\partial u(n-1)} & 0 \\ \frac{\partial \hat{y}(n)}{\partial u(1)} & \frac{\partial \hat{y}(n)}{\partial u(2)} & \frac{\partial \hat{y}(n)}{\partial u(3)} & \frac{\partial \hat{y}(n)}{\partial u(4)} & \cdots & \frac{\partial \hat{y}(n)}{\partial u(n-1)} & \frac{\partial \hat{y}(n)}{\partial u(n)} \end{bmatrix} \tag{5.17}$$

La diagonale principale di (5.17), che rappresenta le derivate di $\hat{y}$ rispetto ad u valutate nell'istante corrente, può essere ottenuta direttamente dall'uscita del filtro di sensitività rispetto ad u. Si nota inoltre che la matrice è triangolare bassa. Ciò è motivato dalla causalità del sistema: gli ingressi futuri non hanno effetto sul sistema all'istante corrente, ma solo sull'uscita negli istanti successivi a quello in cui sono applicati.

5.2 Dal continuo al discreto

Nel paragrafo precedente si è dato per scontato il passaggio da tempo continuo a tempo discreto. In questo paragrafo si mostra come tale trasformazione viene

effettuata, e si dimostra la congruenza tra le due cifre di merito (a tempo continuo (5.8) e a tempo discreto (5.9)).

Il sistema LFT (3.2) può essere scritto, equivalentemente, come il sistema:

$$\begin{cases} \dot{x}(t) & = f\ (x(t), z(t), u(t)) \\ 0 & = v\ (x(t), z(t), u(t)) \\ y(t) & = g\ (x(t), z(t), u(t)) \\ x(0) & = x_0 \end{cases} \tag{5.18}$$

dove l'equazione $0 = v\ (x(t), z(t), u(t))$ indica la risoluzione dei vincoli algebrici nel problema LFT, e z raccoglie le variabili di vincolo del sistema LFT. Questa rappresentazione ci permette di esplicitare il significato delle operazioni di derivazione a tempo continuo.
Tramite una discretizzazione implicita, eseguita tramite il metodo di Eulero all'Indietro, si ottiene il sistema a tempo discreto:

$$\begin{cases} x(k+1) & = x(k) + h_{k+1}\, f\,(x(k+1), z(k+1), u(k+1)) \\ 0 & = v\,(x(k+1), z(k+1), u(k+1)) \\ y(k+1) & = g\,(x(k+1), z(k+1), u(k+1)) \\ x(0) & = x_0 \end{cases} \tag{5.19}$$

dove $x\,(k)$ indica naturalmente il valore della variabile x nell'istante t_k ed h_{k+1} é il passo di integrazione $t_{k+1} - t_k$.

Per ottenere le derivate di y rispetto ad u nei vari istanti che compongono la matrice (5.17), si possono effettuare le derivate direttamente dal sistema (5.19). In particolare, la derivata di y rispetto ad u nell'istante corrente risulterà pari a:

$$\frac{dy\,(k)}{du\,(k)} = \left.\frac{dg}{du}\right|_k \tag{5.20}$$

Mentre le derivate calcolate su tutti i componenti del sistema risulteranno:

$$\begin{cases} \frac{dx(k+1)}{du(k)} & = \frac{dx(k)}{du(k)} + h\left[\left.\frac{\partial f}{\partial x}\right|_{k+1}\frac{dx(k+1)}{du(k)} + \left.\frac{\partial f}{\partial z}\right|_{k+1}\frac{dz(k+1)}{du(k)} + \left.\frac{\partial f}{\partial u}\right|_{k+1}\frac{du(k+1)}{du(k)}\right] \\ 0 & = \left.\frac{\partial v}{\partial x}\right|_{k+1}\frac{dx(k+1)}{du(k)} + \left.\frac{\partial v}{\partial z}\right|_{k+1}\frac{dz(k+1)}{du(k)} + \left.\frac{\partial v}{\partial u}\right|_{k+1}\frac{du(k+1)}{du(k)} \\ \frac{dy(k+1)}{du(k)} & = \left.\frac{\partial g}{\partial x}\right|_{k+1}\frac{dx(k+1)}{du(k)} + \left.\frac{\partial g}{\partial z}\right|_{k+1}\frac{dz(k+1)}{du(k)} + \left.\frac{\partial g}{\partial u}\right|_{k+1}\frac{du(k+1)}{du(k)} \end{cases} \tag{5.21}$$

e, vista la non correlazione degli ingressi nei vari istanti di tempo (ovvero, dato che la matrice di covarianza degli ingressi è diagonale)si ha:

$$\begin{cases} \frac{dx(k+1)}{du(k)} & = \frac{dx(k)}{du(k)} + h_{k+1}\left[\left.\frac{\partial f}{\partial x}\right|_{k+1}\frac{dx(k+1)}{du(k)} + \left.\frac{\partial f}{\partial z}\right|_{k+1}\frac{dz(k+1)}{du(k)}\right] \\ 0 & = \left.\frac{\partial v}{\partial x}\right|_{k+1}\frac{dx(k+1)}{du(k)} + \left.\frac{\partial v}{\partial z}\right|_{k+1}\frac{dz(k+1)}{du(k)} \\ \frac{dy(k+1))}{du(k)} & = \left.\frac{\partial g}{\partial x}\right|_{k+1}\frac{dx(k+1)}{du(k)} + \left.\frac{\partial g}{\partial z}\right|_{k+1}\frac{dz(k+1)}{du(k)} \end{cases} \tag{5.22}$$

Esplicitando dunque il vincolo (operazione possibile, visto che il vincolo è una relazione puramente algebrica):

$$\frac{dz(k+1)}{du(k)} = -\left(\left.\frac{\partial v}{\partial z}\right|_{k+1}\right)^{-1} \left.\frac{\partial v}{\partial x}\right|_{k+1} \frac{dx(k+1)}{du(k)} \tag{5.23}$$

e, sostituendolo nell'equazione precedente:

$$\begin{cases} \frac{dx(k+1)}{du(k)} & = \frac{dx(k)}{du(k)} + h_{k+1}\left[\left.\frac{\partial f}{\partial x}\right|_{k+1} - \left.\frac{\partial f}{\partial z}\right|_{k+1}\left(\left.\frac{\partial v}{\partial z}\right|_{k+1}\right)^{-1}\left.\frac{\partial v}{\partial x}\right|_{k+1}\right]\frac{dx(k+1)}{du(k)} \\ \frac{dy(k+1)}{du(k))} & = \left\{\left.\frac{\partial g}{\partial x}\right|_{k+1} - \left.\frac{\partial g}{\partial z}\right|_{k+1}\left(\left.\frac{\partial v}{\partial z}\right|_{k+1}\right)^{-1}\left.\frac{\partial v}{\partial x}\right|_{k+1}\right\}\frac{dx(k+1)}{du(k)} \end{cases} \tag{5.24}$$

E di conseguenza, esplicitando la prima relazione:

$$\begin{cases} \frac{dx(k+1)}{du(k)} & = \left\{I_{nx} - h_{k+1}\left[\left.\frac{\partial f}{\partial x}\right|_{k+1} - \left.\frac{\partial f}{\partial z}\right|_{k+1}\left(\left.\frac{\partial v}{\partial z}\right|_{k+1}\right)^{-1}\left.\frac{\partial v}{\partial x}\right|_{k+1}\right]^{-1}\right\}\frac{dx(k)}{du(k)} \\ \frac{dy(k+1)}{du(k)} & = \left\{\left.\frac{\partial g}{\partial x}\right|_{k+1} - \left.\frac{\partial g}{\partial z}\right|_{k+1}\left(\left.\frac{\partial v}{\partial z}\right|_{k+1}\right)^{-1}\left.\frac{\partial v}{\partial x}\right|_{k+1}\right\}\frac{dx(k+1)}{du(k)} \end{cases} \tag{5.25}$$

Quindi allo scopo di determinare la derivata richiesta ad un passo è necessario il calcolo della derivata dello stato nell'ingresso all'instante corrente. Questa può essere calcolata semplicemente scalando nel tempo di un istante le prime due equazioni del sistema (5.19), ottenendo quindi:

$$\begin{cases} x(k) & = x(k-1)) + h_k\, f\left(x(k), z(k), u(k)\right) \\ 0 & = v\left(x(k), z(k), u(k)\right) \end{cases} \tag{5.26}$$

derivando poi rispetto ad $u\,(k)$ si ottiene:

$$\begin{cases} \frac{dx(k)}{du(k)} & = \frac{dx(k-1)}{du(k)} + h_k\left[\left.\frac{\partial f}{\partial x}\right|_k \frac{dx(k)}{du(k)} + \left.\frac{\partial f}{\partial z}\right|_k \frac{dz(k)}{du(k)} + \left.\frac{\partial f}{\partial u}\right|_k \frac{du(k)}{du(k)}\right] \\ 0 & = \left.\frac{\partial v}{\partial x}\right|_k \frac{dx(k)}{du(k)} + \left.\frac{\partial v}{\partial z}\right|_k \frac{dz(k)}{du(k)} + \left.\frac{\partial v}{\partial u}\right|_k \frac{du(k)}{du(k)} \end{cases} \tag{5.27}$$

E di conseguenza, considerando che le derivate di u rispetto a sé stesso nell'istante corrente sono identità e che, siccome il sistema (5.19) è causale, $\frac{dx(k-1)}{du(k)} = 0$, si può scrivere:

$$\begin{cases} \frac{dx(k)}{du(k)} & = h_k\left[\left.\frac{\partial f}{\partial x}\right|_k \frac{dx(k)}{du(k)} + \left.\frac{\partial f}{\partial z}\right|_k \frac{dz(k)}{du(k)} + \left.\frac{\partial f}{\partial u}\right|_k\right] \\ 0 & = \left.\frac{\partial v}{\partial x}\right|_k \frac{dx(k)}{du(k)} + \left.\frac{\partial v}{\partial z}\right|_k \frac{dz(k)}{du(k)} + \left.\frac{\partial v}{\partial u}\right|_k \end{cases} \tag{5.28}$$

Esplicitando il vincolo rispetto a z

$$\frac{dz(k)}{du(k)} = -\left(\left.\frac{\partial v}{\partial z}\right|_k\right)^{-1}\left[\left.\frac{\partial v}{\partial x}\right|_k \frac{dx(k)}{du(k)} + \left.\frac{\partial v}{\partial u}\right|_k\right] \tag{5.29}$$

e sostituendo come già fatto precedentemente si ha:

$$\frac{dx\,(k)}{du\,(k)} = h_k \left[\left.\frac{\partial f}{\partial x}\right|_k - \left.\frac{\partial f}{\partial z}\right|_k \left(\left.\frac{\partial v}{\partial z}\right|_k \right)^{-1} \left.\frac{\partial v}{\partial x}\right|_k \right] \frac{dx\,(k)}{du\,(k)} + h_k \left[\left.\frac{\partial f}{\partial u}\right|_k - \left.\frac{\partial f}{\partial z}\right|_k \left(\left.\frac{\partial v}{\partial z}\right|_k \right)^{-1} \left.\frac{\partial v}{\partial u}\right|_k \right] \quad (5.30)$$

da cui:

$$\frac{dx(k)}{du(k)} = \left\{ I_{nx} - h_k \left[\left.\frac{\partial f}{\partial x}\right|_k - \left.\frac{\partial f}{\partial z}\right|_k \left(\left.\frac{\partial v}{\partial z}\right|_k \right)^{-1} \left.\frac{\partial v}{\partial x}\right|_k \right] \right\}^{-1} h_k \left[\left.\frac{\partial f}{\partial u}\right|_k - \left.\frac{\partial f}{\partial z}\right|_k \left(\left.\frac{\partial v}{\partial z}\right|_k \right)^{-1} \left.\frac{\partial v}{\partial u}\right|_k \right] \quad (5.31)$$

e:

$$\frac{dx(k+1)}{du(k)} = \left\{ I_{nx} - h_{k+1} \left[\left.\frac{\partial f}{\partial x}\right|_{k+1} - \left.\frac{\partial f}{\partial z}\right|_{k+1} \left(\left.\frac{\partial v}{\partial z}\right|_{k+1} \right)^{-1} \left.\frac{\partial v}{\partial x}\right|_{k+1} \right] \right\}^{-1} \frac{dx\,(k)}{du(k)} \quad (5.32)$$

$$\frac{dy(k+1)}{du(k)} = \left[\left.\frac{\partial g}{\partial x}\right|_{k+1} - \left.\frac{\partial g}{\partial z}\right|_{k+1} \left(\left.\frac{\partial v}{\partial z}\right|_{k+1} \right)^{-1} \left.\frac{\partial v}{\partial x}\right|_{k+1} \right] \frac{dx\,(k+1)}{du\,(k)} \quad (5.33)$$

Le componenti delle (5.31), (5.32), (5.33) espressi tra parentesi quadre sono le derivate totali ottenibili dai filtri LFT costruiti come in seguito, nella sezione 5.3. In particolare, si ottengono le seguenti espressioni:

$$\left(\frac{d}{dx} \left(\frac{dx}{dt} \right) \right)\Big|_k = \left.\frac{df}{dx}\right|_k = \left[\left.\frac{\partial f}{\partial x}\right|_k - \left.\frac{\partial f}{\partial z}\right|_k \left(\left.\frac{\partial v}{\partial z}\right|_k \right)^{-1} \left.\frac{\partial v}{\partial x}\right|_k \right] \quad (5.34)$$

$$\left.\frac{dy}{dx}\right|_k = \left.\frac{dg}{dx}\right|_k = \left[\left.\frac{\partial g}{\partial x}\right|_k - \left.\frac{\partial g}{\partial z}\right|_k \left(\left.\frac{\partial v}{\partial z}\right|_k \right)^{-1} \left.\frac{\partial v}{\partial x}\right|_k \right] \quad (5.35)$$

$$\left(\frac{d}{du} \left(\frac{dx}{dt} \right) \right)\Big|_k = \left.\frac{df}{du}\right|_k = \left[\left.\frac{\partial f}{\partial u}\right|_k - \left.\frac{\partial f}{\partial z}\right|_k \left(\left.\frac{\partial v}{\partial z}\right|_k \right)^{-1} \left.\frac{\partial v}{\partial u}\right|_k \right] \quad (5.36)$$

e di conseguenza, unendo le precedenti con le (5.31),(5.32), (5.33) si ottengono le

relazioni:

$$\frac{dx\,(k)}{du\,(k)} = \left[I_{nx} - h_k \left.\left(\frac{d}{dx}\left(\frac{dx}{dt}\right)\right)\right|_k\right]^{-1} h_k \left.\left(\frac{d}{du}\left(\frac{dx}{dt}\right)\right)\right|_k \tag{5.37}$$

$$\frac{dx\,(k+1)}{du\,(k)} = \left[I_{nx} - h_{k+1} \left.\left(\frac{d}{dx}\left(\frac{dx}{dt}\right)\right)\right|_{k+1}\right]^{-1} \frac{dx\,(k)}{du(k)} \tag{5.38}$$

$$\frac{dy(k+1)}{du(k)} = \left[\left.\frac{dy}{dx}\right|_{k+1}\right] \frac{dx(k+1)}{du(k)} \tag{5.39}$$

Quindi la derivata ad un passo ricercata è:

$$\frac{dy\,(k+1)}{du(k)} = \left\{\left.\frac{dg}{dx}\right|_{k+1}\right\}\left\{I_{nx} - h_{k+1} \left.\frac{df}{dx}\right|_{k+1}\right\}^{-1}\left\{I_{nx} - h_k \left.\frac{df}{dx}\right|_k\right\}^{-1} h_k \left.\frac{df}{du}\right|_k \tag{5.40}$$

Si noti come la derivata ad un passo calcolata nella (5.40) sia ottenibile a partire dalle derivate totali delle funzioni dinamiche del sistema (5.18), valutate negli istanti corretti. Queste derivate totali possono essere calcolate per simulazione mediante un filtro LFT di sensitività, analogamente a quando fatto nel paragrafo 3.1.

Si mostra ora l'equivalenza delle cifre di merito del controllo ottimo a tempo discreto e a tempo continuo.
Ripartendo dalla cifra di merito a tempo continuo (5.8):

$$J_0 = \min_{u(t),t\in[t_0,t_0+T_c]} J(u(t)) = \min_{u(t)} \frac{1}{2}\frac{1}{T_c}\int_{t_o}^{t_0+T_c} u(t)^T R u(t) + \left(y(t) - y^o(t)\right)^T Q\left(y(t) - y^o(t)\right) dt$$

eseguendo una discretizzazione con $dt = (t_{k+1} - t_k) = h_k$ si ha:

$$J_{0_k} = \min_{u_k} \frac{1}{2}\frac{1}{N_c}\sum_{k=0}^{N_c}\left(u_k^T R\, u_k + \left(y_k - y_k^o\right)^T Q\left(y_k - y_k^o\right)\right) h_k \tag{5.41}$$

che risulta equivalente alla (5.9) anche a fronte di una serie di passi a lunghezza variabile. Di seguito, per rendere più semplice la trattazione, h_k si considera contenuta nei termini R e Q.

5.3 Derivazione dei filtri LFT per il calcolo delle sensitività

Per ottenere il secondo filtro necessario al calcolo della derivata dell'uscita rispetto all'ingresso, è necessario derivare il sistema di equazioni LFT rispetto ad u,

esattamente come fatto precedentemente rispetto a δ nel filtro di sensitività per l'identificazione.

Effettuando tale derivazione, si ottiene:

$$\begin{aligned} \dot{x}_u &= Ax_u + B_1\Delta z_u + B_2\Theta_\omega\omega_u + B_3 \\ z_u &= C_1x_u + D_{11}\Delta z_u + D_{12}\Theta_\omega\omega_u + D_{13} \\ \omega_u &= C_2x_u + D_{21}\Delta z_u + D_{22}\Theta_\omega\omega_u + D_{23} \\ y_u &= C_3x_u + D_{31}\Delta z_u + D_{32}\Theta_\omega\omega_u + D_{33} \end{aligned} \tag{5.42}$$

Dove, analogamente con quanto fatto in precedenza nel sistema LFT per l'identificazione, ogni variabile viene derivata rispetto ad u e vale la relazione

$$\Theta_\omega = \left.\frac{\partial\Theta}{\partial\omega}\right|_\omega$$

Inoltre, l'ordine di derivazione segue la regola [1]:

$$\dot{x}_u = \frac{d}{du}\left(\frac{dx}{dt}\right) = \frac{df}{du}$$

Dal filtro (5.42) si possono ottenere, di conseguenza, $\dot{x}_u$ e y_u.

Operando analogamente, effettuando però la derivazione rispetto allo stato corrente del sistema, si ha:

$$\begin{aligned} \dot{x}_x &= A + B_1\Delta z_x + B_2\Theta_\omega\omega_x \\ z_x &= C_1 + D_{11}\Delta z_x + D_{12}\Theta_\omega\omega_x \\ \omega_x &= C_2 + D_{21}\Delta z_x + D_{22}\Theta_\omega\omega_x \\ y_x &= C_3 + D_{31}\Delta z_x + D_{32}\Theta_\omega\omega_x \end{aligned} \tag{5.43}$$

Si noti che:

- il termine relativo ad u scompare per l'indipendenza dell'ingresso dallo stato (nel sistema dinamico)
- come prima, l'ordine di derivazione rimane [2]:

$$\dot{x}_x = \frac{d}{dx}\left(\frac{dx}{dt}\right) = \frac{df}{dx}$$

E quindi dal filtro (5.43) è possibile ottenere $\dot{x}_x$ e y_x.

I quattro termini ottenuti in questa sezione sono esattamente quelli che compongono il sistema (5.18) e che vanno a costituire le derivate necessarie al calcolo delle (5.40) che compongono il sistema (5.17).

[1] Si noti come si tratti di derivate totali, e non parziali, delle espressioni della dinamica

[2] Si noti che l'ordine opposto di derivazione non darebbe alcun risultato concreto.

5.4 Costruzione della matrice derivata

Nelle due sezioni precedenti si è mostrato come sia possibile ottenere da un opportuno filtro LFT le derivate necessarie al calcolo della (5.40). Questa è la derivata di y calcolata rispetto ad u ad un passo. Per ottenere tutte le derivate necessarie alla (5.17), tuttavia, è necessario determinare le derivate di sensitività dell'uscita rispetto all'ingresso in qualsiasi passo all'interno dell'orizzonte di predizione.
Per fare ciò in questa sezione si provvede a ripetere il procedimento della sezione 5.2, generalizzandolo a un qualsiasi passo di predizione.
Si riporta per comodità la (5.40):

$$\frac{dy\,(k+1)}{du\,(k)} = \left\{ \left.\frac{dg}{dx}\right|_{k+1} \right\} \left\{ I_{nx} - h_{k+1} \left.\frac{df}{dx}\right|_{k+1} \right\}^{-1} \left\{ I_{nx} - h_k \left.\frac{df}{dx}\right|_{k} \right\}^{-1} h_k \left.\frac{df}{du}\right|_{k}$$

e si adotta una nuova notazione, più sintetica, per definire le derivate delle funzioni componenti il sistema (5.18). Questa notazione combina le definizioni date nelle (5.34), (5.35) e (5.36) con le uscite dei sistemi LFT (5.42) e (5.43).

Si scrivono dunque le definizioni:

$$\tilde{A}(k) \triangleq \dot{x}_x\,(t_k) = \left.\frac{df}{dx}\right|_k \tag{5.44}$$

$$\tilde{B}(k) \triangleq \dot{x}_u\,(t_k) = \left.\frac{df}{du}\right|_k \tag{5.45}$$

$$\tilde{C}(k) \triangleq y_x\,(t_k) = \left.\frac{dg}{dx}\right|_k \tag{5.46}$$

$$\tilde{D}(k) \triangleq y_u\,(t_k) = \left.\frac{dg}{du}\right|_k \tag{5.47}$$

e la (5.40) come:

$$\frac{dy(k+1)}{du(k)} = \tilde{C}(k+1)) \left\{ I_{nx} - h_{k+1}\,\tilde{A}(k+1) \right\}^{-1} \left\{ I_{nx} - h_k\,\tilde{A}(k) \right\}^{-1} h_k\,\tilde{B}\,(k) \tag{5.48}$$

Notando inoltre, dalla (5.20), che:

$$\frac{dy(k)}{du(k)} = \tilde{D}(k)) \tag{5.49}$$

e ridefinendo:

$$\begin{aligned} A(k) &= \left\{ I_{nx} - h_k\,\tilde{A}(k)) \right\}^{-1} \\ B(k) &= h_k\,\tilde{B}(k) \\ C(k) &= \tilde{C}(k) \\ D(k) &= \tilde{D}(k) \end{aligned} \tag{5.50}$$

si ha:

$$\frac{dy(k+1)}{du(k))} = C(k+1)A(k+1)A(k)B(k) \tag{5.51}$$

Partendo dal sistema LFT "a due passi":

$$\begin{cases} x(k+2) & = x(k+1) + h\, f\ (x(k+2), z(k+2), u(k+2)) \\ 0 & = v\ (x(k+2), z(k+2), u(k+2)) \\ y(k+1) & = g\ (x(k+2), z(k+2), u(k+2)) \\ x(0) & = x_0 \end{cases} \tag{5.52}$$

Analogamente a quanto fatto prima, il sistema LFT derivato "a due passi" risulta essere:

$$\begin{cases} \frac{dx(k+2)}{du(k)} & = \left\{ I_{nx} - h_{k+2} \left[\frac{\partial f}{\partial x}\Big|_{k+2} - \frac{\partial f}{\partial z}\Big|_{k+2} \left(\frac{\partial v}{\partial z}\Big|_{k+2} \right)^{-1} \frac{\partial v}{\partial x}\Big|_{k+2} \right]^{-1} \right\} \frac{dx(k+1)}{du(k)} \\ \frac{dy(k+2)}{du(k)} & = \left\{ \frac{\partial g}{\partial x}\Big|_{k+2} - \frac{\partial g}{\partial z}\Big|_{k+2} \left(\frac{\partial v}{\partial z}\Big|_{k+2} \right)^{-1} \frac{\partial v}{\partial x}\Big|_{k+2} \right\} \frac{dx(k+2)}{du(k)} \end{cases} \tag{5.53}$$

Dove, unendo le (5.32), (5.33), si ottiene:

$$\frac{dx(k+2))}{du(k)} = \left\{ I_{nx} - h_{k+2} \left[\frac{\partial f}{\partial x}\Big|_{k+2} - \frac{\partial f}{\partial z}\Big|_{k+2} \left(\frac{\partial v}{\partial z}\Big|_{k+2} \right)^{-1} \frac{\partial v}{\partial x}\Big|_{k+2} \right]^{-1} \right\} \frac{dx(k+1)}{du(k)} \tag{5.54}$$

$$\frac{dy(k+2)}{du(k)} = \left[\frac{\partial g}{\partial x}\Big|_{k+2} - \frac{\partial g}{\partial z}\Big|_{k+2} \left(\frac{\partial v}{\partial z}\Big|_{k+2} \right)^{-1} \frac{\partial v}{\partial x}\Big|_{k+2} \right] \frac{dx(k+1)}{du(k)} \tag{5.55}$$

$$\frac{dx(k+1)}{du(k)} = \left\{ I_{nx} - h_{k+1} \left[\frac{\partial f}{\partial x}\Big|_{k+1} - \frac{\partial f}{\partial z}\Big|_{k+1} \left(\frac{\partial v}{\partial z}\Big|_{k+1} \right)^{-1} \frac{\partial v}{\partial x}\Big|_{k+1} \right] \right\}^{-1} \frac{dx(k)}{du(k)} \tag{5.56}$$

$$\frac{dy(k+1)}{du(k)} = \left[\frac{\partial g}{\partial x}\Big|_{k+1} - \frac{\partial g}{\partial z}\Big|_{k+1} \left(\frac{\partial v}{\partial z}\Big|_{k+1} \right)^{-1} \frac{\partial v}{\partial x}\Big|_{k+1} \right] \frac{dx(k+1)}{du(k)} \tag{5.57}$$

Applicando le sostituzioni già mostrate nelle (5.34), (5.35), (5.36), si ottiene

facilmente:

$$\frac{dx(k)}{du(k)} = \left\{ I_{nx} - h_k \left.\frac{\partial f}{\partial x}\right|_k \right\}^{-1} h \left.\frac{\partial f}{\partial u}\right|_k \tag{5.58}$$

$$\frac{dx(k+1)}{du(k)} = \left\{ I_{nx} - h_{k+1} \left.\frac{\partial f}{\partial x}\right|_{k+1} \right\}^{-1} \frac{dx(k)}{du(k)} \tag{5.59}$$

$$\frac{dy(k+1)}{du(k)} = \left\{ \left.\frac{\partial g}{\partial x}\right|_{k+1} \right\} \frac{dx(k+1)}{du(k)} \tag{5.60}$$

$$\frac{dx(k+2)}{du(k)} = \left\{ I_{nx} - h_{k+2} \left.\frac{\partial f}{\partial x}\right|_{k+2} \right\}^{-1} \frac{dx(k+1)}{du(k)} \tag{5.61}$$

$$\frac{dy(k+2)}{du(k)} = \left\{ \left.\frac{\partial g}{\partial x}\right|_{k+2} \right\} \frac{dx(k+2)}{du(k)} \tag{5.62}$$

Unendo tutte le equazioni precedenti, la derivata a due passi risulta essere:

$$\begin{aligned}\frac{dy(k+2)}{du(k)} = \left\{ \left.\frac{dg}{dx}\right|_{k+2} \right\} \left\{ I_{nx} - h_{k+2} \left.\frac{\partial f}{\partial x}\right|_{k+2} \right\}^{-1} \\ \left\{ I_{nx} - h_{k+1} \left.\frac{\partial f}{\partial x}\right|_{k+1} \right\}^{-1} \left\{ I_{nx} - h_k \left.\frac{\partial f}{\partial x}\right|_k \right\}^{-1} h_k \left.\frac{\partial f}{\partial u}\right|_k\end{aligned} \tag{5.63}$$

che, scritta in accordo alle definizioni date in precedenza, risulta:

$$\frac{dy(k+2)}{du(k)} = C(k+2)A(k+2)A(k+1)A(k)B(k) \tag{5.64}$$

Ripetendo la procedura e generalizzando di conseguenza i risultati, si ha:

$$\frac{dy(k+n)}{du(k)} = C(k+n)\prod_{i=0}^{n} A(k+(n-i))B(k) \tag{5.65}$$

Si può ottenere l'espressione della matrice (5.17) come nella matrice (5.66)

$$\frac{dY}{dU} = \left[\begin{array}{cccc|c} D(1) & 0 & 0 & \cdots & 0 \\ C(2)A(1)B(1) & D(2) & 0 & & 0 \\ C(3)A(2)A(1)B(1) & C(3)B(2) & D(3) & & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ C\left(N_p\right)\left(\prod_{j=N_p-1}^{1} A(j)\right)B(1) & \cdots & \cdots & C\left(N_p\right)A\left(N_p-1\right)B\left(N_p-1\right) & D\left(N_p\right) \end{array}\right] \tag{5.66}$$

Completata la matrice della derivata che permette di calcolare il Gradiente e l'Hessiano della cifra di merito del controllo ottimo, è possibile utilizzarla per ottenere, dalla formula di Newton ricorsiva, l'azione di controllo ottimale lungo l'orizzonte di predizione.

Capitolo 6

MPC da LFT

6.1 Controllo MPC non lineare basato su LFT

L'obiettivo ora è combinare i risultati ottenuti nei capitoli precedenti, adattandoli e modificandoli ove necessario, per ottenere un sistema di controllo completo che operi sulla base del solutore LFT. In questo capitolo, dunque, si ipotizza che il sistema LFT rappresenti fedelmente il funzionamento del sistema sotto controllo, cioè che la taratura dell'LFT eseguita per identificarne i parametri ignoti o soggetti ad errore sia stata effettuata in maniera perfetta secondo il procedimento illustrato nella sezione 3.1. Di conseguenza, il valore di predizione ottenuto dall'LFT di primo stadio sarà per noi, in questo capitolo, equivalente al valore di misura y_{mis}. Questa ipotesi è sensata solo a condizione che l'algoritmo di identificazione adottato per ottenere i parametri incerti converga rapidamente ai valori corretti di stima. In caso contrario, infatti, il valore di $\hat{y}$ su cui si basa la nostra ottimizzazione della variabile di controllo può discostarsi anche molto dal valore di misura, inficiando gran parte del lavoro del controllore.

La soluzione del problema di controllo ottimo su cui si basa il controllo MPC è a sua volta basata sui seguenti elementi:

Optimizer: Ottimizzatore (si veda la (5.12))

$\frac{\partial \mathbf{y}}{\partial u}$**:** Matrice di sensitività (si veda la (5.66))

LFT: Filtri LFT di primo e secondo stadio (5.1),(5.42) e (5.43)

Barrier: Costruttore della Barrier function secondo (4.4)

J: Costruttore della cifra di merito e del suo Gradiente ed Hessiano

Lo schema complessivo è presentato in Figura 6.1.

Nel dettaglio, avremo che i due filtri LFT derivati rispetto ad x e rispetto ad u sono rappresentati in Figura 6.2 ed in Figura 6.3. In entrambi i casi si nota

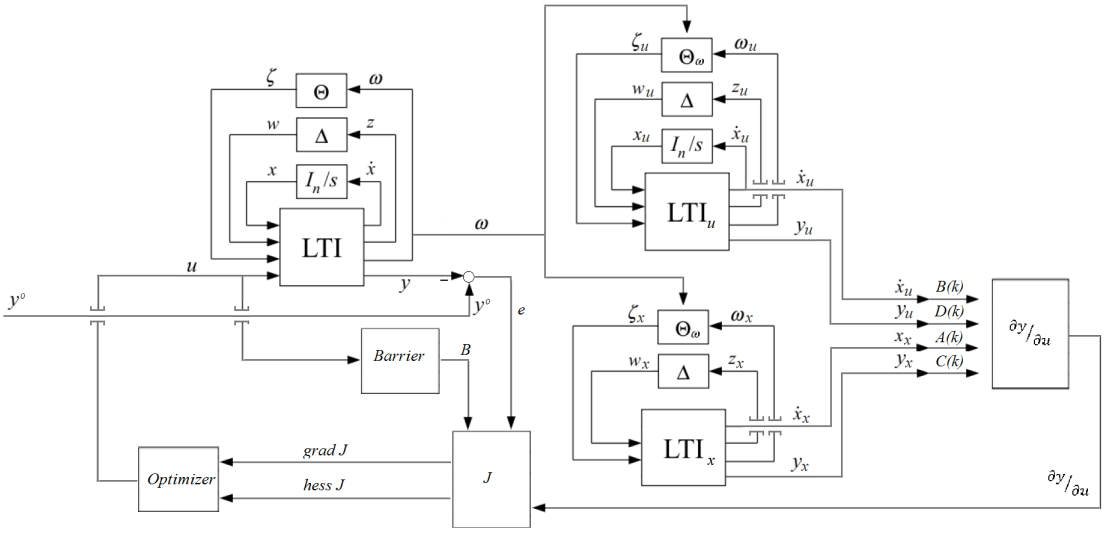

Figura 6.1: Sistema di controllo MPC LFT-based

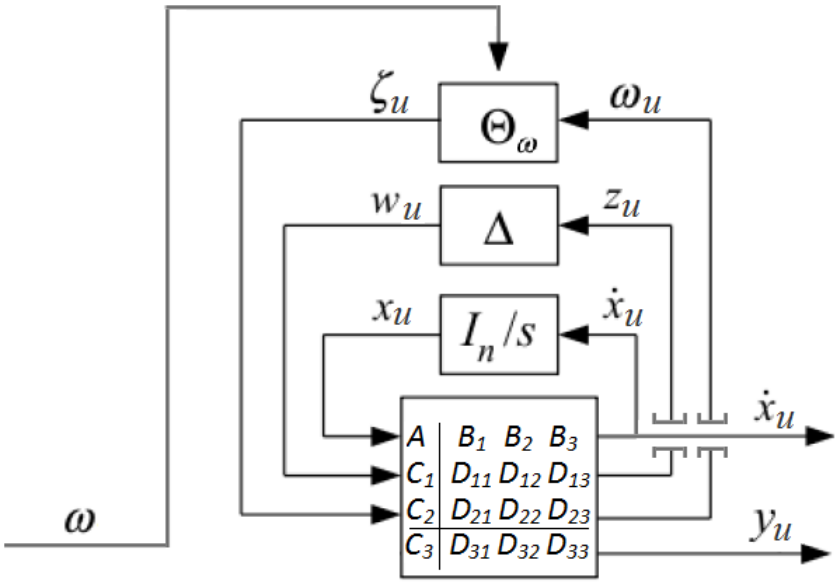

Figura 6.2: Schema LFT derivato rispetto ad u

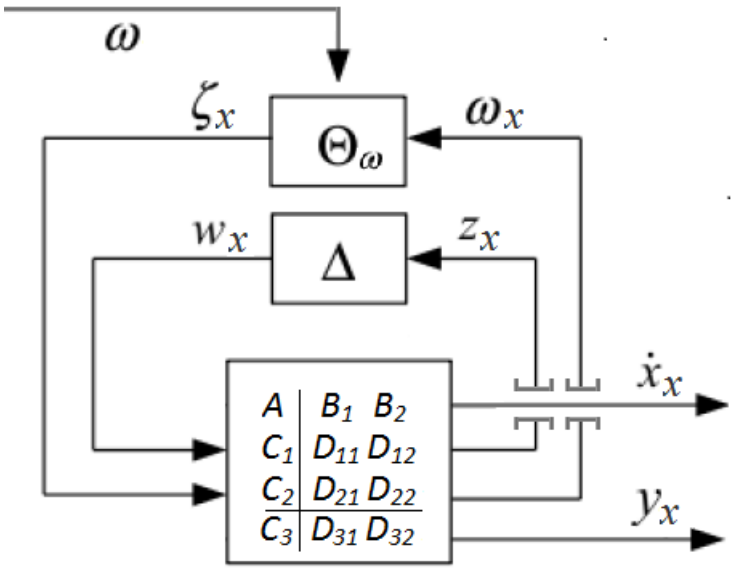

Figura 6.3: Schema LFT derivato rispetto ad x

come i due filtri di sensitività rispetto allo stato e all'ingresso, come ogni schema LFT, siano composti da:

- una parte Lineare Tempo Invariante
- una retroazione contenente le funzioni non lineari note
- una retroazione contenente i parametri identificati

6.2 Gradiente ed Hessiano della cifra di merito completa

Calcolando Gradiente ed Hessiano della (4.3) ed includendo anche le Barrier function si ottiene:

$$\begin{aligned}\nabla J &= \left.\frac{\partial J_o}{\partial u}\right|_{u^{(r)}} + \left.\frac{\partial B}{\partial u}\right|_{u^{(r)}} = \\ &= \frac{1}{N_p}\sum_{j=k+1}^{k+N_p}\left(u^T R + (\hat{y}-y^o)^T Q\left(\left.\frac{\partial \hat{y}}{\partial u}\right|_{u^{(r)}}\right)\right) + \left.\frac{\partial B}{\partial u}\right|_{u^{(r)}} \qquad (6.1)\\ \mathcal{H}(J) &= \left.\frac{\partial^2 J_o}{\partial u^2}\right|_{u^{(r)}} + \left.\frac{\partial^2 B}{\partial u^2}\right|_{u^{(r)}} = \\ &= \frac{1}{N_p}\sum_{j=k+1}^{k+N_p}\left(R + \left(\left.\frac{\partial \hat{y}}{\partial u}\right|_{u^{(r)}}\right)^T Q\left(\left.\frac{\partial \hat{y}}{\partial u}\right|_{u^{(r)}}\right)\right) + \left.\frac{\partial^2 B}{\partial u^2}\right|_{u^{(r)}} \qquad (6.2)\end{aligned}$$

Per comodità, si riporta di seguito l'espressione di B:

$$B(u) = -\ln\left(1-\left(\frac{u-u_{min}}{u_{MAX}-u_{min}}\right)^N\right) - \ln\left(1-\left(\frac{u_{MAX}-u}{u_{MAX}-u_{min}}\right)^N\right)$$

Da cui si ottengono le derivate:

$$\frac{\partial B}{\partial u} = \frac{N\left(\frac{L}{2}-u\right)^{N-1}}{L\left(\left(\frac{L}{2}-u\right)^N-1\right)} - \frac{N\left(\frac{L}{2}+u\right)^{N-1}}{L\left(\left(\frac{L}{2}+u\right)^N-1\right)} \qquad (6.3)$$

$$\frac{\partial^2 B}{\partial u^2} = \frac{N^2\left(\frac{L}{2}-u\right)^{2N-2} - N(N-1)\left(\frac{L}{2}-u\right)^{N-2}}{L^2\left(\left(\frac{L}{2}-u\right)^N-1\right)^2} + \qquad (6.4)$$

$$+ \frac{N^2\left(\frac{L}{2}+u\right)^{2N-2} - N(N-1)\left(\frac{L}{2}+u\right)^{N-2}}{L^2\left(\left(\frac{L}{2}+u\right)^N-1\right)^2} \qquad (6.5)$$

dove

$$L = U_{max} - U_{min} \qquad (6.6)$$

Le espressioni precedenti dipendono esclusivamente da u, da N e dall'intervallo di limitazione scelto per le variabili di controllo. Questa caratteristica permette di calcolare le funzioni (6.3) ed (6.5) indipendentemente dai filtri LFT, e basandosi unicamente sull'azione di controllo corrente da minimizzare. Per questo nello schema di controllo (6.1) é stato rappresentato il sistema di generazione della Barrier function come separato da quello di generazione dell'Hessiano e del Gradiente

della cifra di merito del controllore. Le derivate appena calcolate, se rappresentate nell'intervallo L che esprime il campo di ammissibilità dell'ingresso, possono essere rappresentate graficamente come in Figura 6.4.

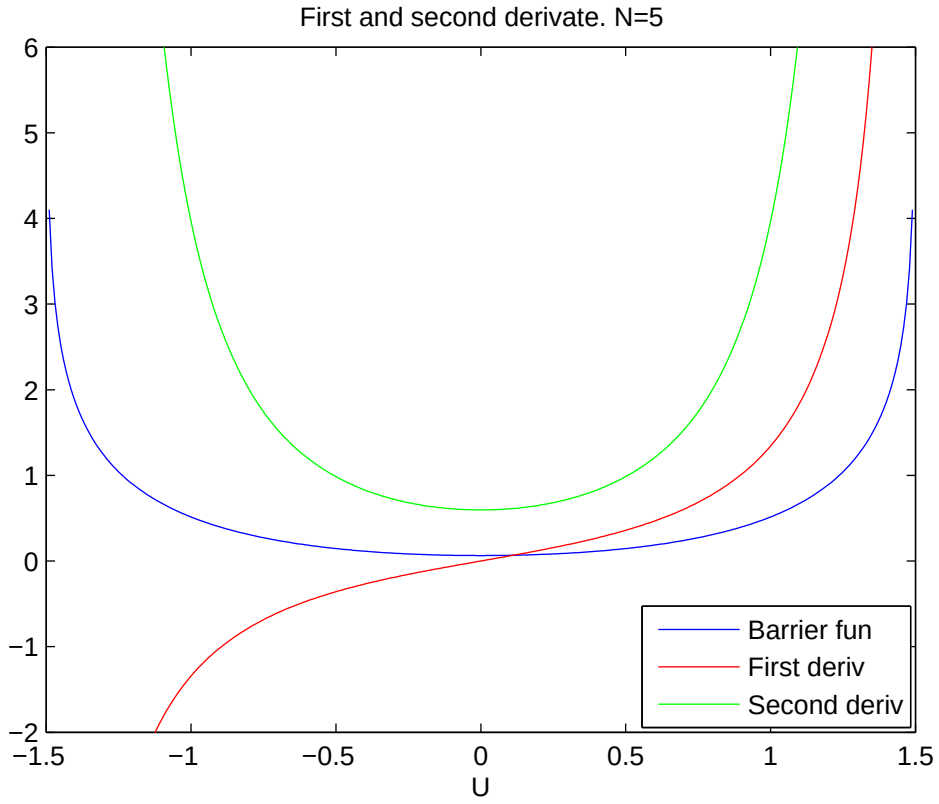

Figura 6.4: Grafici del Gradiente e dell'Hessiano della Barrier function

6.3 Ciclo di identificazione e controllo

In questa sezione si descrive un ciclo di controllo predittivo basato sull'algoritmo iterativo sopra costruito.

Parte1: Costruzione delle matrici del sistema LFT come simulatore di base

Parte2: Costruzione delle matrici che compongono i termini dell'acceleratore (si veda la sezione 6.4)

Parte3: Costruzione e calcolo di tutte le matrici booleane di sparsità

Parte3: Stima iniziale del valore dei parametri incogniti del sistema

Parte4: Simulazione real-time del sistema tramite il filtro LFT; salvataggio termini comuni

Parte5: Ottimizzazione real-time dell'ingresso tramite i filtri di sensitività

Parte6: Applicazione del controllo ottimo ed aggiornamento dell'orizzonte su cui si esegue il controllo

Parte7: Misurazione delle uscite e aggiornamento delle matrici

6.4 Calcolo termini comuni per l'aumento dell'efficienza

In questa sezione si definiscono i termini che possono essere calcolati nel primo filtro LFT del sistema di ottimizzazione del controllo, allo scopo di poter simulare i filtri di sensitività rispetto ad ingressi e stato effettuando semplici interpolazioni e valutazioni puntuali delle espressioni già calcolate in precedenza. Questo é ciò che permette, più di tutto, un considerevole risparmio di tempo per l'ottimizzazione richiesta dal controllore MPC che si vuole ottenere.

A questo proposito si inizi considerando i filtri LFT di sensitività ricavati nelle sezioni precedenti (5.42) e (5.43).
Raggruppando z e ω si ha:

$$\begin{cases} \dot{x}_u & = Ax_u + B_1\Delta z_u + B_2\Theta_\omega\omega_u + B_3 \\ \begin{bmatrix} z_u \\ \omega_u \end{bmatrix} & = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} x_u + \begin{bmatrix} D_{11} \\ D_{21} \end{bmatrix} \Delta z_u + \begin{bmatrix} D_{12} \\ D_{22} \end{bmatrix} \Theta_\omega\omega_u + \begin{bmatrix} D_{31} \\ D_{32} \end{bmatrix} \\ y_u & = C_3x_u + D_{31}\Delta z_u + D_{32}\Theta_\omega\omega_u + D_{33} \end{cases} \tag{6.7}$$

$$\begin{cases} \dot{x}_x & = A + B_1\Delta z_x + B_2\Theta_\omega\omega_x \\ \begin{bmatrix} z_x \\ \omega_x \end{bmatrix} & = \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} + \begin{bmatrix} D_{11} \\ D_{21} \end{bmatrix} \Delta z_x + \begin{bmatrix} D_{12} \\ D_{22} \end{bmatrix} \Theta_\omega\omega_x \\ y_x & = C_3 + D_{31}\Delta z_x + D_{32}\Theta_\omega\omega_x \end{cases} \tag{6.8}$$

E quindi:

$$\begin{bmatrix} z_u \\ \omega_u \end{bmatrix} = \begin{bmatrix} I_z - D_{11}\Delta & -D_{12}\Theta_\omega \\ -D_{21}\Delta & I_\omega - D_{22}\Theta_\omega \end{bmatrix}^{-1} \left(\begin{bmatrix} C_1 \\ C_2 \end{bmatrix} x_u + \begin{bmatrix} D_{31} \\ D_{32} \end{bmatrix} \right) \tag{6.9}$$

$$\begin{bmatrix} z_x \\ \omega_x \end{bmatrix} = \begin{bmatrix} I_z - D_{11}\Delta & -D_{12}\Theta_\omega \\ -D_{21}\Delta & I_\omega - D_{22}\Theta_\omega \end{bmatrix}^{-1} \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} \tag{6.10}$$

Definendo:

$$\Gamma_1(\omega) \triangleq \begin{bmatrix} I_z - D_{11}\Delta & -D_{12}\Theta_\omega \\ -D_{21}\Delta & I_\omega - D_{22}\Theta_\omega \end{bmatrix}^{-1} \tag{6.11}$$

Con semplici calcoli si ottengono i sistemi equivalenti:

$$\begin{cases} \dot{x}_u &= \begin{bmatrix} A + \begin{bmatrix} B_1\Delta & B_2\Theta_\omega \end{bmatrix} \Gamma_1(\omega) \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} & B_3 + \begin{bmatrix} B_1\Delta & B_2\Theta_\omega \end{bmatrix} \Gamma_1(\omega) \begin{bmatrix} D_{31} \\ D_{32} \end{bmatrix} \end{bmatrix} \begin{bmatrix} x_u \\ I \end{bmatrix} \\ y_u &= \begin{bmatrix} C_3 + \begin{bmatrix} D_{31}\Delta & D_{32}\Theta_\omega \end{bmatrix} \Gamma_1(\omega) \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} & D_{33} + \begin{bmatrix} D_{31}\Delta & D_{32}\Theta_\omega \end{bmatrix} \Gamma_1(\omega) \begin{bmatrix} D_{31} \\ D_{32} \end{bmatrix} \end{bmatrix} \begin{bmatrix} x_u \\ I \end{bmatrix} \end{cases} \tag{6.12}$$

$$\begin{cases} \dot{x}_x &= A + \begin{bmatrix} B_1\Delta & B_2\Theta_\omega \end{bmatrix} \Gamma_1(\omega) \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} \\ y_x &= C_3 + \begin{bmatrix} D_{31}\Delta & D_{32}\Theta_\omega \end{bmatrix} \Gamma_1(\omega) \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} \end{cases} \tag{6.13}$$

Chiamando quindi, analogamente a quanto già fatto nella sezione 3.2:

$$\Gamma_2(\omega) \triangleq A + \begin{bmatrix} B_1\Delta & B_2\Theta_\omega \end{bmatrix} \Gamma_1(\omega) \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} \tag{6.14}$$

$$\Gamma_3(\omega) \triangleq B_3 + \begin{bmatrix} B_1\Delta & B_2\Theta_\omega \end{bmatrix} \Gamma_1(\omega) \begin{bmatrix} D_{31} \\ D_{32} \end{bmatrix} \tag{6.15}$$

$$\Gamma_4(\omega) \triangleq C_3 + \begin{bmatrix} D_{31}\Delta & D_{32}\Theta_\omega \end{bmatrix} \Gamma_1(\omega) \begin{bmatrix} C_1 \\ C_2 \end{bmatrix} \tag{6.16}$$

$$\Gamma_5(\omega) \triangleq D_{33} + \begin{bmatrix} D_{31}\Delta & D_{32}\Theta_\omega \end{bmatrix} \Gamma_1(\omega) \begin{bmatrix} D_{31} \\ D_{32} \end{bmatrix} \tag{6.17}$$

si ottiene che i sistemi (6.12) e (6.13) possono essere riscritti come:

$$\begin{cases} \dot{x}_u &= \Gamma_2(\omega)\, x_u + \Gamma_3(\omega) \\ y_u &= \Gamma_4(\omega)\, x_u + \Gamma_5(\omega) \end{cases} \tag{6.18}$$

$$\begin{cases} \dot{x}_x &= \Gamma_2(\omega) \\ y_x &= \Gamma_4(\omega) \end{cases} \tag{6.19}$$

Con questo procedimento è dunque possibile ottenere, dalle matrici del sistema LFT di primo stadio e dalla funzione derivata di Θ rispetto ad ω, i valori dei filtri LFT per la sensitività rispetto all'ingresso e allo stato. Da questi si mostra (si veda la sezione 5.4) come sia possibile ottenere, direttamente in uscita, le derivate che, valutate nell'istante corrente di integrazione (secondo il procedimento delineato nella sezione 5.2), permettono di ottenere i valori da inserire nella matrice (5.17) usata per ottenere il Gradiente e l'Hessiano della cifra di merito da minimizzare seguendo la Formula di Newton (5.12).

6.5 Estensione Multiple Input Multiple Output

L'analisi effettuata nei capitoli 5 e 6 in forma SISO può facilmente essere estesa in forma MIMO, nel modo affrontato in questa sezione.

Ipotizzando di voler effettuare un controllo su un sistema con:

- m ingressi, di cui m_c manipolabili e m_d disturbi,
- p uscite misurate
- n variabili di stato

Se consideriamo di definire:

$$U(k) = \begin{bmatrix} u_1(k) & u_2(k) & \dots & u_m(k) \end{bmatrix} \tag{6.20}$$
$$X(k) = \begin{bmatrix} x_1(k) & x_2(k) & \dots & x_n(k) \end{bmatrix} \tag{6.21}$$
$$Y(k) = \begin{bmatrix} y_1(k) & y_2(k) & \dots & y_p(k) \end{bmatrix} \tag{6.22}$$
$$Y_o(k) = \begin{bmatrix} y_1^o(k) & y_2^o(k) & \dots & y_p^o(k) \end{bmatrix} \tag{6.23}$$

dove $u_m(k)$ si riferisce, naturalmente, al valore dell'ingresso m-esimo nell'istante k, e si sono inseriti negli ingressi u al sistema sia le variabili manipolabili (e quindi rispetto cui è possibile eseguire un'ottimizzazione), sia gli ingressi di disturbo valutati all'istante corretto.
Si ottengono così i seguenti vettori:

$$U = \begin{bmatrix} U(k) & U(k+1) & \cdots & U\left(k+N_p\right) \end{bmatrix} \tag{6.24}$$
$$X = \begin{bmatrix} X(k) & X(k+1) & \cdots & X\left(k+N_p\right) \end{bmatrix} \tag{6.25}$$
$$Y = \begin{bmatrix} Y(k) & Y(k+1) & \cdots & Y\left(k+N_p\right) \end{bmatrix} \tag{6.26}$$
$$Y^o = \begin{bmatrix} Y^o(k) & Y^o(k+1) & \cdots & Y^o\left(k+N_p\right) \end{bmatrix} \tag{6.27}$$

Di conseguenza, il sistema generico a tempo discreto si può scrivere come:

$$\begin{cases} X(k+1) & = F\left(X(k), U(k)\right) \\ Y(k) & = G\left(X(k), U(k)\right) \end{cases} \tag{6.28}$$

La cifra di merito del controllo ottimo (5.9) viene quindi scritta, equivalentemente:

$$J_o = \frac{1}{2}\frac{1}{N}\left(U^T R U + \left(Y - Y^o\right)^T Q \left(Y - Y^o\right)\right) \tag{6.29}$$

dove R e Q sono due matrici diagonali composte da N matrici R e Q rispettivamente:

$$R = \begin{bmatrix} R & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & R \end{bmatrix} \qquad Q = \begin{bmatrix} Q & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & Q \end{bmatrix} \tag{6.30}$$

Con il medesimo procedimento già descritto nella sezione 5.1 si può facilmente ottenere:

$$\nabla J_o = \frac{1}{N}\left[U^T R + (Y - Y^o)^T Q \left(\left.\frac{\partial Y}{\partial U}\right|_{U^r}\right)\right] \tag{6.31}$$

$$H\left(J_p\right) = \frac{1}{N}\left[R + \left(\left.\frac{\partial Y}{\partial U}\right|_{U^r}\right)^T Q \left(\left.\frac{\partial Y}{\partial U}\right|_{U^r}\right)\right] \tag{6.32}$$

Ancora una volta, solo con la derivata di Y rispetto ad U possiamo ottenere ciò che ci serve per la minimizzazione della cifra di merito del controllo rispetto alle variabili manipolabili.

La derivata ha la ormai ben nota espressione:

$$\frac{\partial Y}{\partial U} = \begin{bmatrix} \frac{\partial Y(k)}{\partial U(k))} & 0_{pxm} & \cdots & 0_{pxm} \\ \frac{\partial Y(k+1)}{\partial U(k)} & \frac{\partial Y(k+1)}{\partial U(k+1)} & & \vdots \\ \vdots & & \ddots & 0_{pxm} \\ \frac{\partial Y(k+N_p)}{\partial U(k)} & \cdots & \cdots & \frac{\partial Y(k+N_p)}{\partial U(k+N_p)} \end{bmatrix} \tag{6.33}$$

Si noti quindi che ogni elemento della (6.33) è a sua volta una matrice dall'espressione:

$$\frac{\partial Y\left(k_i\right)}{\partial U\left(k_j\right)} = \begin{bmatrix} \frac{\partial y_1(k_i)}{\partial u_1(k_j)} & \frac{\partial y_1(k_i)}{\partial u_2(k_j)} & \cdots & \frac{\partial y_1(k_i)}{\partial u_m(k_j)} \\ \frac{\partial y_2(k_i)}{\partial u_1(k_j)} & \frac{\partial y_2(k_i)}{\partial u_2(k_j)} & & \\ \vdots & & \ddots & \vdots \\ \frac{\partial y_p(k_i)}{\partial u_1(k_j)} & & \cdots & \frac{\partial y_p(k_i)}{\partial u_m(k_j)} \end{bmatrix} \tag{6.34}$$

Capitolo 7

Applicazione: Controllo MPC di un quad

In quest'ultimo capitolo del presente lavoro si affronta il problema di guida senza pilota di un veicolo tramite il controllo MPC delineato nei capitoli precedenti. Si è scelto questo problema perché si tratta di un caso di studio che:

- È un sistema fortemente non lineare
- Fa parte dell'insieme dei problemi che richiedono un'elevata frequenza di esecuzione del controllo
- È un esempio pratico e di grande attualità
- Ha come obiettivo del controllo una traiettoria continua nel tempo; si tratta di un problema di inseguimento, quindi più complicato del semplice mantenimento di un sistema nell'intorno di una posizione di equilibrio.
- Il sistema possiede 5 variabili di stato, 2 variabili di ingresso e 2 di uscita. Ci permette quindi di testare il funzionamento del controllo costruito su un problema multivariabile, mantenendo però allo stesso tempo un livello di complessità abbastanza basso da permettere un'analisi efficace.

7.1 Modello del sistema single-track

Il sistema che si desidera controllare è il QUAD (un veicolo a quattro ruote progettato per andare prevalentemente fuoristrada) attualmente a disposizione del Dipartimento di Elettronica del Politecnico di Milano, in cui è stato scritto questo elaborato. Per eseguire il controllo di traiettoria si è sviluppato un modello semplificato single-track (comunemente chiamato "modello bicicletta") con

parametri identificati in precedenza. Il modello in questione risulta essere:

$$\begin{cases} mv\cos\beta(\dot{\beta}+\dot{\psi}) & = F_{yf}+F_{yr} \\ I_z\ddot{\psi} & = aF_{y_f}-bF_{y_r}+M_z \\ F_{yf} & = C_f\alpha_f \\ F_{yr} & = C_r\alpha_r \\ \alpha_f & = \delta-(\beta+\frac{a\dot{\psi}}{v\cos\beta}) \\ \alpha_r & = -\beta+\frac{b\dot{\psi}}{v\cos\beta} \end{cases} \tag{7.1}$$

E può essere riscritto come:

$$\begin{cases} \dot{\beta} & = \frac{F_{y_f}+F_{y_r}}{mv\cos\beta}-\dot{\psi} \\ \ddot{\psi} & = \frac{aF_{y_f}-bF_{y_r}+M_z}{I_z} \\ \dot{\psi} & = \int_0^t \ddot{\psi}dt \\ \dot{x} & = v\cos(\psi-\beta) \\ \dot{y} & = v\sin(\psi-\beta) \end{cases} \tag{7.2}$$

Si noti che il modello costruito è non lineare, con due parametri (C_f e C_r) da stimare e dipendenti dalle condizioni della strada (si tratta dei coefficienti di aderenza delle gomme alla superficie). La rappresentazione del modello single-track si può trovare in Figura 7.1

7.2 Inseguimento di traiettoria

Il problema che si vuole risolvere è un problema di controllo ottimo su una traiettoria. Si desidera, cioè, che il veicolo rappresentato dal modello bicicletta segua un riferimento di traiettoria agendo sulle variabili di controllo velocità v e sterzo δ. Si sono scelti come valori obiettivo $v = 10$m/s e sterzo $\delta = 0$. Il primo è il valore tipico di utilizzo su molti simulatori di guida. Il secondo invece indica la richiesta al sistema di mantenere lo sterzo il più dritto possibile.

La traiettoria viene data in ingresso, nel nostro caso, tramite il risultato di una simulazione Dymola eseguita sul modello single-track e con ingressi di velocità pari a quella nominale di 10 m/s, e sterzo tale da far compiere al modello single-track una figura a forma di S (sterzo $\delta = 0.1\sin t$). Si ottiene così una traiettoria del tipo in Figura 7.2

Il controllo ottimo che si è costruito in questo elaborato ha l'obiettivo di mantenere il veicolo in traiettoria anche in presenza di disturbi. Eseguendo un controllo ottimo in un tratto di curva, alla fine del rettilineo d'inizio, si vuole mostrare come il controllo ottimizzi gli ingressi ed ottenga valori di uscita di simulazione che si mantengono più vicini alla traiettoria richiesta, applicando dunque un effetto correttivo su sterzo e velocità del veicolo.

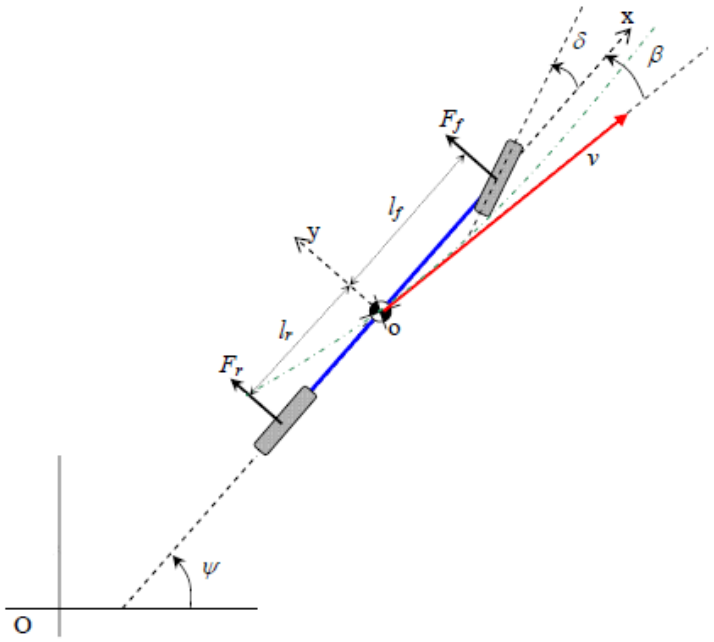

Figura 7.1: Modello single-track

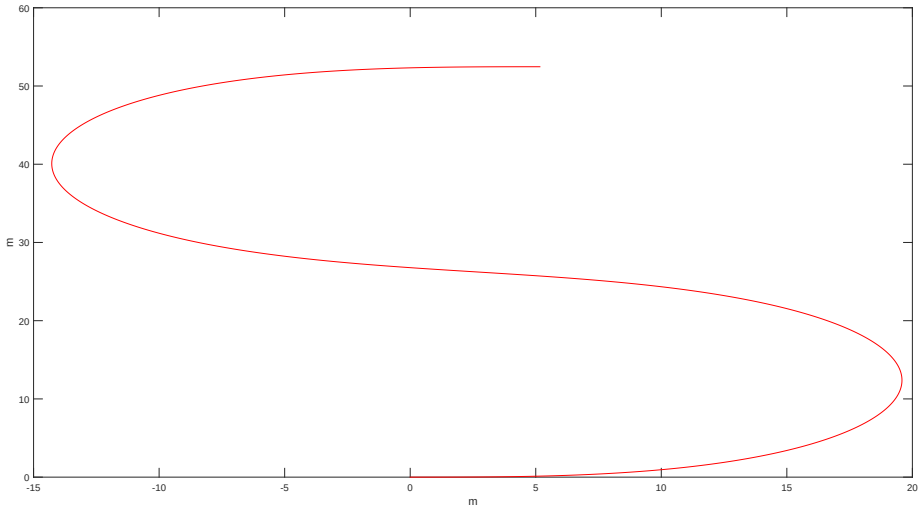

Figura 7.2: Traiettoria obiettivo a forma di S

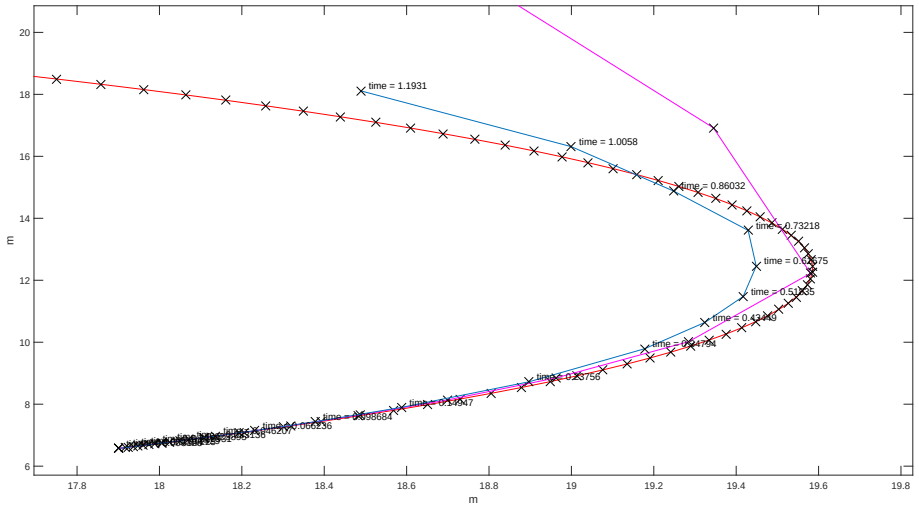

Figura 7.3: Porzione di MPC: controllo ottimo sulla traiettoria

Per testare il controllo eseguito si è compiuto un controllo ottimo su una porzione della traiettoria mostrata in Figura 7.2, e precisamente negli istanti da $t = 2$ a $t = 4$. L'obiettivo del test è mostrare come il nostro sistema di controllo scelga in autonomia il passo da compiere e lo allunghi in rettilineo, per poi ridurlo in prossimità della curva. Si ottiene la traiettoria in Figura 7.3 in cui si indica in rosso la porzione di traiettoria obiettivo, in magenta la traiettoria che si otterrebbe con lo sterzo e la velocità iniziale soggetti a disturbo, e in blu la traiettoria che si ottiene tramite il controllo ottimo. I tempi che sono indicati in figura sono i tempi dei passi scelti dal solutore. L'orizzonte di controllo, settato a 20 passi, conduce ad un tempo di fine controllo di 1.19 sec, dove il tempo di esecuzione del controllo è invece di circa 0.08 sec. Ciò indica come il metodo sia eseguibile in tempo reale per ottenere un controllo NMPC soddisfacente nel tempo.

7.2.1 Uso di Eulero all'indietro per simulazione

Iniziando lo sviluppo del controllore utilizzando il solutore di Matlab per problemi stiff (ode15s), eseguendo come test la curva a sinistra analoga alla precedente, si ottiene il grafico di controllo indicato in Figura 7.4.

Si può notare facilmente l'esecuzione di un elevato numero di passi molto corti all'inizio. Questi passi servono ad ode15s per raggiungere l'ordine a regime. Si veda il paragrafo 2.4.5. Considerando nella cifra di merito i campioni relativi a questi tratti, si ottiene un numero di condizionamento dell'Hessiano elevato, che si rispecchia in una difficoltà di convergenza all'ottimo. Inoltre il tempo di

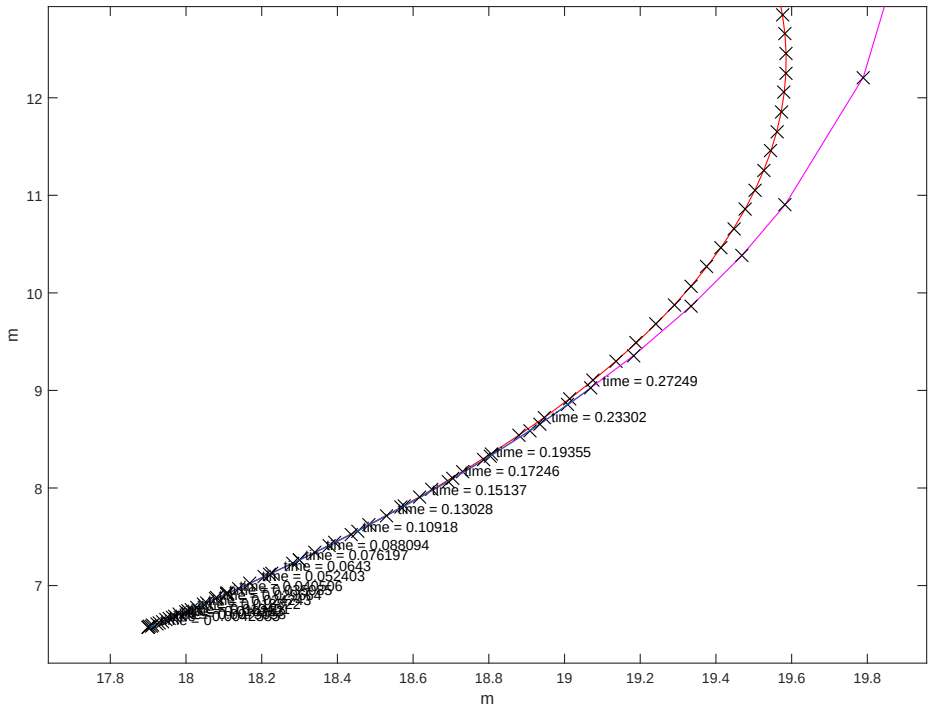

Figura 7.4: Curva a sinistra con ode15s nel solutore

esecuzione del controllo risulta essere di circa 1 secondo, al di sopra del limite di real time per il problema in questione.

É dunque preferibile optare per metodi a passo singolo. In merito, si è scelto Eulero all'indietro per la risoluzione dello stiffness, mentre per quanto riguarda il trattamento del problema DAE di indice 1, costituito dalla simulazione del filtro base, si è scelto di derivare i vincoli e di implementare Eulero all'indietro per sistemi ODE (anziché Eulero all'indietro per sistemi DAE) rendendo embedded il calcolo del Common Term più oneroso.
Così facendo, nel medesimo tratto, sempre con una finestra temporale di 20 passi di soluzione, si ottiene il controllo mostrato in Figura 7.3, con un numero di iterazioni nettamente inferiore per la ricerca dell'ottimo. Come analizzato precedentemente, il tempo di esecuzione è tale da permettere un controllo real-time e una migliore convergenza. I passi eseguiti, inoltre, permettono di coprire in 20 passi di soluzione tutta la curva.

7.2.2 Impostazioni del sistema di controllo

Nel caso preso ad esempio si sono utilizzate le impostazioni seguenti:

Matrici di peso Q ed R sono state settate come matrici identità, con la matrice Q moltiplicata di un fattore 10 rispetto alla matrice R, unitaria. Questo è stato fatto per ottenere un controllo che prediliga l'aderenza alla traiettoria rispetto alla modifica del controllo richiesto.

Orizzonte di controllo Si è utilizzato un orizzonte di controllo di 20 passi, che consente di mantenere lo sforzo computazionale per l'inversione delle matrici componenti l'Hessiano della cifra di merito ad un valore accettabile.

Coefficienti polinomiali Il metodo costruito (si veda in proposito l'Appendice B) lavora a tempo continuo, utilizzando non dei valori ad un campionamento fisso ma una funzione interpolante di cui ottimizzare i coefficienti. In questo caso abbiamo considerato che 5 coefficienti fossero un valore accettabile per rappresentare curve anche complesse senza inficiare l'efficienza del metodo.

Valori di tolleranza Si sono considerati come valori di tolleranza per la traiettoria valori di ordine relativo di 10^{-3},per ottenere una prestazione migliore del controllore.

7.2.3 Risultati ottenuti

Dalla Figura 7.3 si possono notare diverse caratteristiche del controllore sviluppato:

Inseguimento della traiettoria È facile notare che la traiettoria eseguita con l'input iniziale (con il disturbo inserito) diverge dalla traiettoria obiettivo proprio durante la curva. La traiettoria eseguita dal controllo, invece, anticipa la curva e la esegue correggendo i valori di input iniziali.

Esecuzione dei passi Osservando la Figura 7.5 si può notare facilmente come i passi eseguiti dal solutore si allunghino inizialmente, raggiungendo una condizione di regime all'istante $t = 0.10741$. A questo punto i passi, in rettilineo, si allungano fino all'istante $t = 0.41247$. Qui si è entrati in curva e di conseguenza il solutore stringe il passo (si nota il restringimento tra $t = 0.49349$ e $t = 0.66806$), per poi riallargarlo al passo finale.

Precisione del solutore È facile notare come la soluzione ottima si trovi all'interno della curva ideale. Si noti tuttavia come questo comportamento sia da imputare principalmente alla perturbazione dell'ingresso. Questa caratteristica si può vedere ancora meglio se si simula la successiva curva a destra, tra gli istanti $t = 7$ e $t = 8$. Si ottiene infatti la Figura 7.6. L'effetto della perturbazione negativa dell'angolo di sterzo è di allargare la curva a sinistra e di stringere quella a destra. Ciò significa che le direzioni di approccio all'ottimo sono differenti nei due casi; gli errori di approssimazione

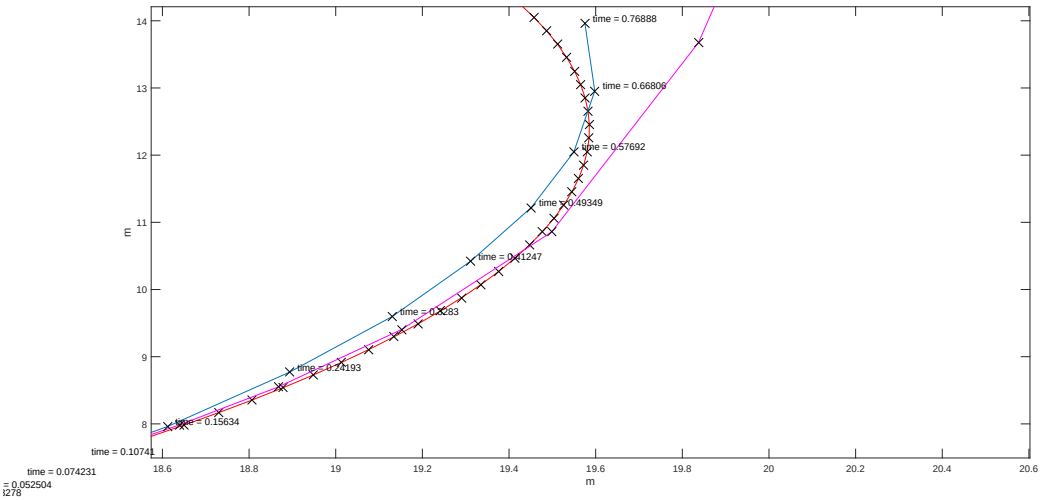

Figura 7.5: Ingrandimento della finestra di controllo precedente

della traiettoria finale (differenti nei due casi) sono dovuti alla condizione di arresto dell'ottimizzatore combinata con l'ordine basso del metodo a passo singolo per ora implementato.

Infine, in Figura 7.7, si mostra come evolvono gli ingressi del sistema sotto l'ottimizzazione effettuata. In magenta, come in precedenza, si indicano i valori degli ingressi iniziali; in blu si indicano i valori degli ingressi ottimi. Si può notare come la velocità venga aumentata all'inizio, diminuita in esecuzione di curva e aumentata nuovamente verso l'uscita (superato il punto di corda). L'angolo di sterzo viene leggermente ridotto (in valore assoluto).

Si è infine emulato manualmente un passo di controllo MPC. Per fare ciò si sono utilizzati i valori ottimi trovati al passo precedente come inizializzazione del passo successivo di controllo ottimo. In Figura 7.8 si nota come, così facendo, i passi iniziali del controllo risultano essere molto più lunghi, dimostrando quindi la possibilità di eseguire il controllo MPC a partire dal controllo ottimo continuo da ogni iterazione di controllo ottimo precedente.

7.3 Conclusioni dell'esperimento

Il controllo ottimo applicato al problema di controllo della traiettoria considerato mette in luce alcune caratteristiche salienti dell'algoritmo costruito, che indicano come l'idea di eseguire un controllo ottimo (e di conseguenza MPC non lineare) utilizzando l'approccio LFT possa essere valida in termini di riduzione del tempo computazionale e di riutilizzo dei calcoli per semplificare il problema di MPC non lineare rendendolo risolvibile on-line anche in casi (come quello considerato) in

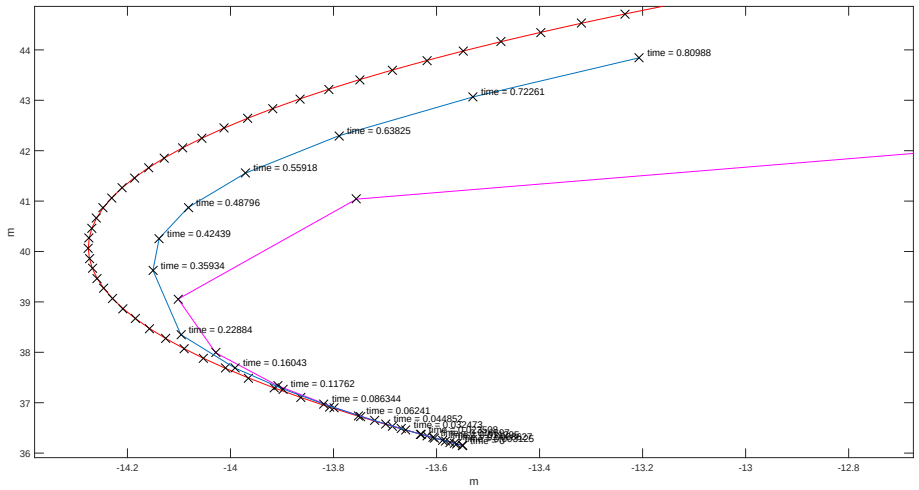

Figura 7.6: Curva a destra

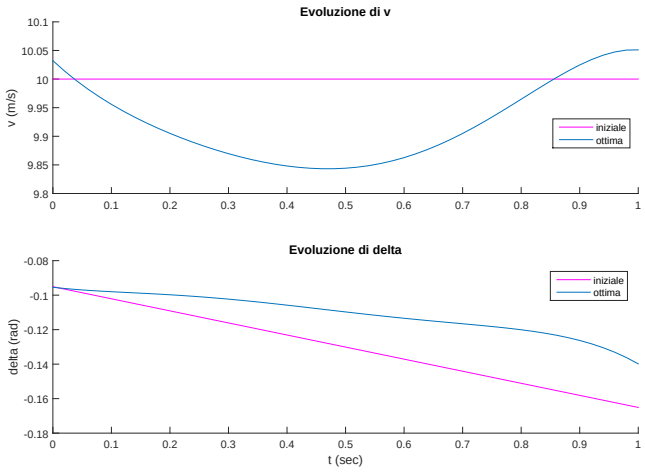

Figura 7.7: Evoluzione dei valori degli ingressi

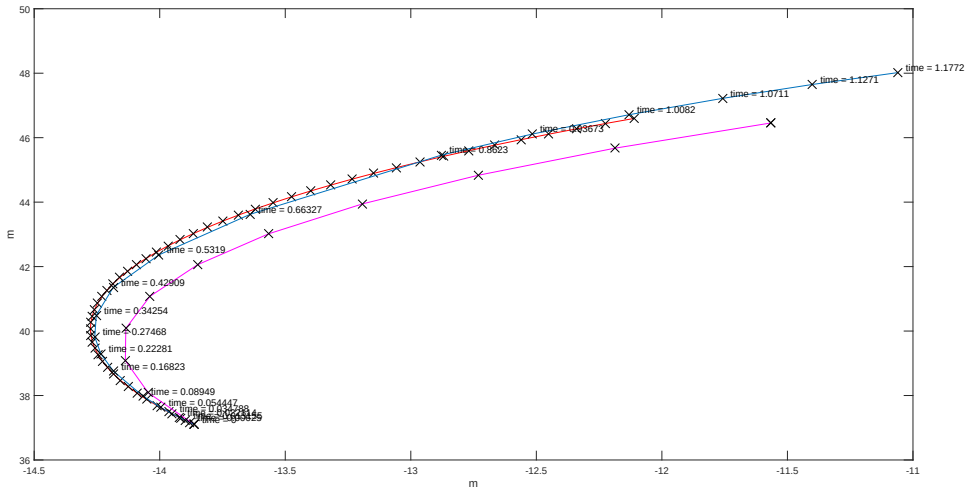

Figura 7.8: Curva a destra, a partire da 7.1 sec

cui il modello su cui eseguire MPC sia fortemente non lineare. Il controllo ottimo ottenuto converge e conduce la traiettoria verso quella ideale. Le differenze ancora visibili dipendono principalmente dall'ordine del metodo di simulazione.

Capitolo 8

Conclusioni e Sviluppi futuri

Nel presente lavoro si è ottenuto, a partire dalla rappresentazione LFT di un sistema non-lineare, un algoritmo efficiente di simulazione e controllo MPC che può essere eseguito in tempo reale su un problema a frequenza elevata, come è il controllo di traiettoria di un veicolo. Si è sviluppato un toolbox (descritto nell'Appendice B) quasi completo, che promette di essere estendibile per ottenere un controllo MPC non lineare completo, con possibilità di creare vincoli in modo nativo.

8.1 Risultati ottenuti

Il lavoro svolto in questo elaborato ha mostrato l'ottenimento dei risultati seguenti:

- ✓ Dimostrazione della fattibilità di implementazione di un MPC non lineare basato sulle proprietà della forma LFT. Infatti le derivate per la costruzione di Hessiano e gradiente non vengono approssimate per differenze finite ma determinate secondo dei filtri simbolici esatti
- ✓ Data la struttura dati relativa alla forma LFT di un modello, costruzione automatica dei filtri di sensitività rispetto agli ingressi da ottimizzare
- ✓ Dimostrazione e ricavo di un metodo semplice per integrare i vincoli sulle variabili (di controllo e di stato) nella cifra di merito del controllo
- ✓ Sviluppo complessivo di metodologie per l'aumento dell'efficienza di calcolo, come per esempio, tra i più importanti:

  - L'impiego del passo variabile per concentrare lo sforzo di calcolo quando/dove le dinamiche diventano rapide e non costringere più alla scelta di un periodo di campionamento fisso.

- La simulazione embedded dei filtri di sensitività grazie alle proprietà della forma LFT.

- Lo sfruttamento delle informazioni di sparsità per rendere ancora più efficiente la simulazione embedded.

- Altre strategie minori per ridurre il peso computazionale nella simulazione durante l'ottimizzazione.

✓ Costruzione di un toolbox completo per esecuzione di controllo ottimo, facilmente estendibile per l'esecuzione di controllo MPC non lineare

✓ Test del toolbox su un problema attuale e significativo, con esito positivo

Il presente elaborato, ed il lavoro svolto, vogliono essere un primo passo significativo in direzione della realizzazione di un nuovo metodo di controllo MPC non lineare che, affrancato dalla necessità di simulazioni non lineare complicate o di troncamenti grossolani delle stesse, possa portare all'uso del controllo MPC anche in campi e sistemi non lineari richiedenti un'elevata efficienza di computazione.

8.2 Sviluppi ancora in corso del toolbox

Il toolbox sviluppato per questo elaborato è aperto in particolare a:

- L'integrazione dei vincoli tramite la Barrier function (si vedano il capitolo 4 e la sezione 6.2). Nonostante in 4 si determini la Barrier Function di Frisch adatta per il problema e in 6.2 si calcolino gradiente ed Hessiano della cifra di merito allargata inserendo anche la Barrier Function, nel toolbox non si è inserito, per semplicità, l'uso di questa funzione. È un miglioramento di facile ed immediata applicazione. È infatti sufficiente sostituire il calcolo della cifra di merito, del gradiente e dell'Hessiano con le loro corrispondenti allargate.

- L'implementazione di metodi a passo singolo e ordine superiore. In merito si può adottare un metodo di Runge-Kutta embedded per problemi stiff (applicato sempre al sistema ODE ottenuto derivando i vincoli del DAE). Questo permette di eseguire passi singoli più ampi, mantenendo un'elevata accuratezza.

- L'uso della sparsità nella costruzione della matrice triangolare bassa di sensitività (5.17). Di non così immediata applicazione è l'espansione dell'uso della sparsità (si veda l'Appendice A in proposito). Attualmente questa è

costruita insieme alla forma LFT, e viene applicata nel calcolo dei termini comuni, che vengono passati al solutore per rendere sensibilmente più efficiente l'integrazione dei filtri LFT di secondo stadio nelle simulazioni embedded. Dato che il calcolo della matrice triangolare di sensitività presenta una struttura fissa, potrebbe essere determinata ed usata la sparsità anche in questo campo.

- Il trattamento di strutture LFT non complete (per esempio senza funzioni non lineari e/o senza blocchi Δ)
- Costruzione automatica della struttura dati LFT a partire da file XML prodotti da altri ambienti di programmazione ad oggetti (per esempio Dymola o OpenModelica).

Appendice A

Matrici di Sparsità

A.1 Definizione

Una matrice di sparsità è definita operativamente nel seguente modo:
Data una matrice qualsiasi di numeri reali:

1. Costruire una matrice booleana di valori FALSE delle medesime dimensioni
2. Per ogni suo elemento:
   (a) se è diverso da zero, porre il corrispondente elemento a TRUE
   (b) altrimenti, lasciare il corrispondente elemento a FALSE

Alla fine di questo procedimento, si ottiene una matrice booleana, della stessa dimensione della matrice di partenza, che sia TRUE dove il corrispondente elemento della matrice di partenza ha valore diverso da zero, e che sia FALSE altrimenti. Ad esempio la matrice di sparsità di:

$$M = \begin{bmatrix} 2 & 1 & 0 & -2 \\ 0 & 0 & 3 & 0.2 \\ -6 & 5 & 0 & 0 \\ 0 & 1 & 4 & 1.2 \\ 0 & 2 & 3.5 & 6 \end{bmatrix}$$

sarà:

$$S_p(M) = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

Il codice Matlab che può essere utilizzato per costruire la matrice di sparsità di una matrice M è illustrato nel Listato seguente.

```
function [Mb] = sparsity_matrix(M)
%SPARSITY_MATRIX Creates the sparsity matrix of M
%   The sparsity matrix is a boolean matrix with
%   'TRUE' where M has element not equal to zero;
%   and 'FALSE' instead. Very useful if M is sparse!!
inds=find(M);
Mb=false(size(M));
Mb(inds)=true; %#ok<FNDSB>
end
```

A.2 Utilizzo

Le matrici di sparsità sono molto utili per determinare dove valutare le matrici cui si riferiscono, in particolare se queste sono matrici sparse. Su una matrice di dimensione 27x20, ad esempio, che ha tuttavia non più di 50 elementi non nulli (e quindi un rapporto di sparsità inferiore al 10%), in termini di tempo computazionale é un grandissimo vantaggio poter valutare la matrice soltanto in quegli elementi che siano diversi da zero; questo accorgimento puó rendere l'esecuzione del codice fino a dieci volte più rapida.

A.3 Definizione del prodotto di matrici

Il prodotto di due matrici di sparsità é la matrice di sparsità del prodotto riga per colonna delle due matrici, esclusi i casi di cancellazioni dovuti somme con valori negativi. Si tratta dunque del prodotto riga per colonna di due matrici considerato nel caso più generico, dove cioè non vi siano azzeramenti dovuti a somme algebriche di opposti.

$$S_{prod} \triangleq prodBoolean\left(S_p\left(M_1\right),\ S_p\left(M_2\right)\right) \tag{A.1}$$

In particolare, quindi:

$$\begin{cases} S_{prod}(i,j) = TRUE & se\ \left(M_1 * M_2\right)(i,j) \neq 0 \\ S_{prod}(i,j) = TRUE & se\ \left(\left(M_1 * M_2\right)(i,j) = 0\ e\ cancellazioni\right) \\ S_{prod}(i,j) = FALSE & altrimenti \end{cases} \tag{A.2}$$

Di conseguenza il prodotto riga per colonna di due matrici di sparsità (*prodBoolean*( )) non può essere definito come il prodotto riga per colonna di due matrici qualsiasi.

Diviene quindi importante ridefinire il prodotto in modo che la proprietà (A.2) sia rispettata. Per ottenere questo effetto è necessario seguire una logica booleana del prodotto righe per colonne, che richiede di tradurre ogni singola operazione elementare che viene eseguita sulle matrici di base in una corrispondente operazione booleana. Le operazioni in questione sono somme e moltiplicazioni, e

viene abbastanza naturale tradurre le somme come OR logici e i prodotti come AND logici. Di conseguenza si avrà che il prodotto booleano sarà definito operativamente come nell'Algoritmo 4.

Algorithm *Prodotto booleano*()
1. Date due matrici booleane comformabili (quindi tali che, se la prima è $m \times n$ e la seconda sia $n \times p$:
2. Per ogni riga della prima matrice, si analizza ogni colonna della seconda matrice. Ogni elemento $M_{i,j}$della matrice prodotto booleano è dato dalla formula:
3. $(A_{i,1} AND\, B_{1,j})\, OR\, (A_{i,2} AND\, B_{2,j})\, OR \ldots\, OR\, (A_{i,n} AND\, B_{n,j})$
4. che è molto simile alla classica formula per il prodotto righe per colonne di matrici, con la sola differenza che la somma diventa un OR logico e il prodotto un AND logico.

A.4 Definizione di determinante e della matrice inversa

La matrice inversa di una matrice booleana di sparsità è, analogamente ala definizione precedente, la matrice booleana di sparsità della matrice inversa, a meno dei casi in cui si verificano cancellazioni dovute al calcolo del determinante:

$$S_{inv} = invBoolean\,(S_p(M))) \tag{A.3}$$

Analogamente a come fatto nella sezione precedente, si ridefiniscono anche il determinante e l'inversa di una matrice, adottando il medesimo approccio.
Di conseguenza il determinante di una matrice booleana sarà:

$$\begin{cases} detBoolean\,(Sp(M)) = TRUE & se\ det(M) \neq 0 \\ detBoolean\,(Sp(M)) = TRUE & se\ det(M) = 0\ e\ cancellazioni \\ detBoolean\,(Sp(M)) = FALSE & altrimenti \end{cases} \tag{A.4}$$

E di conseguenza il determinante booleano è calcolato come il determinante classico, ricorsivamente, utilizzando lo sviluppo di Laplace[1] L'inversa di una matrice booleana è calcolata, analogamente, come matrice composta da tutti i determinanti delle matrici booleane ridotte corrispondenti agli elementi della matrice. Il numero di determinanti da calcolare è quindi dipendente solo dalla dimensione della matrice quadrata da invertire, ed è pari a N^2, dove N è la dimensione della matrice stessa.[2] Questi metodi, ben più pesanti dal punto di vista computazionale del semplice calcolo del determinante di matrice numerica e del conseguente

[1] Si noti che l'algoritmo di Gauss, che si basa sulla riduzione di Gauss, non è applicabile in quanto non è vero, per matrici booleane, che $inv\,(M) * M = I$

[2] Per lo stesso motivo precedente, non è possibile, ancora una volta, utilizzare l'algoritmo di Gauss.

calcolo dell'inversa tramite il metodo di Gauss, permettono però di passare direttamente, tramite lavoro sulle matrici booleane, alla matrice (o al determinante) che indica la presenza o meno di zeri nelle matrici risultanti. L'utilità di questo approccio è data dal fatto che il calcolo della matrici di sparsità delle matrici risultanti dal calcolo dei termini comuni per accelerazione (si veda la sezione 6.4) deve essere effettuato solo la prima volta, al caricamento del modello del sistema da controllare; passate le matrici dei termini comuni (che sono matrici di funzioni) e le relative matrici di sparsità, non resta che valutare le matrici nei punti indicati.

Appendice B

Caratteristiche del toolbox

Si presentano di seguito le caratteristiche del toolbox costruito con Matlab per l'esecuzione di un controllo MPC non lineare. Il toolbox è in grado di sviluppare simulazione, identificazione e controllo ottimo di un sistema non lineare costruito in forma LFT.

B.1 Costruzione della forma LFT

La costruzione della forma LFT può avvenire manualmente o tramite una serie di passaggi attraverso un Toolbox precedente, descritto in [8]. Si richiedono alcuni passaggi di integrazione per ottenere la forma LFT in Matlab. Questa serie di passaggi può essere eseguita facilmente, una volta noto il modello del sistema. Sempre in questa fase si calcolano le componenti di sparsità del modello (come visto nell'Appendice A), le quali infatti non dipendono dalla situazione del modello, ma soltanto dalla sua struttura non lineare e dall'incidenza degli ingressi sugli stati del sistema.

B.2 Solutore della forma LFT

La funzione più importante del Toolbox Ã¨ sicuramente il solutore LFT, il cui obiettivo è integrare il sistema non lineare dati stato iniziale, ingressi e opzioni. Il solutore LFT può essere chiamato nei seguenti modi:

1. ```
   [output, internalSolution,CommonTerms] =
       lftSolver(lftfun,Input,InitialConditions,lftSolverOptions);
   ```

2. ```
   [output, internalSolution,CommonTerms] =
       lftSolver(lftfun,TimeSpan,InitialConditions,lftSolverOptions);
   ```

Il primo metodo viene utilizzato per eseguire la simulazione LFT completa e la simulazione LFT di derivazione per l'identificazione dei parametri, come mostrato nel Capitolo 3, e richiede in ingresso:

- Il modello LFT del sistema (*lftfun*)
- La struttura di ingresso (*Input*)
- La struttura di condizioni iniziali (*InitialConditions*)
- Le (eventuali) opzioni (*lftSolverOptions*)

Il secondo metodo invece si utilizza per la sola simulazione di derivazione per il controllo, e di conseguenza non richiede in ingresso gli input del sistema, bensì il solo tempo di integrazione *Timespan* su cui eseguire le derivazioni.
Entrambi i metodi danno in uscita tre componenti:

output contiene al suo interno tutte le uscite richieste, valutate negli istanti di tempo previsti

internalSolution contiene i valori delle soluzioni del sistema negli istanti di tempo scelti dal solutore

CommonTerms contiene i termini comuni che vengono riutilizzati nelle derivazioni

B.2.1 Caratteristiche dell'input

Di seguito si mostrano le caratteristiche che possono essere sfruttate nell'inserimento degli ingressi al solutore.

Input	**Descrizione**	
lftfun	Struttura LFT. Contiene:	
	LTI	contiene la parte Lineare Tempo Invariante della struttura LFT
	DeltaSym	contiene nomi, indici e estremi dei parametri da identificare nel modello
	DeltaVal	contiene la matrice diagonale che indica le incidenze dei parametri sul sistema
	Theta	contiene un *handle* alla funzione non lineare in retroazione
	dThetadOmega	contiene un *handle* al gradiente di Theta rispetto alle variabili non lineari

	Sparsity	contiene le informazioni di sparsità del sistema LFT
	ISO	struttura di celle contenente i nomi di tutte le variabili del modello
Input	Struttura di Input. Contiene:	
	Type	segnala il tipo di Input. Può essere *discontinuous*, *interpolated* o *continuous*. Nel primo caso i valori di input presenti nel campo *Input.Samples* sono presi come a gradini nel tempo. Nel secondo caso si esegue una interpolazione lineare e nel terzo i valori in *Input.Coefficients* sono interpolati nel tempo *Input.Time* per ottenere i valore di *Input.Samples*.
	Time	indica il vettore temporale su cui agisce l'ingresso inserito. È anche il tempo adottato per determinare inizio e fine della simulazione.
	Samples	contiene i valori, per ogni istante di tempo, di tutti gli ingressi.
	Coefficients	contiene i coefficienti che, nel caso di *Input.Type = continuous*, vengono utilizzati come coefficienti da interpolare su tutto l'orizzonte temporale.
TimeSpan	Vettore degli istanti di tempo in cui eseguire la derivazione rispetto agli ingressi.Utilizzato solo per l'esecuzione del filtro di sensitività rispetto agli ingressi.	
InitialConditions	Struttura contenente le condizioni iniziali del sistema. È composta da:	
	StateInitialConditions	contiene il vettore di condizioni iniziali delle variabili di stato del sistema LFT.

	SensInitialConditions	contiene la matrice di valori delle condizioni iniziali delle derivate delle variabili di stato rispetto agli ingressi, disposti per righe.
lftSolverOptions	Struttura contenente le opzioni del solutore. Si compone di:	
	RelTol	è la tolleranza relativa sulla precisione della soluzione, default $1e-3$.
	AbsTol	è la tolleranza assoluta sulla precisione della soluzione, default $1e-4$.
	MaxOrder	indica il massimo ordine di accuratezza per *ode15s*, default 5.
	BDF	indica l'uso del metodo BDF, default *off*.
	InitialStep	indica il modulo del passo iniziale per il solutore, default $1e-3$.
	MinStep	indica la minima lunghezza del passo per il solutore. Di default è lasciata vuota, condizione in cui viene ignorata.
	MaxStep	indica la massima lunghezza del passo per il solutore. Di default è lasciata vuota, condizione in cui viene ignorata.
	NumberOfSteps	indica il numero di passi che si vuole che il solutore esegua a partire dalle condizioni iniziali. Se vuoto (da default) viene ignorato, altrimenti il solutore si ferma dopo il numero di passi indicato, ignorando i valori successivi in *Input.Time*.
	ShowIntTime	*false* di default, indica se mostrare a video l'istante temporale in cui si esegue ogni passi di simulazione.
	SolutionInterpMethod	permette all'utente di scegliere il tipo di interpolazione che deve essere eseguita sui valori dell'ingresso. Di default è *linear*.

OversamplingMethod	permette all'utente di scegliere il tipo di interpolazione che deve essere svolta per ottenere l'uscita *output*. Di default è *linear*.
SolutionTimeSpan	permette di scegliere il vettore di istanti di tempo in cui valutare la soluzione *output*.
Sensitivity	sceglie come utilizzare il solutore. Ponendo $Sensitivity = 0$ si utilizza il solutore come simulatore di base del filtro LFT al I stadio (si veda in proposito la Figura 3.1). Ponendo $Sensitivity = 'uX'$ con X uguale all'indice della variabile di ingresso, si esegue il filtro LFT di secondo stadio rispetto ad u (si veda in proposito la Figura 6.2. Con $Sensitivity = 'pX'$ e con X uguale all'indice del parametro desiderato, si esegue il filtro LFT di secondo stadio rispetto al parametro scelto (come in Figura 3.1).
SensAlgorithm	imposta l'algoritmo da utilizzare per i calcoli concernenti il filtro di sensitività. Di default è impostato su *ode15s*.
CommonTerms	permette di passare al solutore di sensitività i termini comuni ottenuti dal solutore di primo stadio. Tramite i CommonTerms si permette al solutore di sensitività di effettuare soltanto valutazioni di funzioni già pronte. Ciò consente allo stadio LFT di sensitività di essere circa 100 volte più rapido del primo.

B.2.2 Caratteristiche dell'output

Si mostrano ora le caratteristiche dell'uscita del solutore. L'uscita è composta da un gruppo di matrici che contengono al loro interno interessanti costruzioni. È molto utile, allo scopo di usare il Toolbox, comprendere cosa dà effettivamente in uscita.

Output	Descrizione
output	Matrice in uscita, contenente $1 + n_y$ colonne, dove n_y indica il numero di uscite del sistema LFT. La prima colonna della matrice è il vettore tempo in cui è stata interpolata la soluzione del sistema, mentre le altre contengono i valori delle variabili di uscita in ogni istante di tempo. Il vettore tempo è lo stesso dato in ingresso attraverso *lftSolverOptions.SolutionTimeSpan*, se esistente, altrimenti è quello scelto internamente dal solutore.
internalSolution	Matrice che cambia il suo contenuto in base al valore di *lftSolverOptions.Sensitivity*. Nel filtro LFT di I stadio, infatti, *internalSolution* è una matrice di $1 + n_u + n_x + n_z + n_\omega + n_y$ colonne. La prima colonna contiene tutti gli istanti di tempo scelti dal solutore. Le successive contengono in ordine le variabili di ingresso, le variabili di stato, le variabili da identificare e le variabili con componenti non lineari, tutte valutate negli istanti di tempo alla prima colonna. Nel filtro di sensitività per identificazione, invece, la matrice è composta da $1 + n_x + n_y$ colonne e contiene in ordine gli istanti di tempo scelti dal solutore, i valori delle derivate dello stato e delle uscite in quegli stessi istanti di tempo rispetto al parametro scelto. Infine, nel filtro di sensitività per controllo la matrice è composta da $1 + n_x + n_x + n_y$ colonne e contiene, in ordine: la solita colonna del vettore tempo, la derivata dello stato rispetto all'ingresso scelto, la derivata di $\dot{x}$ rispetto all'ingresso scelto e la derivata dell'uscita rispetto all'ingresso scelto, tutte valutate negli istanti di tempo scelti dal solutore e posti nella prima colonna.

CommonTerms	Struttura che contiene al suo interno le matrici componenti il sistema LFT, sia fisse (matrici che non sono tempo-varianti, cioè relative a componenti del sistema che sono intrinsecamente lineari), sia variabili (cioè relative a componenti non lineari che sono considerate lineari tempo varianti). Le matrici tempo-varianti sono matrici a tre dimensioni, cioè tensori. Ogni piano del tensore rappresenta la matrice tempo-variante valutata in un istante di tempo scelto dal solutore. Il tensore viene utilizzato per interpolazione tra i valori relativi a piani adiacenti.

B.3 Ottimizzatore rispetto agli ingressi

L'altra funzione sviluppata per ottenere il controllo ottimo è la funzione *lftOptInput*, che è in grado di eseguire una ottimizzazione per controllo ottimo. La funzione si chiama con la seguente sintassi:

```
[OptInput, CostFunction, gradient, ConditionNumber, Opthistory]=
    lftOptInput(lftfun,InputKnown,InitialConditions,
        InputOptStart,YToFollow,UObjective,lftSolverOptions,lftOptimOptions);
```

La funzione richiede in entrata i seguenti componenti:

lftfun Analogamente al solutore, è il sistema LFT

InputKnown Contiene una struttura di tipo *Input*, in cui si inseriscono gli ingressi noti o comunque non da ottimizzare (ad esempio disturbi o ingressi esogeni non controllabili).

InitialConditions Esattamente come il solutore, si impongono le condizioni iniziali dello stato del sistema all'inizio della risoluzione del problema di controllo ottimo.

InputOptStart Si pone un iniziale comportamento degli ingressi da ottimizzare all'interno dell'orizzonte.

YToFollow Contiene una struttura, sempre di tipo *Input*, che dà il riferimento da seguire.

UObjective Contiene una struttura analoga a *InputOptStart*, che però ha il significato di ingressi obiettivo. L'azione di controllo deve mantenersi quanto più possibile ai valori richiesti in questo ingresso.

lftSolverOptions Si veda B.2.1: sono le opzioni relative al solutore interno al controllore.

lftOptimOptions Sono le opzioni dell'ottimizzatore. Si tratta di una struttura composta come da Tabella B.3.

Componente di *lftOptimOptions*	**Descrizione**
TolFun	Funzione di tolleranza relativa per l'ottimizzazione. Di default è posta a $1e-3$.
MaxStep	Massimo valore di lunghezza del singolo passo dell'ottimizzatore. Di default è lasciato vuoto così che venga ignorato nell'ottimizzazione.
NumberOfSteps	Massimo numero di passi di ottimizzazione. Di default è posto a 2, perché per ottenere una riduzione della cifra di merito sufficiente all'esecuzione del controllo ottimo di solito bastano 2 passi di ottimizzazione.
PolynomialDegree	Grado del polinomio interpolante per l'ingresso. Il numero di coefficienti dell'ingresso continuo deve essere pari a $1 + PolynomialDegree$.
RelTolX	Tolleranza relativa di esattezza del rapporto tra la norma del passo e la norma locale di x. Indica la condizione di termine dell'iterazione di ottimizzazione. Di default è posta a $1e-3$.
MinNormGrad	Minimo valore della norma del Gradiente per determinarne la positività e di conseguenza l'impatto sulla ricerca della soluzione. Di default è posta a $1e-3$.
EpsilonLambda	Massimo valore di tolleranza per lo zero. Di default è $1e-6$.
MaxIter	Massimo numero di iterazioni di Newton per il calcolo del passo. Di default è posto a 20.
Display	Indica all'ottimizzatore se mostrare le iterazioni di Newton per la ricerca del passo ottimo oppure no. Di default è falso.

StartOptimSample	Indice del primo punto nel tempo da cui iniziare l'ottimizzazione. Può essere utilizzato per considerare tipi di variabili di controllo che possono essere controllati solo dopo un determinato ritardo di tempo. Di default è 1 (nessun ritardo).

Si considerino ora le uscite della funzione *lftOptInput*. È utile analizzarle e comprendere che cosa viene estratto dalla funzione di ottimizzazione degli ingressi per controllo ottimo. Le uscite della funzione sono:

OptInput Uscita principale della funzione, è di tipo *Input* con *Input.Type = 'continuous'*. I suoi coefficienti sono i coefficienti ottimi, di grado *lftOptimOptions.PolynomialDegree*, applicati negli istanti di tempo calcolati dal solutore interno e dati in uscita in *OptInput.Time*.

CostFunction Indica il valore della funzione obiettivo J della cifra di merito MPC.

gradient Indica il gradiente della funzione obiettivo nell'ottimo trovato.

ConditionNumber È il numero di condizionamento dell'hessiano della cifra di merito H. Può essere utile per determinare se il problema è mal condizionato.

Opthistory Contiene la storia dei valori dei coefficienti di input, così come sono evoluti durante la minimizzazione eseguita per trovare l'input ottimo dato in uscita.

B.3.1 Funzionamento dell'ottimizzatore

Analizzati a fondo gli ingressi e le uscite della funzione di controllo ottimo, si esegue una rapida presentazione del funzionamento del codice Matlab che genera il controllo ottimo nel toolbox. Perchè il controllo funzioni correttamente, è fondamentale passare alla funzione di ottimizzazione uno stato iniziale che sia consistente, sia rispetto allo stato, sia rispetto alla sensitività. Per ottenerlo la maniera più semplice è dare i valori di uscita della simulazione LFT dall'inizio fino al punto in cui si comincia ad effettuare controllo ottimo.

Ad ogni iterazione del controllo, si calcola sull'ottimo corrente il valore della funzione di costo J, del gradiente G e dell'Hessiano H. Per effettuare il calcolo si simula l'evoluzione del sistema all'interno dell'orizzonte di predizione (cioè usando il vettore di tempo passato inizialmente), e si simulano le sensitività rispetto ad

ogni ingresso da ottimizzare. Dai valori ottenuti (ed utilizzando i termini comuni) si costruisce la matrice derivata di Y in U ad ogni istante (così come mostrata nella (5.66)), e si costruiscono con essa gradiente, Hessiano e cifra di merito seguendo le formule classiche (esposte in 5.1).
Ottenuti i valori in questione, si applicano i metodi di Gauss e di steepest-descent, come descritti in 4.1, per trovare la direzione di avanzamento ed il passo ideali per calcolare il minimo. Il punto così trovato aggiorna la soluzione per l'input ottimo, e l'algoritmo viene rieseguito fino a convergenza o fino a superamento del massimo numero di passi.

Ringraziamenti

So già che molti leggeranno direttamente questa pagina, quindi per prima cosa ringrazio chi si è dato la pena di leggere le pagine precedenti a questa; spero di non avervi annoiato troppo (e in ogni caso non s'è fatto apposta). Ringrazio ancora di più chi ha perfino tentato di capire qualcosa nello sproloquio fin qui esposto.
Ringrazio di cuore i miei genitori che mi hanno sempre sostenuto in tutto ciò che ho fatto o desiderato fare, sia che conducesse ad un successo che altrimenti. Grazie, grazie, grazie!!
Un grande grazie al prof. Ferretti, per il suo aiuto e per la grandissima professionalità con cui mi ha sempre indicato la via più semplice per migliorare questo elaborato.
Un grandissimo ringraziamento va all'ing. Della Bona, senza il quale non sono sicuro che sarei riuscito a fare quasi nulla di quanto presentato nelle pagine precedenti. Grazie per avermi insegnato tantissimo durante questi mesi passati a lavorare gomito a gomito, a veder crescere e svilupparsi un lavoro comune e a stupirsi insieme di come funzioni bene! Grazie per avermi dimostrato ed insegnato che è possibile lavorare in modo diverso, e che fare innovazione vera significa rimboccarsi le maniche e tentare di andare dove nessuno è mai andato prima, con la sola guida della propria ragione e degli studi fatti.
Ringrazio il prof. Bascetta che mi ha dato l'opportunità di iniziare il lavoro in una tesi che ho sempre sognato di fare.
Grazie a tutti i miei professori, sia del Politecnico che delle scuole precedenti, per la passione che avete messo nel trasferirmi le vostre conoscenze. In particolare un ringraziamento al prof. Nicolosi alle medie per avermi insegnato le basi della programmazione - agli albori dell'informatica! E un grazie alla prof.ssa Brena al liceo per avermi trasmesso la passione per la matematica, che non mi ha più lasciato.
Grazie a Corina, che non mi ha mai lasciato mollare e si è sorbita la cronistoria di tutte le mie perplessità riguardo la riuscita di questo elaborato, e riguardo il futuro in genere. Grazie per avermi ascoltato e amato anche quando sembrava dicessi cose al di fuori di questo mondo.
Grazie a mio fratello che mi ha sopportato nei miei sbalzi d'umore in questi mesi; guardandoti studiare per Meccanica mi sono sentito molto orgoglioso di te e sono

contento di vedere che studi qualcosa che ti piace davvero.
Grazie ai miei parenti, ai nonni e agli zii, nei quali ho sempre trovato orgoglio, amore e stima. Un grazie speciale alla nonna Maria che mi ha allevato.
Un grazie a tutti gli amici della Torrescalla, dove ho costruito tutti i miei anni accademici qui al Politecnico. Grazie di cuore per tutto ciò che mi avete insegnato e per ciò che avete sempre rappresentato per me. Grazie a Paso, mi sei sempre stato d'esempio. Grazie a Gino, che mi ha trasmesso il metodo, la passione per lo studio e per la scoperta di cose nuove, sempre sorridendo. Grazie a Lodo, compagno inseparabile di mille avventure, cento viaggi e centomila emozioni in tutti questi anni. Grazie al Doc, espressione vera della poliedricità (un medico con la passione per l'elettronica non l'ho più trovato). Grazie a Mattia per la tua preziosa amicizia. Grazie a Ricky per l'esuberanza e la spinta a divertirsi e giocare insieme ogni volta possibile. Grazie a Snapo per avermi mostrato come si può essere uomini di grande successo senza dimenticare di divertirsi. Grazie a Luca e a Beppe per essere sempre stati grandi amici prima che Direzione. Grazie a Willy, per aver sempre pronta una battuta o un sorriso. Grazie a tutti quelli con cui ho giocato al TIAS o a calcio in genere, per il piacere di momenti di relax (e di rabbia) insieme.
Grazie ai ragazzi di AIESEC, ho vissuto con voi una splendida esperienza; vivendo insieme un sogno in cui crediamo davvero. Grazie per ogni momento insieme e per ogni risultato ottenuto, e, molto più importante, per ogni fallimento affrontato. In particolare grazie a Elena per avermi aperto questa porta e grazie a Chiara, Berta, Ferrero, Ele, Marilù, Mario, Floriana, Bruno, James, Massi e Luca per un anno stupendo insieme al Nazionale. E un grazie ancora più particolare a Vito anche perchè il lavoro che abbiamo fatto insieme ha avuto davvero un impatto, e ne sono orgoglioso.
Grazie a tutti gli amici per ogni prezioso consiglio.
Infine grazie a Massimo, a Leonardo e a tutti i colleghi di Johnson per l'accoglienza, la splendida opportunità e ciò che mi avete insegnato. Grazie anche per avermi sempre permesso di lavorare a questo elaborato.
Sicuramente avrò dimenticato qualcuno, e mi scuso fin da subito: purtroppo lo spazio qui è limitato mentre il cuore no. Vi ringrazio tutti, soprattutto per essere arrivati fin qui a leggere. Un abbraccio!

Bibliografia

[1]

[2] Della Bona A. Solutore efficiente per la simulazione di modelli lft in cascata. 2014.

[3] Carson J. M. Açikmeşe, A. B. A nonlinear model predictive control algorithm with proven robustness and resolvability. *NASA*, May 2014.

[4] Findeisen R. Nagy Z.K. Allgower, F. Nonlinear model predictive control: From theory to application. *J. Chin. Inst. Chem. Engrs*, 35(3), 2004.

[5] Petzold L. R. Ascher, U. M. *Computer methods for ordinary differential equations and differential algebraic equations*. Society for Industrial & Applied Matemathics, August 1998.

[6] Modelica Association. *Modelica - A Unified Object-Oriented Language for Physical Systems Modelling - Tutorial*. 1.4 edition, 2000.

[7] Craig I. K. Bauer, M. Economic assessment of advanced process control: A survey and framework. *Journal of Process Control*, 18(1):2, 18, January 2008.

[8] Donida F. Lovera M. Casella, F. Integrated modelling and parameter estimation: an lfr-modelica approach. In IEEE, editor, *Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, volume 11 of *1*, pages 8357, 8362, Shanghai, P.R. China, December 2009. IEEE.

[9] Shaw L. Chen, C. C. On receding horizon feedback control. *Automatica*, 18:349, 1982.

[10] Coletsos I. Kokkinis B. Cryssoverghi, I. Discretization methods for optimal control problems with state constraints. *Journal of Computational and Applied Mathematics*, 191(1):1,31, 2006.

[11] J. Rabinowitz P. Davis, P. *Methods of numerical integration*. Academic Press, 1975.

[12] M. Casella F. Romani C. Donida, F. Lovera. Integrated modelling and parameter estimation: an lfr-modelica approach. In IEEE, editor, *IEEE Conference on Decision and Control*, volume 1 of *1*, page 8357, 2009.

[13] Sokoler L. E. Jorgensen J. B. Edlund, K. A primal-dual interior-point linear programming algorithm for mpc. In IEEE, editor, *Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, volume 1 of *1*, pages 351, 356, Shanghai, P.R. China, December 2009. IEEE.

[14] Kazemi M. A. Razzaghi M. Elnagar, G. The pseudospectral legendre method for discretizing optimal control problems. *IEEE Transactions on Automatic Control*, 40(10):1793, 1796, October 1995.

[15] Hu H. Gu, D. A stabilizing receding horizon regulator for nonholonomic mobile robots. *IEEE*, 1552(3098), 2005.

[16] Ekholm K. Petter S. Johansson R. Tunestal P. Karlssson, M. Multiple-input multiple-output model predictive control of a diesel engine.

[17] Abel D. Katriniok, A. Ltv-mpc approach for lateral vehicle guidance by front steering at the limits of vehicle dynamics. In IEEE, editor, *50th IEEE Conference on Decision and Control and European Control Conference*, volume 1 of *1*, pages 6828,6833, Orlando, FL, USA,, December 2011. IEEE.

[18] Shuang L. Cannon M. Kouvaritakis, B. A line search improvement of efficient mpc. volume 1 of *4*, Marriott Waterfront, Baltimore, MD, USA, July 2010. American Control Conference.

[19] J. Raimondo D.M. Allgower F. Magni, L. *Nonlinear Model Predictive Control*. Springer, 2009.

[20] C. M. Shetty Mokhtar S. Bazaraa, Hanif D. Sherali. *Nonlinear Programming: Theory and Algorithms*. Wiley Ed., 2006.

[21] Zainabolhoda H. Moshiri B. Mousavi, M. A. Ltv-mpc based path planning of an autonomous vehicle via convex optimization. *IEEE*, 3(13), 2013.

[22] Morari M. Oliveira, S.L. Robust model predictive control for nonlinear systems. In Pasadera, editor, *Proceedings of the 33rd Conference on Decision end Control*, 1994.

[23] Barbarisi O. Scala S. Glielmo L. Palmieri, G. A preliminary study to integrate ltv-mpc lateral vehicle dynamics control with a slip control. In IEEE, editor, *Joint 48th IEEE Conference on Decision and Control and 28th Chinese Control Conference*, volume 12 of *6*, pages 4625,4630, Shanghai, P.R. China, December 2009. IEEE.

[24] Zhang L. W. Ren, Y. H. The dual algorithm based on a class of nonlinear lagrangians for nonlinear programming. In IEEE, editor, *Proceedings of the 6th World Congress on Intelligent Control and Automation*, volume 1 of *1*, pages 934, 938, Dalian, China, June 2006. IEEE.

[25] Westerlund T. Still, C. A sequential cutting plane algorithm for solving convex nlp problems. *European Journal of Operational Research*, 1(173):444, 464, 2006.

[26] Westerlund T. Still, C. A linear programming-based optimization algorithm for solving nonlinear programming problems. *European Journal of Operational Research*, 1(200):658, 670, 2010.

[27] T. Tashiro. Vehicle steering control with mpc for target trajectory tracking of autonomous reverse parking. In IEEE, editor, *2013 IEEE International Conference on Control Applications (CCA)*, volume 1 of *1*, pages 247, 251, Hyderabad, India, August 2013. IEEE.

[28] Murray R. M. Walsh G. Teel, A. R. Nonholonomic control systems: From steering to stabilization with sinusoids. In IEEE, editor, *Proceedings of the 31st Conference on Decision and Control*, volume 11 of *10*, pages 1603, 1609, Tucson, Arizona, US, December 1992. IEEE.

[29] Apel T. Troltzsch F. Vexler B. Thienel, K. C. *Discretization strategies for optimal control problems with parabolic partial differential equations*. PhD thesis, Universitat der Bundeswehr Munchen, April 2013.

[30] Zhang Y. Villalobos, M. C. A trust-region interior-point method for nonlinear programming. In ACM, editor, *TAPIA*, volume 1 of *1*, pages 7, 9, Albuquerque, New Mexico, USA, October 2005. ACM.

[31] A. Zheng. A computationally efficient nonlinear mpc algorithm. In AACC, editor, *Proceedings of the American Control Conference*, volume 1 of *1*, pages 1623, 1627, Albuquerque, New Mexico, June 1997. AACC.

Elenco delle figure