Università Politecnica delle Marche
Scuola di Dottorato di Ricerca in Scienze dell'Ingegneria
Curriculum in Computer, Management and Automation Engineering

---

Problemi di controllo ottimo per sistemi non lineari con struttura State-Dependent Coefficient, implicazioni ed aspetti computazionali

Tesi di Dottorato di:
Giuseppe Antonio Scala

Tutor:
Prof. Andrea Bonci

Coordinatore del Curriculum:
Prof. Franco Chiaraluce

XXXIV ciclo - nuova serie

Università Politecnica delle Marche
Scuola di Dottorato di Ricerca in Scienze dell'Ingegneria
Curriculum in Computer, Management and Automation Engineering

---

Problemi di controllo ottimo per sistemi non lineari con struttura State-Dependent Coefficient, implicazioni ed aspetti computazionali

Tesi di Dottorato di:
Giuseppe Antonio Scala

Tutor:
Prof. Andrea Bonci

Coordinatore del Curriculum:
Prof. Franco Chiaraluce

XXXIV ciclo - nuova serie

Università Politecnica delle Marche
Scuola di Dottorato di Ricerca in Scienze dell'Ingegneria
Facoltà di Ingegneria
Via Brecce Bianche – 60131 Ancona (AN), Italy

Abstract

The category of dynamical systems most established and studied for the longest time is that of linear systems; however, most of the real systems are non-linear, the State Dependent Coefficient (SDC) approach proposes a method to represent them with structures similar to those of the linear world retaining their non-linear nature. Linear time invariant systems have been extensively studied, it is possible to clearly identify characteristics such as reachability, observability and stability. The control techniques can achieve the overall optimum in a mathematically certain way if the system meets the requirements. The most frequent strategy is to linearize the system around the working point, obtaining satisfactory performances valid in the local neighborhood, this approach in many cases is too restrictive. For example, in the aerospace field, where the devices work in every part of the atmosphere under significantly different conditions, or more simply in applications where the working point moves away from that of linearization. Unlike non-linear control techniques, such as sliding mode or feedback linearization, the State Dependent Riccati Equation (SDRE) control offers a hybrid alternative. Using the theory of linear optimal control, it is possible to extend the effectiveness of the Linear Quadratic Regulator (LQR) control to non-linear systems through the use of SDC matrices and SDRE control. This allows to control the system in an area of the state space that is larger than the neighborhood working point where the system has been linearized. The SDRE control technique has been applied to various devices, such as missiles, aircraft, UAVs, autonomous systems, biomedical systems and robots. The robustness, flexibility of this technique and today's ability to implement control in real time have attracted the attention of the community. Despite the growing documentation on the SDRE technique, there is still a significant lack of theoretical justifications on the choice of SDC matrices and their effectiveness in representing the system.
This thesis illustrates how to represent a system in state space with terms dependent on the state itself. The most common methods for constructing SDC matrices have been collected, improving some aspects and suggesting alternatives. The SDRE control with SDC structure is used and studied in efficiency, efficacy and ductility with methods known in the literature and new modalities. An innovative solution is proposed for this control in the choice of the cost function, which can also be extended to the well-known LQR. In conclusion, all

the techniques on different case studies commonly used in literature and some real application cases are illustrated.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1.

Introduzione

La categoria di sistemi dinamici più assodata e studiata da maggior tempo è quella dei sistemi lineari; tuttavia, la maggior parte dei sistemi reali è non lineare, l'approccio State Dependent Coefficient (SDC) propone un metodo per rappresentarli con strutture simili a quelle del mondo lineare ma conservando la loro natura non lineare. I sistemi lineari tempo invarianti sono stati ampiamente studiati, è possibile identificare chiaramente le caratteristiche come raggiungibilità, osservabilità e stabilità. Le tecniche di controllo possono raggiungere l'ottimo globale in modo matematicamente certo se il sistema soddisfa i requisiti richiesti. La strategia più frequente è quella di linearizzare il sistema attorno al punto di lavoro, ottenendo delle prestazioni soddisfacenti valide nell'intorno locale, questo approccio in molti casi è troppo restrittivo. Ad esempio, nel campo aerospaziale, dove i dispositivi lavorano in ogni parte dell'atmosfera in condizioni significativamente diverse, oppure più semplicemente in applicazioni dove il punto di lavoro si allontana da quello di linearizzazione. A differenza di tecniche di controllo non lineari, come sliding mode o feedback linearization, il controllo State Dependent Riccati Equation (SDRE) propone un'alternativa ibrida. Utilizzando la teoria del controllo ottimo lineare, è possibile estendere l'efficacia del controllo Linear Quadratic Regulator (LQR) a sistemi non lineari passando attraverso l'uso di matrici SDC ed il controllo SDRE. Questo permette di controllare il sistema in un'area dello spazio di stato più estesa dell'intorno del punto di lavoro dove si sarebbe linearizzato il sistema. La tecnica di controllo SDRE è stata applicata su diversi dispositivi, come missili, aereomobili, UAV, sistemi autonomi, sistemi biomedicali e robot. La robustezza, flessibilità di questa tecnica e la possibilità odierna di implementare il controllo in tempo reale hanno attirato l'attenzione della comunità. Nonostante la crescente documentazione sulla tecnica SDRE è presente tutt'oggi una significativa mancanza di giustificazioni teoriche sulla scelta delle matrici SDC e la loro efficacia nel rappresentare il sistema.
In questa tesi viene illustrato come rappresentare un sistema in spazio di stato con termini dipendenti dallo stato stesso. Sono stati raccolti i metodi più comuni per costruire le matrici SDC migliorandone alcuni aspetti e proponendo

alternative. Il controllo SDRE con struttura SDC viene utilizzato e studiato in efficienza, efficacia e duttilità con metodi noti in letteratura e nuove modalità. Viene proposta una soluzione alternativa per tale controllo nella scelta della funzione di costo, estendibile anche al noto LQR. In conclusione, sono illustrate tutte le tecniche su diversi casi di studio utilizzati comunemente in letteratura ed alcuni casi di applicazione reale.

Capitolo 2.

Stato dell'Arte

La struttura SDC fu proposta da Pearson nel 1962 [1], un sistema non lineare può essere rappresentato in spazio di stato definendo i coefficienti delle matrici parametrici rispetto allo stato stesso, State Dependent Coefficients (SDC). Per controllare tali sistemi non lineari mediante la classica funzione di costo quadratico, occorre risolvere l'equazione di Riccati a sua volta dipendente dallo stato e dal tempo in ogni istante, tale approccio è noto ad oggi in letteratura come State Dependent Riccati Equation (SDRE). Questo metodo porta ad una soluzione sub-ottima, che confrontata con la traiettoria ottimale risulta avere una differenza accettabile. Quando il controllo ottimo non può essere stimato agevolmente a causa di una natura non lineare del sistema, il controllo SDRE dimostra efficacia, efficienza e duttilità. Il problema principale del metodo era risolvere l'equazione di Riccati ad ogni istante di tempo, che nel '60 era fattibile solo offline. Negli ultimi 20 anni, si cerca di reintrodurre questo metodo grazie all'avanzamento tecnologico che ci permette di implementare il metodo in tempo reale.
Da 1962 ad oggi, sono stati fatti diversi studi applicativi e teorici sul metodo in oggetto, una review del 2010 di Cimen [2] raccoglie i punti più importanti, ampliata nel 2012 [3]. La tecnica SDRE è stata ampliata da Wernli e Cook (1975) [4], studiata indipendentemente da Cloutier, D'Souza, and Mracek (1996) [5] e da Hammett (1997) [6], fino alla formulazione di definizioni e teoremi da parte di Cloutier, Stansbery and Sznaier nel 1999 [7]. In tali lavori viene mostrato il controllo non lineare a tempo infinito SDRE che nel caso multivariabile è localmente asintoticamente stabile e localmente asintoticamente ottimo, mentre nel caso scalare è ottimo. Inoltre, viene mostrato che l'algoritmo soddisfa asintoticamente le condizioni necessarie di Pontryagin nel caso generale multivariabile. Altri autori hanno analizzato e messo in discussione tali risultati, Erdem nel 2001 [8] ha mostrato le difficoltà della ricerca dell'ottimo globale, che dipende dalla scelta di come la matrice della dinamica è costruita. Sulla base di questo contributo teorico sono state fatte numerose applicazioni, come: un sistemi di guida missili [9] [10]; sistemi di controllo satelliti [11] [12] [13]; robotica [14]; controllo pancreas artificiali [15], ductedfan [16].

In tempi più recenti, ci sono state ulteriori applicazioni del metodo per verificarne fattibilità ed efficacia. Nel [17] un robot manipolatore a 3 DOF è stato controllato con la tecnica SDRE dove la dinamica è parametrizzata rispetto posizione e velocità dei giunti. Alcuni lavori hanno confrontato direttamente il metodo con tecniche non lineari. Nel [18] Razzaghi, il controllo SDC è confrontato con lo sliding mode, evidenziando che per sistemi sufficientemente semplici o di piccole dimensioni come SISO un approccio con tecniche esatte non lineari è preferibile. Piccoli passi sono stati compiuti anche in campo teorico studiando la regione di attrazione come illustrato nell'articolo [19]. Il problema presenta una notevole complessità, con una soluzione molto corposa e difficile da gestire in applicazione. Recentemente si è applicato l'approccio SDRE anche nella stima dello stato di sistemi complessi come le forze di contatto fra strada e pneumatici di un veicolo e lo stato dell'auto stessa, comparandola con la tecnica del filtro di Kalman esteso in [20]. La tecnica SDRE è ormai descritta in alcuni libri di testo già dal 2015, come "Nonlinear Systems and Control" [21]. Negli ultimi anni, alcuni autori hanno provato a tracciare un metodo per selezionare la forma SDC che meglio controlla il sistema, cercando di impostare condizioni di raggiungibilità ed osservabilità tuttavia molto complesse ed articolare, come in [22]. Di particolare interesse lo studio della tecnica applicata a sistemi di guida di missili, autori come Khamis [23] [24] hanno studiato l'applicazione per orizzonte a tempo finito ed infinito, per sistemi deterministici e stocastici, focalizzandosi su applicazioni missilistiche, fino a meccanismi più piccoli e complessi come mani robotiche [25]. L'articolo [26] di Topputo (2015) ha proposto una ulteriore implementazione del metodo che sfrutta maggiormente la possibilità di rappresentazione SDC. Definendo per il singolo sistema un set di matrici parametriche che in combinazione fra loro (mediante un $0 \leq \alpha \leq 1$) rappresenta il sistema e ne origina il controllo SDRE. Lo scopo è di utilizzare la rappresentazione più efficiente istante per istante, ma aggiunge ulteriori parametri al sistema. Si è approcciato il problema della stabilità espandendo lo studio della regione di attrazione, partendo dallo studio di stabilità locale nell'articolo [27] è stato proposto un metodo per calcolare la regione in modo dinamico al variare dello stato, per cercare di ovviare al problema irrisolto di stabilità globale. Oltre a questi contributi più teorici, negli ultimi anni sono numerose le applicazioni del metodo SDRE in diversi campi [28] reti di conversioni DC distribuite, [29] un Preview Controller per veicoli autonomi, [30] manovre di rendezvous in campo aereospaziale, [31] controllo di un robot a due ruote, [32] controller per l'analisi delle cure per il cancro.
In conclusione, una statistica sulla tecnica SDRE [33] mostra che quasi tre quarti delle pubblicazioni riguardano l'uso della tecnica SDRE come controller, di queste appena un terzo affronta l'aspetto teorico mentre il resto si concentra su applicazioni, mentre solo un quarto dei lavori applica la tecnica SDRE come

osservatore o filtro. La tecnica risulta mostrare caratteristiche molto interessanti, quali duttilità ed efficacia, ma necessita ancora di una struttura teorica salda che permetta di ottimizzare il controllo nel miglior modo possibile e nel minor sforzo possibile, sia in termini di tempo di realizzazione sia in termini di risorse necessarie.

Capitolo 3.

State-Dependent Coefficient parametrization

3.1. Struttura del sistema

Si consideri un sistema non lineare affine all'ingresso, nella forma:

$$\begin{cases} \dot{x} = f(x) + g(x)u \\ y = h(x) \end{cases} \tag{3.1}$$

L'affinità all'ingresso semplifica il problema, l'approccio può essere generalizzato eliminando questo vincolo complicando sensibilmente la tecnica di parametrizzazione. Lo scopo di questo approccio è di riportare il sistema ad una forma matriciale. Per questo fine, una volta scelte le variabili che compongono lo stato è possibile identificare un set di matrici A,B,C dipendenti dallo stato, tali che:

$$\begin{aligned} A(x)x &= f(x) \\ B(x) &= g(x) \\ C(x)x &= h(x) \end{aligned} \tag{3.2}$$

Il sistema assume la forma matriciale più comune:

$$\begin{cases} \dot{x} = A(x)x + B(x)u \\ y = C(x)x \end{cases} \tag{3.3}$$

Questa forma permette di applicare la classica funzione di costo quadratico:

$$\int (x'Q(x)x + u'R(x)u)\, dt \tag{3.4}$$

Nei capitoli successivi viene mostrato come calcolare la soluzione sfruttando le convenzioni note per il caso lineare e come queste sono influenzate dalla forma SDC. Un paragrafo è dedicato a come calcolare le matrici e come la

loro forma possa essere scelta arbitrariamente, garantendo un grado di libertà aggiuntivo per modellare il sistema di controllo. Tale molteplicità della forma del sistema ha un impatto su quanto efficacemente la rappresentazione cattura le caratteristiche del sistema; quali controllabilità, osservabilità ed autovalori.

3.2. Costruzione delle matrici

In questo capitolo sono raccolte le metodologie più comuni in letteratura per costruire le matrici SDC, viene proposto un schema per utilizzare tali metodi ed un nuovo approccio iterativo per scegliere come rappresentare al meglio il sistema scegliendo opportunamente i coefficienti.

Dato un sistema di equazioni $dx = f(x)$ si vuole ottenere una matrice $A(x)$ tale che $f(x) = A(x)x$.

Per prima cosa è utile catalogare le possibili funzioni non lineari. Sia f nella forma $f(x) = a_1(x) + a_2(x) + \cdots + a_n(x)$ dove ciascun coefficiente a_i è potenzialmente una funzione non lineare. Sia a_i appartenente al seguente insieme:

1. Esponenziale: x^n
2. Prodotto fra variabili di stato: $x_1 x_2$
3. Funzione trigonometrica: $\sin(x), \cos(x)$
4. Funzione logaritmica: $\log(x)$

Sia $c(x)$ il coefficiente dipendente dallo stato tale che $a_i = c_i(x)x$ dove $i = 1...n$. Le non linearità sopra elencate sono così trattate:

1. $x^n = x^{n-1}x = c_i(x)x$ quindi $c_i(x) = x^{n-1}$
2. In questo caso la soluzione è molteplice per ogni variabile di stato che fa parte del prodotto, ad esempio $x_1 = c_i(x)$ ed $x = x_2$ oppure $x_2 = c_i(x)$ ed $x = x_1$
3. L'approccio principale è sfruttare il limite notevole moltiplicando e dividendo per x, $\frac{\sin(x)}{x} = c_i(x)$ così per $x \to 0$ $c_i(x) \to 1$ [1];
   un approccio nuovo qui proposto è $1 = c_i(x)$ e $\sin(x_k) = \overline{x}$ ossia la variabile di stato è la funzione trigonometrica stessa, un cambio di variabile da x_k a $\overline{x}$, questa parametrizzazione verrà approfondita nei capitoli successivi
4. Similmente al caso precedente, si può sfruttare un limite notevole oppure modificare lo stato stesso

I punti due e tre rappresentano la prima modalità di accesso alle molteplici rappresentazioni in spazio di stato del medesimo sistema non lineare, messe a disposizione dalla parametrizzazione SDC. Un ulteriore metodo iterativo, qui proposto, amplia ulteriormente la possibilità di scelta della rappresentazione. Si ipotizzi di voler inserire un coefficiente in un termine della matrice per cambiare l'evoluzione del termine al variare dello stato (una sua traiettoria), influenzare gli autovalori della matrice o elidere un elemento nullo [8]. Si inseriscono due elementi non lineare la cui somma è nulla in $f(x)$ come ad esempio:

$$f(x) = f(x) + x_1x_2 - x_2x_1 \tag{3.5}$$

Data la scelta che il punto due consente, ci saranno due coefficienti diversi $c_1 = x_1$ e $c_2 = -x_2$; questo porta ad una matrice:

$$A = \begin{bmatrix} a_{11} + c_2 & a_{12} + c_1 \\ a_{21} & a_{22} \end{bmatrix}$$
$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$
$$f(x) + x_1x_2 - x_2x_1 = A(x)x$$

Apparentemente il contributo di questo artificio può essere trascurabile, ma in alcuni casi questo metodo porta il controllo da un sub-ottimo ad un ottimo, un esempio verrà illustrato nel capitolo dedicato. Per aumentare l'efficacia della rappresentazione SDC allo scopo di ottimizzare il controllo, si propone la seguente procedura iterativa:

1. Individuare il coefficiente da modificare valutando gli autovalori (ad esempio nulli o vicini a zero)
2. Selezionare la colonna della matrice A che più influenza l'autovalore, una prima ipotesi è scegliere la norma di ordine di grandezza più vicina all'autovalore
3. Aggiungere un prodotto misto $\alpha(x)\beta(x) - \alpha(x)\beta(x)$, in modo che un termine è posizionato nella colonna evidenziata al passo precedente alla riga più opportuna, mentre il secondo che annulla la il totale è in una colonna differente coerente con la struttura
4. Valutarne l'effetto e modificare le funzioni α, β con prove successive

A seguire sono riportati degli esempi, dal più semplice, un sistema a due equazioni con due sole non-linearità (esponenziale e prodotto misto), ad uno più complesso con diverse non linearità e funzioni trigonometriche. Durante l'attività di dottorato è stato studiato anche un modello di un veicolo a due ruote,

caratterizzato da una struttura fortemente non lineare, che è stato linearizzato solo per alcune delle variabili di sistema e rappresentato in spazio di stato con la tecnica SDRE. Questo studio è stato oggetto di pubblicazione [34], dove è stato validato il modello mediante confronto con un software multibody specifico. Tuttavia, la complessità di tale sistema, composto da minimo 12 equazioni, non ha permesso uno studio sull'impatto della rappresentazione SDRE con le tecniche illustrate nei capitoli successivi. Per tale fine sarà necessario in futuro mettere a punto delle tecniche iterative automatizzabili.

3.2.1. Esempio di sistema elementare

Di seguito un esempio semplice che risulta utile a capire la modalità di costruzione di matrici SDC e che è stato usato in vari lavori noti in letteratura [8]:

$$\begin{cases} \dot{x}_1 = -x_1 + x_1 x_2^2 \\ \dot{x}_2 = -x_2 + x_1 u \end{cases} \tag{3.6}$$

Questo sistema può essere facilmente ricondotto alla forma SDC in tre diverse combinazioni tutte coincidenti con $f(x)$:

$$A_1 = \begin{bmatrix} -1 + x_2^2 & 0 \\ 0 & -1 \end{bmatrix} \tag{3.7}$$

$$A_2 = \begin{bmatrix} -1 & x_1 x_2 \\ 0 & -1 \end{bmatrix} \tag{3.8}$$

$$A_3 = \begin{bmatrix} -1 & x_1 x_2 \\ -x_1 x_2 & -1 + x_1^2 \end{bmatrix} \tag{3.9}$$

$$B = \begin{bmatrix} 0 \\ x_1 \end{bmatrix} \tag{3.10}$$

Le matrici A_1 ed A_2 differiscono grazie al punto 2 della classificazione delle non linearità. Nel A_1 è stato scelto $c_2(x) = x_2^2$ e $x = x_1$ mentre in A_2 la combinazione è $c_2(x) = x_1 x_2$ e $x = x_2$. La matrice A_3 parte dalla scelta di A_2 ed aggiunge due termini che si elidono fra loro come visto in (3.5) dove $\alpha(x) = x_1^2$ e $\beta(x) = x_2$ nella seconda equazione.

3.2.2. Esempio di sistema fisico a media complessità: Doppio Pendolo

Un sistema noto per la sua complessità e per essere un sistema caotico è il doppio pendolo inverso, schema in figura 3.1, il suo moto è definito caotico se

lo stato iniziale ha angoli maggiori di 90°, questa caratteristica può evidenziare l'efficacia del metodo SDC e del controllo.

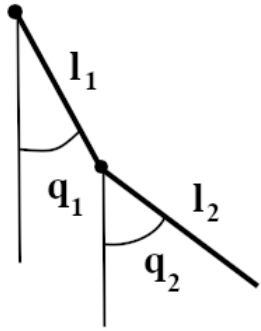


Figura 3.1.: Schema doppio pendolo

Di seguito le equazioni differenziali:

$$\begin{cases} \ddot{q}_1 = & -\frac{m_2 \cos(q_1-q_2)\sin(q_1-q_2)\dot{q}_1^2}{d} - \frac{l_2 m_2 \sin(q_1-q_2)\dot{q}_2^2}{l_1 d} + \\ & -\frac{g\sin(q_1)(m_1+m_2) - g m_2 \cos(q_1-q_2)\sin(q_2)}{l_1 d} - \frac{u}{l_1 d} \\ \ddot{q}_2 = & \frac{l_1 \sin(q_1-q_2)(m_1+m_2)\dot{q}_1^2}{l_2 d} + \frac{m_2 \cos(q_1-q_2)\sin(q_1-q_2)\dot{q}_2^2}{d} + \\ & -\frac{g\sin(q_2)(m_1+m_2) - g\cos(q_1-q_2)\sin(q_1)(m_1+m_2)}{l_2 d} + \frac{u\cos(q_1-q_2)}{l_2 d} \end{cases} \tag{3.11}$$

$$d = -m_2 \cos(q_1 - q_2)^2 + m_1 + m_2$$

Dove g è l'accelerazione di gravità, m_1 ed m_2 sono le masse del primo e secondo braccio, l_1 ed l_2 sono le lunghezze del primo e secondo braccio, q_1 ed q_2 sono gli angoli dei rispettivi bracci considerando lo zero nel punto di equilibrio stabile verso il basso ed u è l'attuazione che agisce sul primo giunto. Per semplicità, si ipotizzino dei parametri pari a $m_1 = 0.1\,\text{kg}$, $m_2 = 0.1\,\text{kg}$, $l_1 = 0.1\,\text{m}$, $l_1 = 0.1\,\text{m}$. Il problema fondamentale del sistema è la sua stretta dipendenza dalle funzioni trigonometriche. Queste rendono più complesso allocare i coefficienti nella matrice della dinamica dipendente dallo stato. Occorre utilizzare limiti notevoli come illustrato nel paragrafo precedente al passo 3, cosicché il terzo termine di ciascuna delle due equazioni differenziali possa essere inserito nella matrice A. Trasformando le equazioni con tali artifici matematici si ottiene la rappresentazione:

$$
\begin{aligned}
X &= \begin{bmatrix} q_1 \\ q_2 \\ \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} \quad A = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \\
B &= \begin{bmatrix} 0 & 0 & b_{31} & b_{41} \end{bmatrix} \\
a_{31} &= \frac{294.3 \sin(q_1)}{q_1 d} + \frac{98.1 \sin(q_1)\cos(2q_2)}{q_1 d} \\
a_{32} &= -\frac{98.1 \cos(q_1)\sin(2q_2)}{q_2 d} \\
a_{33} &= \frac{\sin(2q_1 - 2q_2)\dot{q}_1}{d} \\
a_{34} &= \frac{2 \sin(q_1 - q_2)\dot{q}_2}{d} \\
a_{41} &= -\frac{196.2 \sin(2q_1)\cos(q_2)}{q_1 d} \\
a_{42} &= \frac{196.2 \sin(q_2)}{q_2 d} + \frac{196.2 \cos(2q_1)\sin(q_2)}{q_2 d} \\
a_{43} &= -\frac{4 \sin(q_1 - q_2)\dot{q}_1}{d} \\
a_{44} &= -\frac{\sin(2q_1 - 2q_2)\dot{q}_2}{d} \\
b_{31} &= \frac{200}{d} \\
b_{41} &= -\frac{200 \cos(q_1 - q_2)}{d} \\
d &= \cos(2q_1 - 2q_2) - 3
\end{aligned}
\tag{3.12}
$$

Il problema diviene implementativo, poiché quando il sistema passa nel punto di singolarità introdotto dall'artificio occorre modificare la rappresentazione con un'approssimazione, evitando l'errore numerico nella singolarità che mina l'efficacia del sistema. In pratica, ogni limite notevole utilizzato aggiunge a denominatore una variabile di stato (come per a_{31},a_{32},a_{41},a_{42}) che al tendere a zero generano un problema numerico nel punto singolare. Per ovviare, occorre definire un intervallo ($-0.1 < q_1 < 0.1$) dove il coefficiente $\sin(q_1)/q_1$ è approssimato al valore 1.
Si propone una rappresentazione alternativa che evita l'artificio matematico su menzionato. Se si rappresenta con una variabile di stato l'intera funzione trigonometrica dell'angolo anziché solo quest'ultimo, si ottiene una rappresentazione SDC più semplice ed efficace. Siano definite le nuove variabili come segue:

$$
\begin{aligned}
s_q &= \sin(q) & c_q &= \cos(q) \\
x_1 &= s_{q_1} & x_2 &= s_{q_2} \\
\dot{x}_1 &= c_{q_1}\dot{q}_1 & \dot{x}_2 &= c_{q_2}\dot{q}_2
\end{aligned}
$$

Il sistema può essere scritto nella forma:

$$
\begin{aligned}
X &= \begin{bmatrix} s_{q_1} \\ s_{q_2} \\ \dot{q}_1 \\ \dot{q}_2 \end{bmatrix} \quad A = \begin{bmatrix} -s_{q_1}\dot{q}_1 & 0 & c_{q_1} + s_{q_1}^2 & 0 \\ 0 & -s_{q_2}\dot{q}_2 & 0 & c_{q_2} + s_{q_2}^2 \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \\
B &= \begin{bmatrix} 0 & 0 & b_{31} & b_{41} \end{bmatrix} \\
a_{31} &= -\frac{196.2(1 + c_{q_2}^2)}{d} \\
a_{32} &= \frac{196.2\, c_{q_1} c_{q_2}}{d} \\
a_{33} &= 2\frac{\dot{q}_1(2\, s_{q_2} c_{q_1}^2 c_{q_2} - 2\, s_{q_1} c_{q_1} c_{q_2}^2 + s_{q_1} c_{q_1} - s_{q_2} c_{q_2})}{d} \\
a_{34} &= 2\frac{\dot{q}_2(s_{q_2} c_{q_1} - s_{q_1} c_{q_2})}{d} \\
a_{41} &= 2\frac{196.2(s_{q_1} s_{q_2} + c_{q_1} c_{q_2})}{d} \\
a_{42} &= -2\frac{196.2}{d} \\
a_{43} &= 2\frac{\dot{q}_1(2\, c_{q_2} s_{q_1} - 2\, c_{q_1} s_{q_2})}{d} \\
a_{44} &= 2\frac{\dot{q}_2(c_{q_2} s_{q_2} - c_{q_1} s_{q_1} + 2\, c_{q_1} c_{q_2}^2 s_{q_1} - 2\, c_{q_1}^2 c_{q_2} s_{q_2})}{d}
\end{aligned}
\tag{3.13}
$$

Le nuove equazioni ausiliarie sono ottenute sostituendo le nuove variabili alla forma vista nel caso precedente e utilizzando l'artificio (3.5). La scelta ha un costo, il controllo non è più in grado di distinguere autonomamente fra il punto di equilibrio 0° e 180° così come 90° e 270°. Il sistema si stabilizzerà verso il punto di equilibrio più vicino da cui è partito nello stato iniziale. Per ovviare a tale difetto occorre un controllore più esterno che sposta il sistema più vicino al punto di equilibrio desiderato. Da notare che, nel caso fosse necessario percorrere più giri o avvicinarsi da una direzione oraria anziché antioraria, il sistema nella prima rappresentazione non distingue la periodicità dei giri multipli o versi (180°=−180° o 360°=0°), quindi anche in questo caso occorre un supervisore per ovviare a tale problema qualora fosse richiesto.

3.3. Caratterizzazione del sistema SDC

La caratterizzazione SDC permette di rappresentare lo stesso sistema non lineare mediante l'uso di matrici differenti. Questa scelta piuttosto che essere arbitraria potrebbe essere guidata dallo studio delle caratteristiche del sistema stesso, quali raggiungibilità, osservabilità ed autovalori (questi ultimi nel caso di caratterizzazione "pointwise"). In letteratura [2], sono state discusse sia raggiungibilità che osservabilità "pointwise", in esse la condizione di rango pieno è calcolata in modo simile alla condizione di controllabilità e di osservabilità dei sistemi lineari ma considerando le rispettive matrici dipendenti dallo stato, è verificata per ogni punto del dominio dello stato. [7] ha dimostrato l'ottimalità locale del metodo così discusso, mentre [8] ha esposto il limite di questo approccio, ovvero che la raggiungibilità pointwise non è detto coincidere con la raggiungibilità non lineare del sistema. Se il sistema non lineare viene studiato a livello generale, ovvero riferendosi dalle funzioni non lineari del sistema, si può pensare di verificare la condizione di raggiungibilità con l'algebra di Lie. Allo scopo di legare la specifica rappresentazione matriciale del sistema non lineare alla sua condizione di raggiungibilità non lineare, di seguito sono proposti i calcoli che permettono appunto di legare la condizione di raggiungibilità alla specifica matrice $A(x)$, sulla base dei quali è possibile trarre delle nuove considerazioni sulla tecnica SDC.

E' noto [35] che un sistema non lineare è localmente raggiungibile se esiste una distribuzione non singolare di dimensione minima che sia involutiva, che contenga le colonne di $B(x)$ e che sia invariante sotto l'azione dei campi vettoriali f e g_i. L'esistenza dell'elemento minimo non singolare nella famiglia di distribuzioni è generata dall'insieme:

$$R = \{g; [f; g]\} \tag{3.14}$$

in cui le parentesi quadre rappresentano la Lie bracket:

$$[f; g] = \frac{\partial g}{\partial x} f - \frac{\partial f}{\partial x} g \tag{3.15}$$

Nel seguito, per semplicità di trattazione si utilizzerà un sistema di dimensione 2, i calcoli possono essere poi facilmente estesi al caso di dimensione n. Ricordando che per l'equazione (3.2) si può scrivere la raggiungibilità in spazio di stato come:

$$R = \{g; [A(x)x; g]\} \tag{3.16}$$

Se $A(x)x$ fosse lineare ovvero se semplicemente $A(x)$=A le Lie Bracket porterebbero alla nota matrice di raggiungibilità, la stessa che è utilizzata nel metodo pointwise. Questa forma più complessa può essere rappresentata in modo da

agevolare il calcolo. Nel caso dimensione 2, la matrice $A(x)$ è 2x2, mentre x è dimensione 2x1, questo prodotto risulta il vettore f di dimensione 2x1, ma può essere scritto come una sommatoria di vettori colonna di A rappresentati come $a_i(x)$ dove $i = 1, 2$, moltiplicati per la singola variabile di stato $x = [x_1, x_2]^T$, come segue:

$$\begin{aligned} &f = A(x)x = a_1(x)x_1 + a_2(x)x_2 \\ &dove\ A(x) = [a_1(x)\ a_2(x)] = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}\ ed\ x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \end{aligned} \tag{3.17}$$

Tale scelta permette di calcolare la Lie bracket più agevolmente, a_i sono vettori colonna ed x_i sono scalari.

$$\begin{aligned} [f; g] = [Ax; g] = [(a_1x_1 + a_2x_2); g] = [a_1x_1; g] + [a_2x_2; g] = \\ = [a_1; g]x_1 - a_1\mathcal{L}_g(x_1) + [a_2; g]x_2 - a_2\mathcal{L}_g(x_2) \end{aligned}$$

questo poiché risulta

$$\begin{aligned} [a_ix_i; g] &= \frac{\partial g}{\partial x}a_ix_i - \frac{\partial(a_ix_i)}{\partial x}g = \\ &= \frac{\partial g}{\partial x}a_ix_i - \left(\frac{\partial a_i}{\partial x}x_i + a_i\frac{\partial x_i}{\partial x}\right)g = \\ &= \left(\frac{\partial g}{\partial x}a_i - \frac{\partial a_i}{\partial x}g\right)x_i - a_i\frac{\partial x_i}{\partial x}g = \\ &= [a_i; g]x_i - a_i\mathcal{L}_g(x_i) \end{aligned}$$

Dove $\mathcal{L}_g$ è la Lie derivative ed in questo caso può essere calcolata semplicemente:

$$\begin{aligned} \mathcal{L}_g(x_1) &= \frac{\partial x_1}{\partial x}g(x) = [1\ 0]\begin{bmatrix} g_1 \\ g_2 \end{bmatrix} = g_1 \\ [a_1; g] &= \frac{\partial g}{\partial x}a_1 - \frac{\partial a_1}{\partial x}g = J_ga_1 - J_{a_1}g \end{aligned}$$

Dove $J_g(x)$ è il jacobiano della componente g affine all'ingresso rispetto lo stato, similmente per J_{a_i} è il jacobiano della i-esima colonna rispetto lo stato. L'equazione risultante può essere riordinata e raggruppata come segue:

$$\begin{aligned} &(J_ga_1 - J_{a_1}g)x_1 - a_1g_1 + (J_ga_2 - J_{a_2}g)x_2 - a_2g_2 = \\ &J_g(a_1x_1 + a_2x_2) - (J_{a_1}x_1 + J_{a_2}x_2)g - (a_1g_1 + a_2g_2) \end{aligned} \tag{3.18}$$

Ricordando che la derivata del vettore f è il jacobiano, se si calcola la derivata

nella forma SDC si ottiene una relazione utile:

$$\begin{aligned}
&\frac{\partial f(x)}{\partial x} = J(x) \\
&f(x) = A(x)x \\
&\frac{\partial f(x)}{\partial x} = \frac{\partial (A(x)x)}{\partial x} = \frac{\partial A(x)}{\partial x}x + A(x)\frac{\partial x}{\partial x} = \frac{\partial A(x)}{\partial x}x + A(x) \\
&J(x) = \frac{\partial A(x)}{\partial x}x + A(x)
\end{aligned} \tag{3.19}$$

Da notare la derivata di una matrice rispetto ad un vettore $\frac{\partial A(x)}{\partial x}$, questa è calcolata derivando la matrice $A(x)$ per ogni elemento di x moltiplicato a sua volta per il vettore x, organizzando queste derivate in un vettore si ottengono vettori colonna che sommati ad $A(x)$ riportano $J(x)$. Si può rappresentare come segue:

$$\frac{\partial A(x)}{\partial x}x = \left[\frac{\partial A(x)}{\partial x_1}x; \frac{\partial A(x)}{\partial x_2}x; \ldots; \frac{\partial A(x)}{\partial x_n}x\right]$$

Possiamo verificare il risultato ottenuto in (3.18) riportando in forma matriciale le sommatorie di vettori, mentre per il secondo termine che è una somma di prodotti di matrici 2x2 per uno scalare si ottiene:

$$\begin{aligned}
&J_g(a_1x_1 + a_2x_2) - (J_{a_1}x_1 + J_{a_2}x_2)g - (a_1g_1 + a_2g_2) = \\
&J_g(Ax) - \left(\frac{\partial A}{\partial x}x\right)g - (Ag) = \\
&J_gAx - \left(\frac{\partial A}{\partial x}x + A\right)g = \\
&J_gAx - Jg = [f; g]
\end{aligned}$$

Il risultato (3.18) può essere generalizzato per un sistema di ordine n, tenendo presente che x è sempre un vettore colonna i calcoli restano validi, risulta:

$$\begin{aligned}
[f; g] = [Ax; g] &= \left(\sum_{i=1}^{n}[a_i; g]x_i\right) - \left(\sum_{i=1}^{n}a_i\mathcal{L}_g(x_i)\right) = \\
&= J_g\left(\sum_{i=1}^{n}a_ix_i\right) - \left(\sum_{i=1}^{n}J_{a_i}x_i\right)g - \left(\sum_{i=1}^{n}a_ig_i\right)
\end{aligned} \tag{3.20}$$

Dal risultato finale, si vede come la sola matrice di raggiungibilità che normalmente descriverebbe un sistema lineare non rappresenta per intero la caratteristica, in quanto mancherebbe dei termini della seconda e terza sommatoria in (3.20). Se g è invariante rispetto allo stato allora il jacobiano di g J_g è nullo e la prima sommatoria si elide. La formula rappresenta la raggiungibilità completa del sistema non lineare, la differente forma della matrice $A(x)$ fa variare

i singoli termini dell'equazione lasciando il risultato invariato. Se identificato qual'è l'equilibrio più efficiente degli elementi dei singoli termini di (3.20) che porta ad un controllo più efficace, si ottiene un valido strumento per valutare la forma della matrice $A(x)$ scelta.
Nasce spontaneo un parallelismo con il metodo più semplice di linearizzazione, dove si approssima il sistema con taylor al primo ordine in zero. La matrice della dinamica è semplicemente il jacobiano del sistema $A = J = \frac{\partial f}{\partial x}$. Osservando la relazione (3.19) si evidenzia l'errore di approssimazione del metodo nel termine mancante $\frac{\partial A(x)}{\partial x}x$. L'errore di approssimazione cresce tanto più la velocità con cui varia lo stato del sistema è maggiore.
La trattazione è del tutto analoga per lo studio di osservabilità. Per semplicità si considereranno sistemi con feedback dallo stato.

3.3.1. Esempio di un sistema elementare

Si consideri il sistema (3.6) nelle sue tre rappresentazioni A_1, A_2 ed A_3 equivalenti. Si calcoli la raggiungibilità con l'equazione (3.20). Considerando la formulazione classica con f e g:

$$R = \begin{bmatrix} 0 & -2x_1^2x_2 \\ x_1 & x_1x_2^2 \end{bmatrix}$$

Si ottiene per il sistema A_1:

$$[a_1; g]x_1 = \begin{bmatrix} -2x_1x_2 \\ x_2^2 - 1 \end{bmatrix} x_1$$
$$[a_2; g]x_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} x_2$$
$$a_1\mathcal{L}_g(x_1) + a_2\mathcal{L}_g(x_2) = \begin{bmatrix} 0 \\ -x_1 \end{bmatrix}$$

Si ottiene per il sistema A_2:

$$[a_1; g]x_1 = \begin{bmatrix} 0 \\ -1 \end{bmatrix} x_1$$
$$[a_2; g]x_2 = \begin{bmatrix} -x_1^2 \\ x_1x_2 \end{bmatrix} x_2$$
$$a_1\mathcal{L}_g(x_1) + a_2\mathcal{L}_g(x_2) = \begin{bmatrix} x_1^2x_2 \\ -x_1 \end{bmatrix}$$

Si ottiene per il sistema A_3:

$$[a_1; g]x_1 = \begin{bmatrix} 0 \\ x_1^2 - 1 \end{bmatrix} x_1$$

$$[a_2; g]x_2 = \begin{bmatrix} -x_1^2 \\ x_1 x_2 \end{bmatrix} x_2$$

$$a_1\mathcal{L}_g(x_1) + a_2\mathcal{L}_g(x_2) = \begin{bmatrix} x_1^2 x_2 \\ x_1(x_1^2 - 1) \end{bmatrix}$$

Tutte e tre le configurazioni portano alla seconda colonna di R come previsto. Alla luce dell'efficacia dei sistemi di controllo mostrati dai tre sistemi si può considerare se esiste un legame fra la scomposizione dei termini sopra calcolati che esprimono la stessa colonna in modi differenti e l'efficacia stessa. Nel caso della matrice A_1 si può notare che un termine porta ad una colonna nulla, è possibile che ciò sia legato alla scarsa efficacia di tale rappresentazione. Mentre la differenza sostanziale fra A_2 ed A_3 sembra essere nella derivata di Lie più complessa per la terza rappresentazione.

3.4. Valutare la rappresentazione

Essendo chiaro che la parametrizzazione SDC permette infinite rappresentazioni dello stesso sistema, occorre un metodo per identificare la qualità di quella scelta. Essendo sempre valida l'equazione (3.2) (f=A(x)x) ciò che è di maggiore interesse è come la specifica matrice A(x) impatterà sul controllo del sistema non lineare.

Si consideri l'esempio semplice (3.6) La forma A_1 3.7 si è dimostrata inefficace già verificando la matrice di controllabilità che riporta determinante nullo. Simulando la risposta al controllo nella condizione iniziale $x_1 = 2$, $x_2 = 2$, nota la soluzione ottima calcolata dall'equazione Hamilton-Jacobi-Bellman utilizzando $Q = [2,\ 0;\ 0,\ 2]$ e $R = 0.5$ è $u_o = -2x_1x_2$ [8], il controllo di A_2 3.8 si dimostra sub-ottimo. In [8] viene proposta la struttura A_3 3.9, costruita con l'artificio matematico esposto nel 3.2. Questo controllo con la stessa matrice Q ha un comportamento assimilabile a quello del controllo ottimo. Tale esempio mette in luce l'importanza di scegliere la giusta rappresentazione, cosa che ad oggi resta molto complessa non essendoci indicatori univoci che permettono di discernere l'efficacia del controllo senza conoscerne la soluzione ottima.

3.4.1. Determinante ed Autovalori

Nel caso lineare un'indicazione significativa della dinamica del sistema è data dagli autovalori. Nel caso non lineare in analisi occorre rappresentare l'anda-

mento numerico degli autovalori al variare dello stato del sistema. Questo lo rende poco efficiente per dimensioni elevate ma comunque praticabile utilizzando un calcolatore adeguato. Di seguito un'analisi del determinante e degli autovalori delle matrici A_1, A_2, A_3, utili allo scopo di valutare il comportamento di questi valori ed il loro effetto sul controllo.

Per la prima matrice si può notare che il primo autovalore (fig. 3.3) è co-

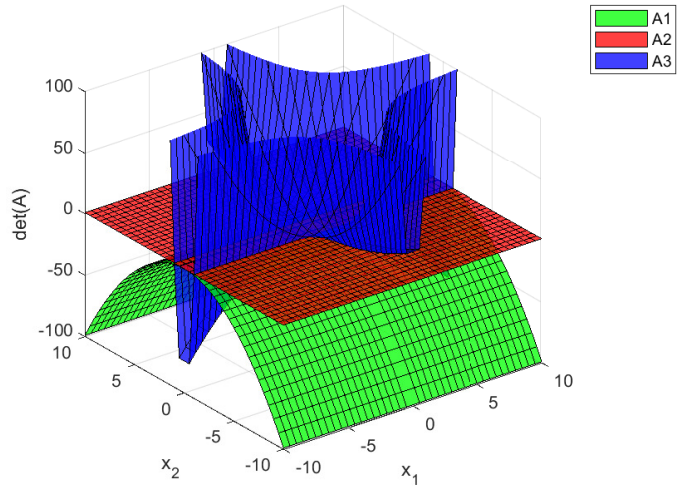


Figura 3.2.: Andamento determinante matrice A (det(A)) al variare dello stato (x_1, x_2); in verde A_1, rosso A_2, blu A_3

stante a -1 ed il determinante (fig. 3.2) tende a $-\infty$ al variare di x_2, con un piccolo cambiamento della matrice la differenza è netta. Il determinante della A_2 è costante ad un valore di 1, l'autovalore λ_1 resta costante ma anche λ_2 è costante. Potrebbe sembrare una situazione desiderabile, in quanto la matrice A_2 conserva le sue caratteristiche al variare di x_1 ed x_2, ma nel caso non lineare questo si traduce in una scarsa adattabilità del controllo. Cercando di interpretare il miglior comportamento mostrato nel controllo da [8], la matrice A_2 non cattura la non linearità del sistema nelle sue caratteristiche. Mentre la A_3 è fortemente influenzata dai valori dello stato sia nel determinante che negli autovalori. Si noti come il determinante passi rapidamente da $-\infty$ a $+\infty$ passando quindi per zero, ma la rapidità con cui cambia dovrebbe permettere al sistema di essere mal condizionato solo in rari e brevissimi istanti.
Di grande importanza è evidenziare che gli autovalori pointwise non danno alcuna indicazione sulla stabilità del sistema non lineare. In questo esempio, sebbene nel punto 2,2 almeno un autovalore sia maggiore di zero in tutti e tre i casi, il sistema reale in evoluzione libera tende a stabilizzarsi con un tempo

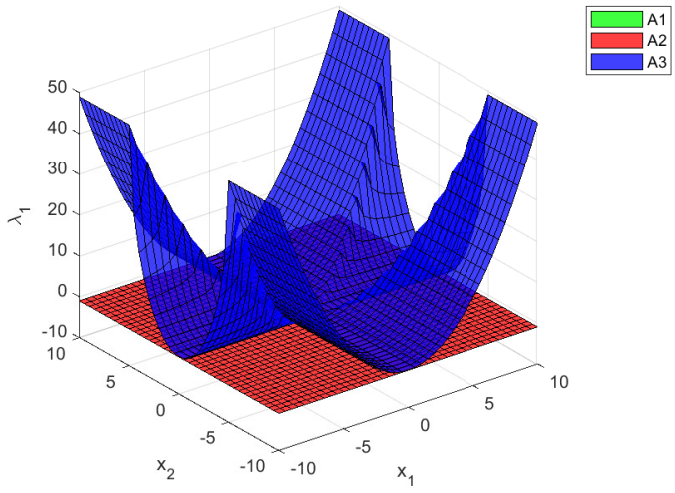


Figura 3.3.: Andamento del primo autovalore della matrice A al variare dello stato (x_1, x_2); in verde A_1, rosso A_2, blue A_3. A_1 e A_2 si sovrappongono.

maggiore di quello controllato tanto più è grande la x_1 iniziale. Se invece si "congelasse" la matrice nel punto iniziale come fosse una linearizzazione, la traiettoria divergerebbe.
Al fine di valutare l'efficacia del controllo potrebbe essere ancor più interessante calcolare l'andamento del determinante ed autovalori della matrice dinamica a ciclo chiuso. Ipotizzando un controllo LQR pointwise (in letteratura SDRE, illustrato nei capitoli successivi) e che lo stato iniziale sia $(x_1, x_2) = (2, 2)$, si calcola la matrice K di feedback e quindi quella a ciclo chiuso ha la forma $(A - BK)$.

Per la matrice A_2, gli autovalori pointwise a ciclo chiuso hanno un andamento molto simile fra loro (fig. 3.6 3.7), con valori fortemente negativi. Al variare della matrice Q la superficie cambia significativamente, partendo dalla diagonale 2,2 ad una quadrata con valore 10 anche su extra diagonale, si può notare un comportamento accentuato dei valori. Quando la matrice dipende dallo stato, con una diagonale x^2, gli autovalori tendono a $-\infty$ più rapidamente. Il determinante della matrice ha un andamento concettualmente simile, fig. 3.5, al variare della matrice Q il valore cresce più rapidamente.
Nel caso della matrice A_3 (fig. 3.9 3.10), si può notare come il comportamento sia meno differente per le diverse matrici di peso. Mentre al variare dello stato, i valori tendono a muoversi su una curva rispetto solo una delle due variabili. Solo la matrice di peso dipendente dallo stato ha una superficie più accentuata

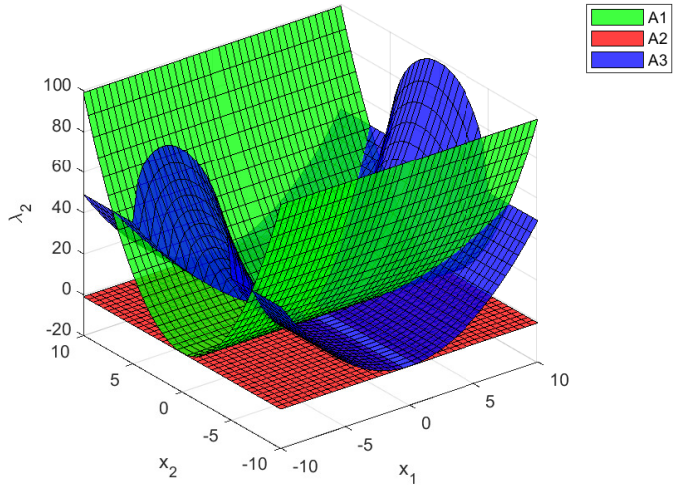


Figura 3.4.: Andamento del secondo autovalore della matrice A al variare dello stato (x_1, x_2); in verde A_1, rosso A_2, blue A_3.

verso $-\infty$. Come nel caso precedente, il determinante in figura 3.8, permette di trarre conclusioni simili sulla variazione della matrice di peso. Da queste prove si potrebbe ipotizzare che la matrice A_3 ha un comportamento migliore nel controllo poiché sintetizza meglio il comportamento non lineare nella dinamica e rende la scelta della matrice Q meno influente sull'efficacia del controllo. Tuttavia, si può notare come la dipendenza dallo stato di Q influenzi maggiormente il risultato per entrambe le matrici, questo potrebbe essere una soluzione complementare al problema di ottimizzare il controllo unicamente con la scelta della matrice $A(x)$.

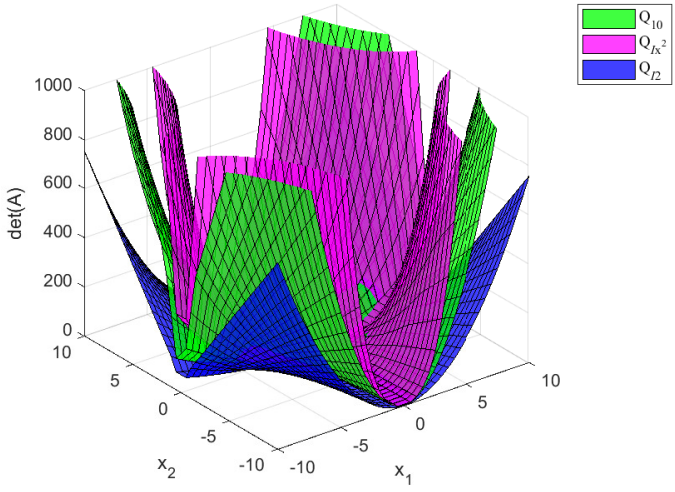


Figura 3.5.: Andamento determinante matrice $(A_2 - BK)$ al variare dello stato (x_1, x_2); in verde Q costante a valore 10, magenta Q diagonale con dipendenza dallo stato, blu Q diagonale costante a valore 2.

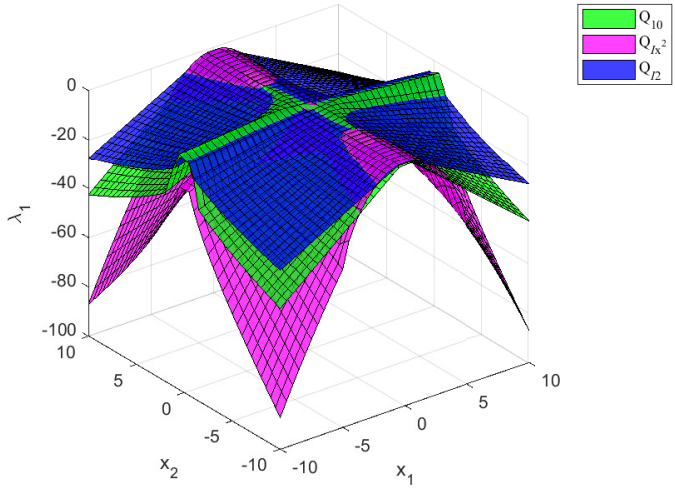


Figura 3.6.: Andamento del primo autovalore della matrice $(A_2 - BK)$ al variare dello stato (x_1, x_2); in verde Q costante a valore 10, magenta Q diagonale con dipendenza dallo stato, blu Q diagonale costante a valore 2.

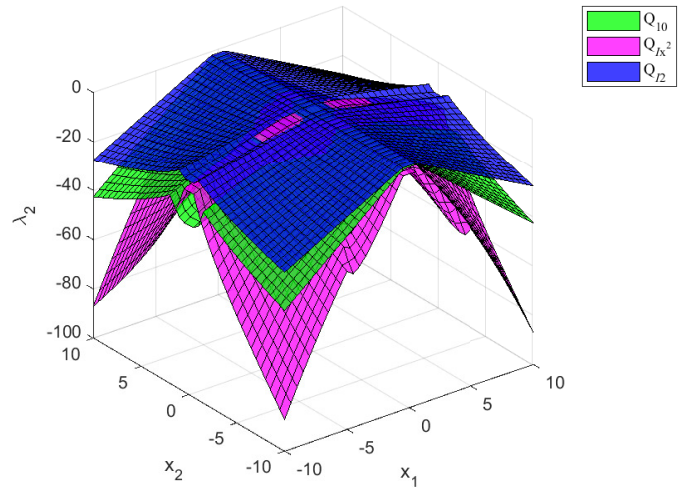


Figura 3.7.: Andamento del secondo autovalore della matrice $(A_2 - BK)$ al variare dello stato (x_1, x_2); in verde Q costante a valore 10, magenta Q diagonale con dipendenza dallo stato, blu Q diagonale costante a valore 2.

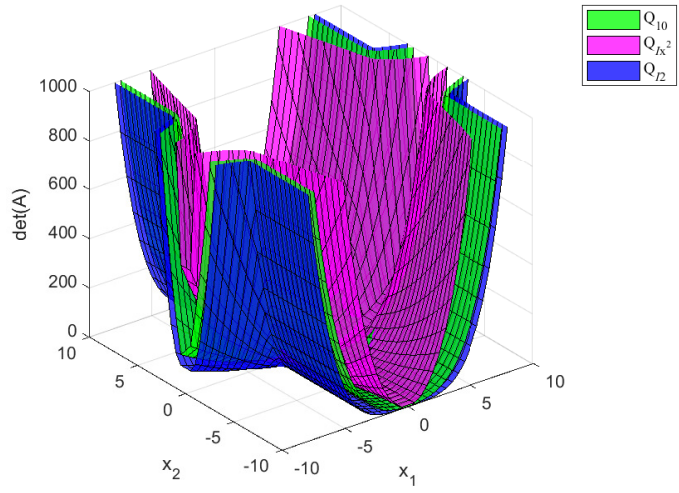


Figura 3.8.: Andamento determinante matrice $(A_3 - BK)$ al variare dello stato (x_1, x_2); in verde Q costante a valore 10, magenta Q diagonale con dipendenza dallo stato, blu Q diagonale costante a valore 2.

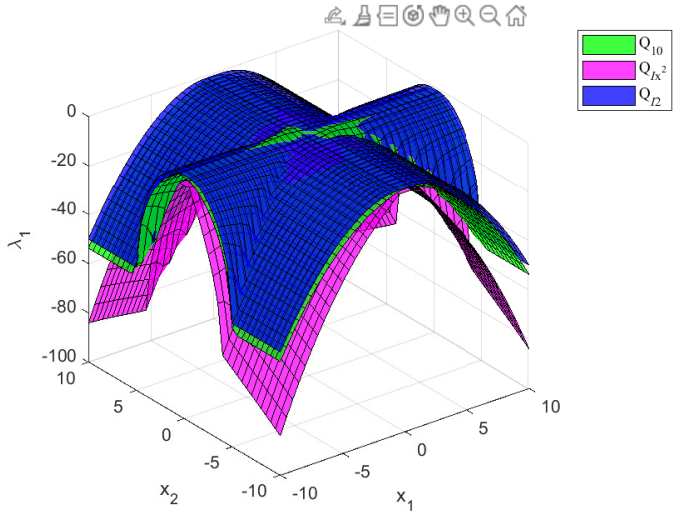


Figura 3.9.: Andamento del primo autovalore della matrice $(A_3 - BK)$ al variare dello stato (x_1, x_2); in verde Q costante a valore 10, magenta Q diagonale con dipendenza dallo stato, blu Q diagonale costante a valore 2.

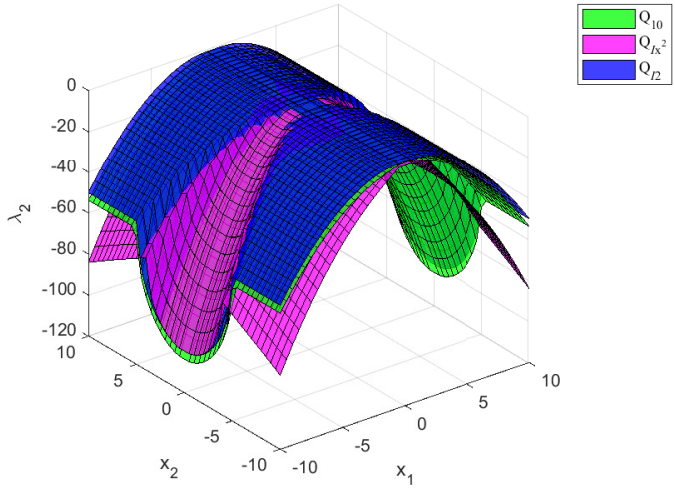


Figura 3.10.: Andamento del secondo autovalore della matrice $(A_3 - BK)$ al variare dello stato (x_1, x_2); in verde Q costante a valore 10, magenta Q diagonale con dipendenza dallo stato, blu Q diagonale costante a valore 2.

Capitolo 4.

Quadratic Regulator SDC

Attraverso la ben nota funzione di costo quadratico (3.4), in questo caso anch'essa dipendente dallo stato, è possibile progettare un controllo analogo al Linear Quadratic Regulator (LQR). Seguendo ipotesi simili al caso lineare con ulteriori condizioni imposte dalla non linearità delle matrici si giunge all'equazione di Riccati algebrica dipendente a sua volta dallo stato (SDRE). Di seguito vengono ripercorsi i passaggi per arrivare alla soluzione tenendo conto della dipendenza dallo stato delle matrici della dinamica e di peso.

4.1. Equazione Hamilton-Bellman

Partendo dalla nota equazione di Hamilton-Bellman tenendo conto della dipendenza dello stato si ha:

$$\begin{aligned} H(x(t), u(t), \lambda(x(t), t), t) &= \frac{1}{2} x^T(t) Q(x(t)) x(t) + \frac{1}{2} u^T(t) R u(t) \\ &\quad + \lambda^T(x(t), t)[A(x(t))x(t) + B(x(t))u(t)] \\ u^*(x(t)) &= -R^{-1} B^T(x(t)) \lambda(x(t), t) \end{aligned} \tag{4.1}$$

Dove $\lambda(x(t))$ è il vettore di costato e $u^*(x(t))$ è lo sforzo di controllo ottimo. Sospendendo la notazione di dipendenza dal tempo di x per semplicità e derivando l'hamiltoniana rispetto al costato e stato si ottiene il sistema:

$$\begin{cases} \dot{x} = \frac{\partial H}{\partial \lambda} = A(x)x + B(x)u \\ \dot{\lambda} = -\frac{\partial H}{\partial x} = -\frac{1}{2}\frac{\partial}{\partial x}(x^T Q(x) x) - \frac{\partial}{\partial x}(\lambda^T A(x) x) - \frac{\partial}{\partial x}(\lambda^T B(x) u) \end{cases} \tag{4.2}$$

Per calcolare le derivate della Hamiltoniana rispetto il vettore x occorre calcolare la derivata di un prodotto di vettori e matrici, noto che $\frac{\partial x^T A x}{\partial x} = (A + A^T)x$, nel caso che A dipenda da x stessa occorre una derivazione di prodotto di funzioni in più tenendo conto della trasposizione necessaria a rispettare le somme

ed i prodotti di matrice. Quindi vale:

$$\begin{aligned}\frac{\partial x^T A(x)x}{\partial x} &= \frac{\partial x^T A(x)}{\partial x}x + \left(x^T A(x)\frac{\partial x}{\partial x}\right)^T = \\ &= \left(\frac{\partial x^T}{\partial x}A(x) + x^T\frac{\partial A(x)}{\partial x}\right)x + A^T(x)x = \\ &= (A(x) + A^T(x))x + x^T\frac{\partial A(x)}{\partial x}x\end{aligned}$$

$$\frac{\partial}{\partial x}(x^T Q(x)x) = 2Q(x)x + x^T\frac{\partial Q(x)}{\partial x}x \tag{4.3}$$

$$\frac{\partial}{\partial x}(\lambda^T A(x)x) = \left(\frac{\partial \lambda^T}{\partial x}A(x) + \lambda^T\frac{\partial A(x)}{\partial x}\right)x + A^T(x)\lambda \tag{4.4}$$

$$\frac{\partial}{\partial x}(\lambda^T B(x)u) = \frac{\partial \lambda^T}{\partial x}B(x)u + \lambda^T\frac{\partial B(x)}{\partial x}u \tag{4.5}$$

Si ipotizza la soluzione del costato e sua derivata nel tempo come:

$$\begin{aligned}\lambda(x(t),t) &= P(x(t),t)x(t) \\ \dot{\lambda}(x(t),t) &= \dot{P}(x(t),t)x + P(x(t),t)\dot{x}(t)\end{aligned} \tag{4.6}$$

Inserendo tali equazioni nel sistema (4.2), sostituendo ad u l'ottimale in (4.1) e risolvendo rispetto alla seconda equazione, si ottiene:

$$\begin{aligned}&\dot{P}(x)x + P(x)[A(x)x + B(x)(-R^{-1}B^T(x)P(x)x)] = -\frac{1}{2}\left(2Q(x)x + x^T\frac{\partial Q(x)}{\partial x}x\right) + \\ &-\left(\frac{\partial (P(x)x)^T}{\partial x}A(x) + (P(x)x)^T\frac{\partial A(x)}{\partial x}\right)x - A^T(x)P(x)x + \\ &-\frac{\partial (P(x)x)^T}{\partial x}B(x)(-R^{-1}B^T(x)P(x)x) - (P(x)x)^T\frac{\partial B(x)}{\partial x}(-R^{-1}B^T(x)P(x)x)\end{aligned}$$

$$\begin{aligned}&\dot{P}(x)x + P(x)[A(x)x - E(x)P(x)x] = -\frac{1}{2}\left(2Q(x)x + x^T\frac{\partial Q(x)}{\partial x}x\right) + \\ &-\left(\left(\frac{\partial x^T}{\partial x}P(x) + x^T\frac{\partial P(x)}{\partial x}\right)A(x) - x^T P(x)\frac{\partial A(x)}{\partial x}\right)x + A^T(x)P(x)x + \\ &-\left(\frac{\partial x^T}{\partial x}P(x) + x^T\frac{\partial P(x)}{\partial x}\right)(-E(x)P(x)x) - x^T P(x)\frac{\partial B(x)}{\partial x}(-B^{-1}(x)E(x)P(x)x) \\ &con \quad E(x) = B(x)R^{-1}B^T(x)\end{aligned}$$

$$\dot{P}(x)x + P(x)A(x)x - P(x)E(x)P(x)x = -Q(x)x - \frac{1}{2}x^T\frac{\partial Q(x)}{\partial x}x+$$
$$- P(x)A(x)x - x^T\frac{\partial P(x)}{\partial x}A(x)x - x^TP(x)\frac{\partial A(x)}{\partial x}x - A^T(x)P(x)x+$$
$$+ P(x)E(x)P(x)x + x^T\frac{\partial P(x)}{\partial x}E(x)P(x)x + x^TP(x)\frac{\partial B(x)}{\partial x}B^{-1}(x)E(x)P(x)x$$

Eliminando il vettore x che motiplica tutti i termini e sospendendo la notazione di dipendenza dallo stato per le matrici si ottiene:

$$\dot{P} = -2PA - A^TP + 2PEP - Q+$$
$$- \frac{1}{2}x^T\frac{\partial Q}{\partial x} - x^T\frac{\partial P}{\partial x}A - x^TP\frac{\partial A}{\partial x}+$$
$$+ x^T\frac{\partial P}{\partial x}EP + x^TP\frac{\partial B}{\partial x}B^{-1}EP$$

Se Le variazioni delle matrici rispetto allo stato tendono a 0, si ottiene essattamente l'equazione di riccati a tempo continuo, se si impone la variazione nulla di P quindi $\dot{P} = 0$ si ottiene la Continous Algebraic Riccati Equation (CARE), dove le matrici A, B, Q dipendono dallo stato x. L'equazione di Riccati deve essere risolta ad ogni istante, da cui il nome "frozen in time", a tempo discreto si traduce in una ricerca della P ad ogni passo di integrazione. Per simulare il comportamento a tempo continuo si può usare uno dei metodi già implementati da matlab che calcola la matrice P, ma data la dipendenza dallo stato sarà necessario ricalcolarla ad ogni variazione dello stato. Per simulare ed implementare su dispositivi reali a tempo discreto il metodo, si è utilizzata la forma più semplice: un metodo ricorsivo con un numero di iterazioni sufficiente a raggiungere una matrice P sufficientemente costante usando l'equazione Discrete Algebraic Riccati Equation (DARE) dipendente dallo stato mediante SDC, come illustrato in letteratura [36].

$$\begin{aligned} P &= Q + A_d^TPA_d - A_d^TPB_d(R + B_d^TPB_d)^{-1}B_d^TPA_d \\ K &= (R + B_d^TPB_d)^{-1}B_d^TPA_d \end{aligned} \tag{4.7}$$

4.2. La matrice Q

In letteratura, la scelta delle matrici di peso Q ed R è un problema aperto anche per il caso lineare. In questo scenario particolare, si introduce la dipendenza dallo stato, alcuni lavori [3] hanno ipotizzato un metodo empirico per la scelta dei termini della diagonale di Q. In questa tesi è proposto un metodo di scelta legato alla dinamica del sistema, tenendo conto anche dei coefficienti extra diagonale. Di seguito la matrice è stata inserita nel processo di costruzione della equazione ARE. Sono inoltre mostrate alcune prove empiriche della

sua efficacia. Sia la matrice Q come il gramiano inverso della dinamica dipendente dallo stato $Q = (A^T A)^{-1}$. Questa scelta soddisfa il vincolo di simmetria e definita positività di Q in quanto la matrice gramiana ha per sua proprietà intrinseca tali caratteristiche, che sono conservate dalla sua inversa. Se si considera l'esempio di Erdem (3.6), utilizzando tale matrice Q, si può notare un risultato molto interessante già analizzando gli autovalori. Si risolve l'equazione di Riccati "congelata nel tempo" nel punto iniziale 2,2, si può calcolare P e di conseguenza K, da cui la dinamica a ciclo chiuso e gli autovalori.

$$Q = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} \quad P = \begin{bmatrix} 2.6372 & 1.0869 \\ 1.0869 & 1.9263 \end{bmatrix} \quad \Lambda = \begin{bmatrix} -4.8526 + 1.5962i \\ -4.8526 - 1.5962i \end{bmatrix} \tag{4.8}$$

$$Q_a = (A^T A)^{-1} \quad P = \begin{bmatrix} 2.7066 & 1.7020 \\ 1.7020 & 1.6778 \end{bmatrix} \quad \Lambda = \begin{bmatrix} -4.3556 + 3.9964i \\ -4.3556 - 3.9964i \end{bmatrix} \tag{4.9}$$

$$u_o = -2x_1x_2 \quad \Lambda = \begin{bmatrix} -5.0000 + 4.0000i \\ -5.0000 - 4.0000i \end{bmatrix} \tag{4.10}$$

Confrontando gli autovalori Λ generati dalla scelta di una matrice Q diagonale 10,10, la matrice proposta Q_a e quelli ottenuti dall'ingresso ottimale, si può notare come la prima sia più vicina per la parte reale ma non immaginaria mentre la seconda è molto vicina per la parte immaginaria ma leggermente più lontana per quella reale. Confrontando anche le matrici P ottenute dalle due Q, sembra che la discrepanza sulla parte immaginaria sia causata dai termini extra diagonale. Apparentemente il beneficio di questa matrice Q_a sembra marginale, ma nel confronto delle traiettorie simulate il controllo più veloce è quello con tale matrice. In figura, le simulazioni sono confrontate per un controllo con u ottimale in giallo, con Q_a e A_2 in rosso, con Q_{diag} e A_2 in blu e con Q_{diag} A_3 in giallo. Il vantaggio è evidente se si considera uno stato iniziale diverso, specialmente se più lontano dalla zona di linearità.

La traiettoria con punto iniziale 5,5 dei diversi controlli mantiene il comportamento precedentemente analizzato, la matrice Q_a resta la più efficiente. Analizzando gli autovalori e confrontandoli si nota come la matrice proposta si adatta più rapidamente con un valore di -39 contro il -26 del controllo ottimo in 2,2. Questo suggerisce un'efficienza costante anche al variare delle condizioni iniziali, una caratteristica fondamentale in capo non lineare. Per trarre conclusioni più esaustive occorre testare il metodo su sistemi più complessi, in quanto questo semplice caso, sebbene non lineare, è per sua natura stabile(per un tempo sufficientemente lungo, qualsiasi punto iniziale tenderà a zero), ciò

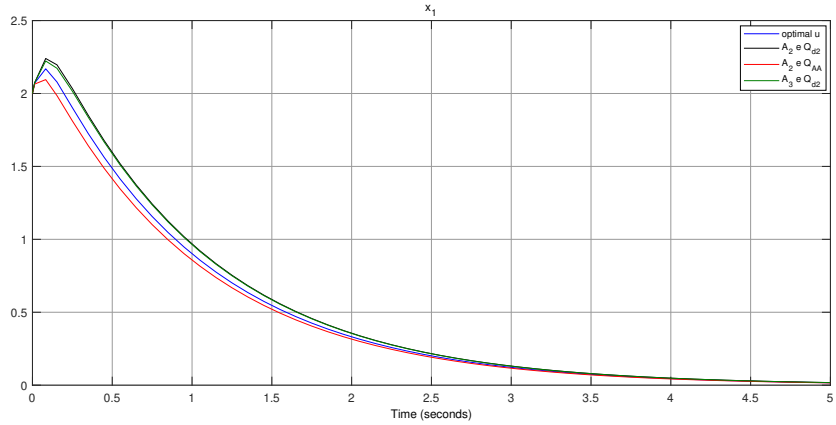


Figura 4.1.: Traiettoria di x_1 al variare della tecnica di controllo con stato iniziale (2,2).

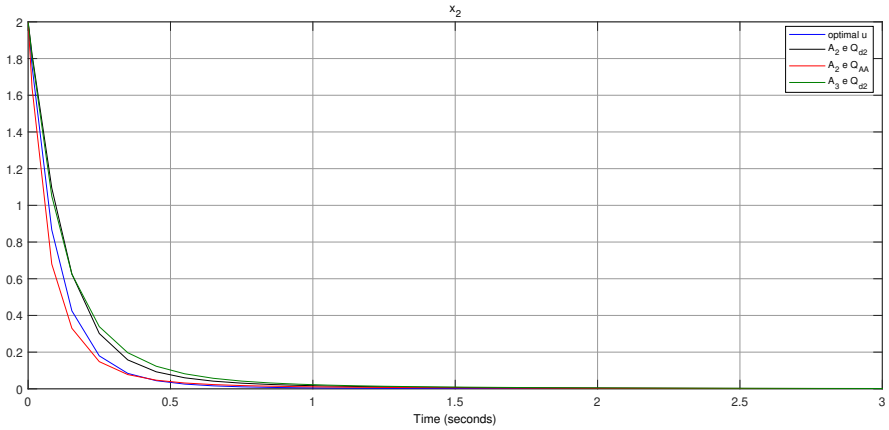


Figura 4.2.: Traiettoria di x_2 al variare della tecnica di controllo con stato iniziale (2,2).

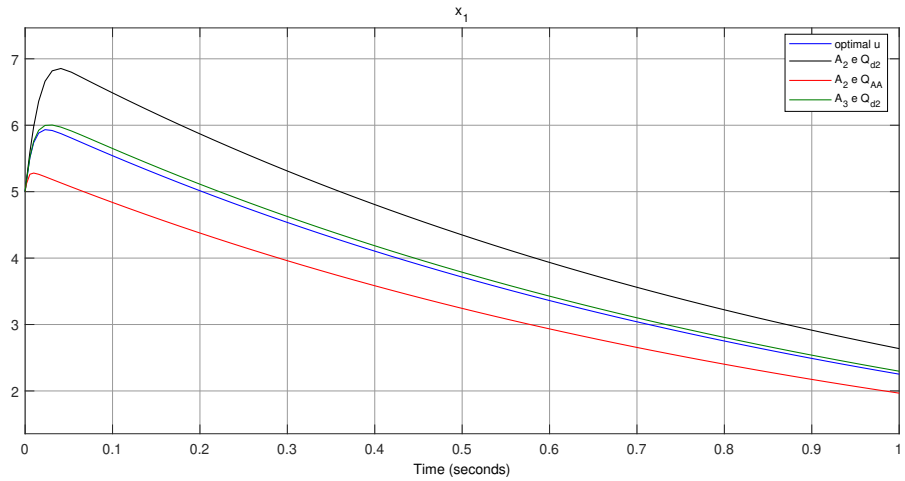


Figura 4.3.: Traiettoria di x_1 al variare della tecnica di controllo con stato iniziale (5,5).

impedisce di valutare a pieno le potenzialità ed i limiti del controllo.

$$Q = \begin{bmatrix} 10 & 0 \\ 0 & 10 \end{bmatrix} \quad P = \begin{bmatrix} 0.7531 & 0.5829 \\ 0.5829 & 1.2119 \end{bmatrix} \quad \Lambda = \begin{bmatrix} -16.1493 + 11.6104i \\ -16.1493 - 11.6104i \end{bmatrix} \tag{4.11}$$

$$Q = (A^T A)^{-1} \quad P = \begin{bmatrix} 14.5049 & 4.8867 \\ 4.8867 & 3.0929 \end{bmatrix} \quad \Lambda = \begin{bmatrix} -39.661 + 39.4905i \\ -39.661 - 39.4905i \end{bmatrix} \tag{4.12}$$

$$u_o = -2x_1x_2 \quad \Lambda = \begin{bmatrix} -26.0000 + 25.0000i \\ -26.0000 - 25.0000i \end{bmatrix} \tag{4.13}$$

4.3. Efficienza computazionale

Al fine di studiare l'efficienza di tale tecnica di controllo su un sistema reale, si è scelto uno dei classici sistemi di studio noti nella letteratura del controllo, quello del pendolo su carrello [37]. Il sistema è stato modellato e realizzato per testare la tecnica di controllo in un caso real-time [38] e confrontarlo con le principali tecniche di controllo lineare [39]. Il carrello è mosso longitudinalmente da due motori DC che controllano le ruote posteriori, un sensore sonar misura la distanza dal muro ed un encoder l'angolo di inclinazione del pendolo, in figura lo schema e la realizzazione 4.5. Il dispositivo è controllato da una scheda XMOS con quattro core dove il primo implementa la logica di controllo,

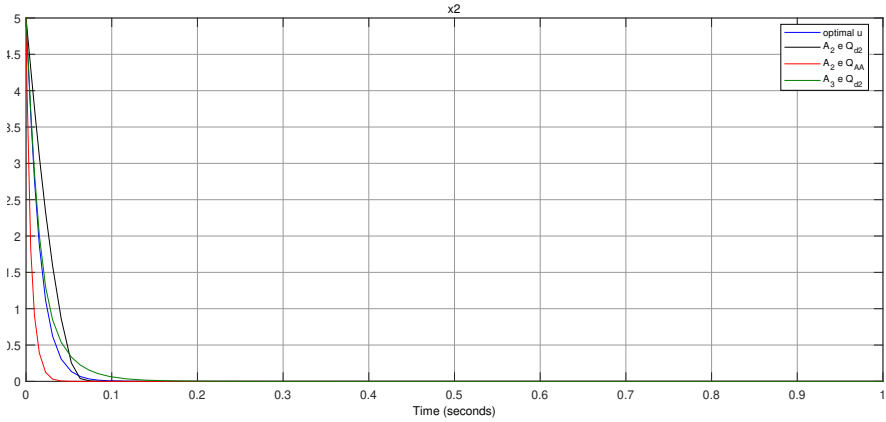
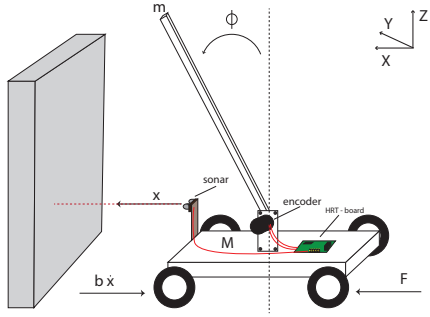
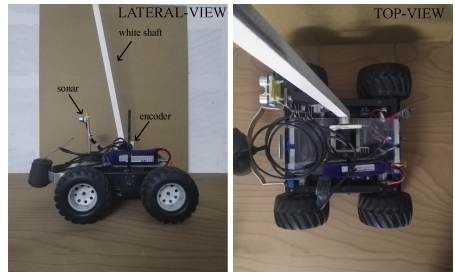


Figura 4.4.: Traiettoria di x_2 al variare della tecnica di controllo con stato iniziale (5,5).



(a)



(b)

Figura 4.5.: Pendolo su carrello: schema e realizzazione.

il secondo è dedicato all'encoder, il terzo al sonar ed il quarto pilota il driver che alimenta i motori mediante modulazione PWM. Ciascun microcontroller registra i tempi di calcolo attivo e comunica con gli altri mediante architettura a scambio messaggi. Le equazioni caratteristiche del dispositivo sono:

$$\begin{aligned} &(m+M)\ddot{x} + (b+\gamma)\dot{x} + m\,L\ddot{\phi}\cos\phi - m\,L\dot{\phi}^2\sin\phi = \beta\,E \\ &-m\,L\cos\phi\ddot{x} - (J_p + m\,L^2)\ddot{\phi} + m\,g\,L\sin\phi = 0 \end{aligned} \tag{4.14}$$

Al fine di migliorare il modello è stato necessario includere il ritardo del motore DC utilizzando l'equazione base identificando le costanti necessarie:

$$F = \frac{(J\,K_p)}{(\tau_p\,R_r)}E - \frac{J}{\tau_p\,R_r^2}\dot{x} \tag{4.15}$$

Tabella 4.1.: Parametri fisici del pendolo su carrello

Parametro	Valore	Descrizione
M[kg]	2.683	Massa del carrello
m[kg]	0.062	Massa del pendolo
L[m]	0.303	Metà lunghezza del pendolo
b[N/m sec]	0.15	Attrito ruote carrello
J_p[kg m^2]	0.0019874	Inerzia del pendolo centrata nel baricentro
g[m/sec^2]	9.8065	Accelerazione gravitazionale
F[N]		Forza longitudinale
x[m]		Spostamento longitudinale
ϕ[rad]		Angolo del pendolo con la verticale
R_r[m]	0.0605	Raggio ruota
M_R[kg]	0.127	Massa ruota
I_R[kg m^2]	2.28e(-4)	Inerzia ruota

Questa forza è inserita nel modello ((4.14)) considerando i termini $\beta = \frac{(J\,K_p)}{(\tau_p\,R_r)}$ e $\gamma = \frac{J}{\tau_p\,R_r^2}$.

Tabella 4.2.: Parametri fisici del motore HN-GH35GMA

Parametri	Valore	Descrizione
τ_{stall} [N m]	0.44	Coppia di stallo
I_{stall} [A]	1.4	Corrente di stallo
K_t [Nm/A]	$\frac{\tau_{stall}}{I_{stall}} = 0.3152$	Costante di coppia
K_e [V sec/rad]	0.4090	Costante forza controelettromotrice
R_a [Ω]	7.49	Resistenza di armatura
B_m [$N\,rad/\,s$]	0.007	Attrito viscoso
J [kg m^2]	0.0018	Inerzia rotore
E [V]		Tensione di alimentazione

Considerando s_3=sin (x_3), c_3=cos (x_3) e η_{nl}=$(J_p+mL^2)(M+m)-m^2L^2\cos^2(x_3)$, le matrici SDC equivalenti sono:

$$A_{c_{nl}}(\boldsymbol{x}_s) = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & 0 & 1 \\ 0 & a_{42} & a_{43} & a_{44} \end{bmatrix} \quad B_{c_{nl}}(\boldsymbol{x}_s) = \begin{bmatrix} 0 \\ b_{21} \\ 0 \\ b_{41} \end{bmatrix} \qquad (4.16)$$

$$C_{c_{nl}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$a_{22} = -\frac{(b+\gamma)(J_p+mL^2)}{\eta_{nl}} \qquad a_{23} = -\frac{\sin(x_3)\cos(x_3)m^2L^2g}{x_3\eta_{nl}}$$

$$a_{24} = \frac{mL\sin(x_3)x_4(J_p+mL^2)}{\eta_{nl}} \qquad a_{42} = \frac{mL(b+\gamma)\cos(x_3)}{\eta_{nl}}$$

$$a_{43} = \frac{mgL\sin(x_3)(M+m)}{x_3\eta_{nl}} \qquad a_{44} = -\frac{m^2L^2\sin(x_3)\cos(x_3)x_4}{\eta_{nl}}$$

$$b_{21} = \frac{J_p+mL^2}{\eta_{nl}}\,\beta \qquad b_{41} = -\frac{mL\cos(x_3)}{\eta_{nl}}\,\beta$$

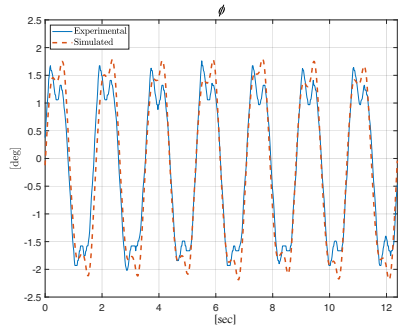


Figura 4.6.: Controllo SDRE confronto simulazione ed acquisizione reale: angolo del pendolo ϕ.

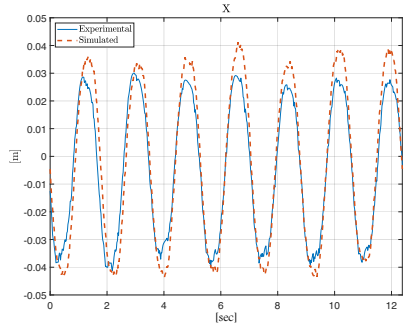


Figura 4.7.: Controllo SDRE confronto simulazione ed acquisizione reale: spostamento longitudinale x.

Il modello non lineare è stato confrontato con i dati acquisiti dal sistema reale con controllo SDRE. I grafici 4.6,4.8 e 4.7 mostrano un comportamento molto simile, la discrepanza maggiore è nel momento in cui il motore è alimentato con tensioni vicine a 0, la probabile causa è la Dead-Zone (circa 4V) molto presente di questi motori.

Le equazioni ((4.14)) sono state linearizzate attorno allo zero dell'angolo del pendolo $\phi \simeq 0$ per i controlli lineari in spazio di stato, quindi $cos(\phi) \simeq 1$, $sin(\phi) \simeq \phi$, $\dot{\phi}^2 \simeq 0$. Le equazioni risultanti sono:

$$\begin{aligned} (m+M)\ddot{x} + (b+\gamma)\dot{x} + m\,L\ddot{\phi} &= \beta\,E \\ -\,m\,L\ddot{x} - (J_p + m\,L^2)\ddot{\phi} + m\,g\,L\phi &= 0 \end{aligned} \tag{4.17}$$

Discretizzando il sistema per l'implementazione con il metodo di Eulero, dove

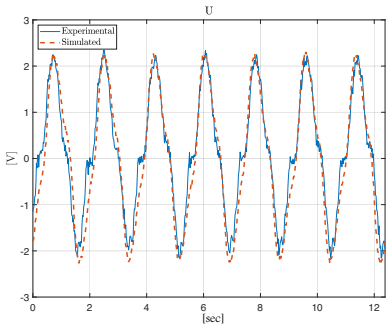


Figura 4.8.: Controllo SDRE confronto simulazione ed acquisizione reale: tensione motore u.

il tempo di campionamento ΔT, si ha:

$$\begin{aligned} \boldsymbol{x}_s[k+1] &= A_d\,\boldsymbol{x}_s[k] + B_d\,u[k] \\ y[k] &= C\boldsymbol{x}[k] \end{aligned} \tag{4.18}$$

Dove $u[k] = E[k]$, le matrici sono:

$$\begin{aligned} A_d &= \begin{bmatrix} 1, & \Delta T & 0, & 0 \\ 0, & 1+\Delta T a_{22}, & \Delta T a_{23}, & 0 \\ 0, & 0, & 1, & \Delta T \\ 0, & \Delta T a_{42}, & \Delta T a_{43}, & 1 \end{bmatrix} \\ B_d &= \begin{bmatrix} 0 \\ \Delta T b_{21} \\ 0 \\ \Delta T b_{41} \end{bmatrix} \quad C_d = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \end{aligned} \tag{4.19}$$

Utilizzando il sistema lineare e discreto è stato implementato il controllo LQR ed MPC, questo è stato confrontato con il controllo SDC basato sul sistema non lineare discretizzato. In appendice A sono riportate le implementazioni in codice C++ in ambiente XMOS. Di seguito i grafici 4.9, 4.10 e 4.11 delle acquisizioni dei tre sistemi di controllo per confrontare l'efficacia, partendo da uno stato iniziale di 6° di inclinazione del pendolo e 0 cm dal punto di riferimento.

I controllo lineari hanno un limite dovuto appunto alla loro linearizzazione, con prove successive per questo sistema, superata la soglia degli 10° l'unico controllo ancora funzionante è l'SDRE come mostrato in figura 4.12.

Una modifica dell'algoritmo di controllo può favorire l'efficienza nel tempo medio senza inficiare troppo l'efficacia, nei grafici 4.14, 4.15 e 4.16 il confronto

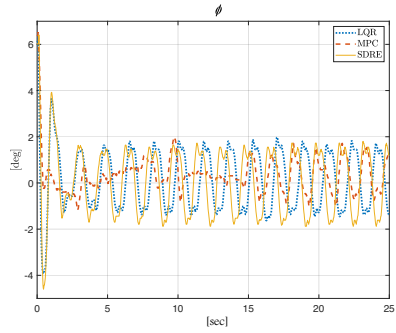


Figura 4.9.: Acquisizioni del pendolo su carrello per i tre controlli (LQR, MPC, SDRE): angolo d'inclinazione pendolo ϕ.

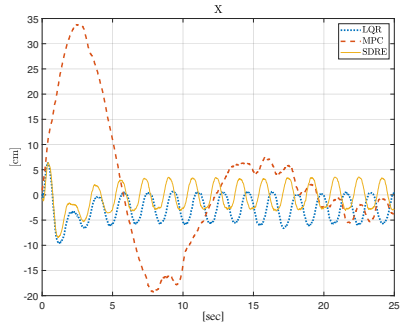


Figura 4.10.: Acquisizioni del pendolo su carrello per i tre controlli (LQR, MPC, SDRE): spostamento longitudinale x.

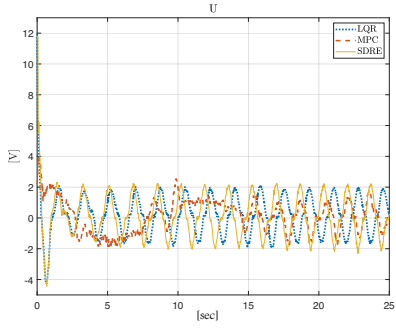


Figura 4.11.: Acquisizioni del pendolo su carrello per i tre controlli (LQR, MPC, SDRE): tensione motore U.

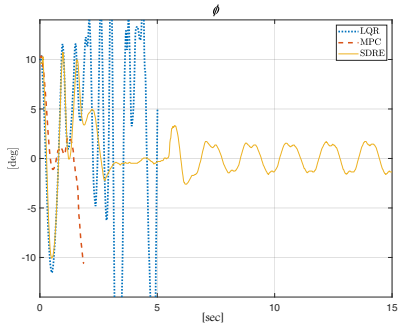


Figura 4.12.: Acquisizioni del pendolo su carrello per i tre controlli (LQR, MPC, SDRE): angolo d'inclinazione ϕ.

Tabella 4.3.: Tempi di esecuzione task caso Migliore e Peggiore

Task	Migliore	Peggiore
Encoder	0.00331 ms	0.00331 ms
Driver motore	1.50025 ms	2.00025 ms
Sonar	1.58035 ms	6.57300 ms
LQR	1.90454 ms	4.01384 ms
MPC	2.06109 ms	4.55033 ms
SDRE	11.9710 ms	12.5170 ms
MET-SDRE	3.25336 ms	8.15412 ms

fra le tecniche. Se consideriamo che il controllo SDRE in zona lineare è equivalente al LQR, è possibile evitare di ripetere la ricerca della soluzione ad ogni iterazione come richiesto dall'algoritmo. Quando il sistema esce dalla zona di linearità, il controllo riprenderà la ricerca della soluzione SDRE. Tale approccio proposto è nominato come Minimal Execution Time SDRE (MET-SDRE), il diagramma di flusso è in figura 4.13 dove $\phi_m = -4°$ e $\phi_M = 4°$.

Ogni acquisizione ha registrato i tempi per svolgere i task di acquisizione dei sensori e di controllo per le varie tecniche, calcolando i tempi medi migliori e peggiori è possibile confrontare l'efficienza computazionale delle tecniche a parità di prestazione (stato iniziale $\phi_0 = 6°$). Di seguito riassunti in tabella 4.3 i risultati. Si può notare come una piccola miglioria del sistema SDRE porta a prestazioni paragonabili con il controllo lineare conservando la possibilità di funzionare anche al di fuori della zona di linearizzazione.

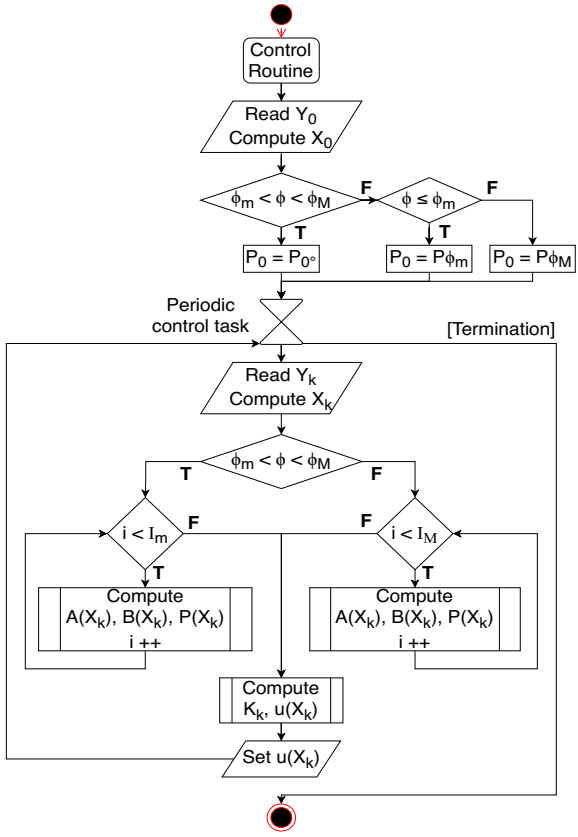


Figura 4.13.: Diagramma di flusso dell'algoritmo MET-SDRE. In figura T ed F rappresentano True e False.

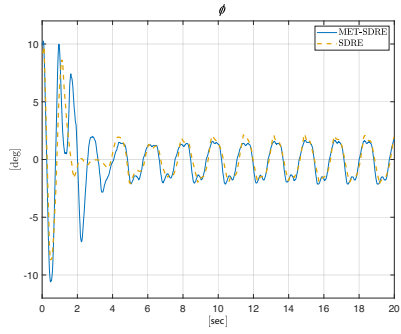


Figura 4.14.: Acquisizioni del pendolo su carrello per i due controlli (SDRE e MET-SDRE): angolo di inclinazione ϕ.

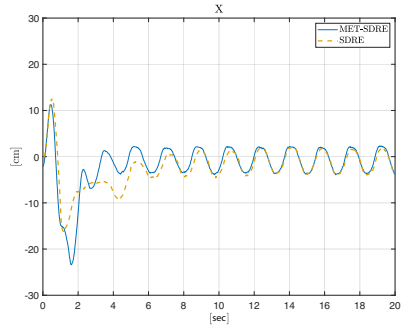


Figura 4.15.: Acquisizioni del pendolo su carrello per i due controlli (SDRE e MET-SDRE): spostamento longitudinale x.

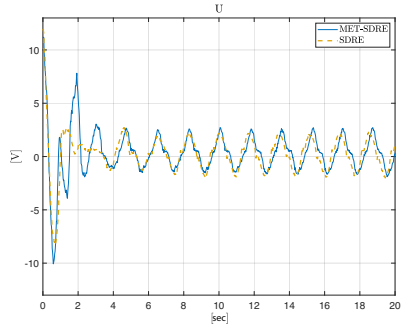


Figura 4.16.: Acquisizioni del pendolo su carrello per i due controlli (SDRE e MET-SDRE): tensione motore U.

Capitolo 5.

Caso di studio industriale: filtro Kalman SDC

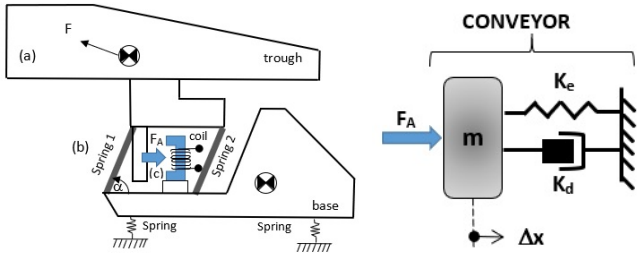
5.1. Sistema industriale di imbustamento

Per valutare l'efficacia del metodo in campo industriale sono stati considerati due componenti di una macchina per l'imbustamento di alimenti, in collaborazione con l'azienda *Miele S.p.A.*. Il piatto vibrante che alimenta il cestello di pesatura ed il sistema di pesatura stesso. Su questi dispositivi si è implementato un osservatore di kalman dello stato ed è proposta una forma SDC per le matrici che lo caratterizzano, sulla base di lavori che in letteratura hanno studiato l'osservabilità SDC [40] [41]. Nel caso del piatto vibrante l'osservatore deve stimare la posizione del sistema dalla misura dell'accelerazione misurata da un sensore inerziale, il controllo utilizza il metodo bang-off. Questo sistema si è mostrato troppo semplice per beneficiare nell'uso di un filtro di kalman SDC, la sua zona di lavoro è molto vicina a quella lineare anche considerando la variazione di peso indotta dal carico che sposta nel piatto. Mentre per la pesatura, lo stimatore SDC è utilizzabile per calcolare il peso del carico basandosi sulla misura di compressione della cella di carico, apparentemente si stima la misura stessa essendo le due correlate da una costante, ma l'obbiettivo è elidere la dinamica del transitorio nel più breve tempo possibile.

5.1.1. Piatto vibrante

Il piatto vibrante è rappresentato in figura 5.2, costituito da due copri, il primo è il piatto che trasporta il materiale da impacchettare connesso alla base da due placche che agiscono come molle, la base è a sua volata attaccata al terreno da tre molle con lo scopo di ammortizzare le vibrazioni della base a bassa frequenza, foto 5.3d. L'attuatore è una elettrocalamita posta fra il piatto e la base. Trascurando l'azione delle molle fra base e terreno, il sistema può essere modellato con la classica forma massa-molla-smorzatore.

Figura 5.1.: Macchina di pesatura multitesta.



(a) (b)

Figura 5.2.: Schema e modello del piatto vibrante.

I parametri fisici misurati sono riassunti nella tabella 5.1. L'equazione può essere riscritta in funzione della pulsazione di risonanza ω_0 ed il fattore di smorzamento ζ, come noto in letteratura [42], di seguito riportata:

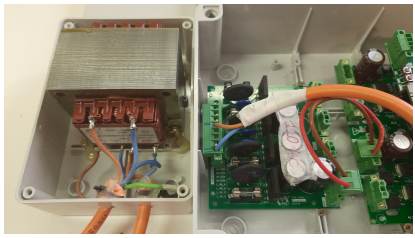
$$\ddot{x} + 2\zeta\omega_0\dot{x} + \omega_0^2 x = K_p\omega_0^2 u \tag{5.1}$$

Dove x è lo spostamento, K_p la costante di potenza che esprime la forza elettromagnetica in funzione della tensione di alimentazione e pulsazione di risonanza e u è la tensione di alimentazione del dispositivo con valore ammesso 0 o 160 V. In spazio di stato lineare l'equazione e rappresentata dalle matrici:

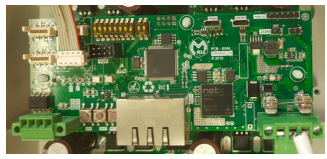
$$\dot{X} = AX + Bu = \begin{bmatrix} 0 & 1 \\ -\omega_o^2 & -2\zeta\omega_0 \end{bmatrix} X + \begin{bmatrix} 0 \\ K_p\omega_0^2 \end{bmatrix} u \tag{5.2}$$

$$y = CX = \begin{bmatrix} 0 & 1 \end{bmatrix} X \tag{5.3}$$

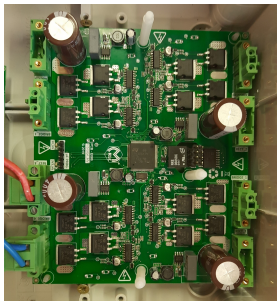
Si è scelto di implementare un controllo bang-off che necessita di un osservatore dello stato per ottenere spostamento e velocità del piatto vibrante.



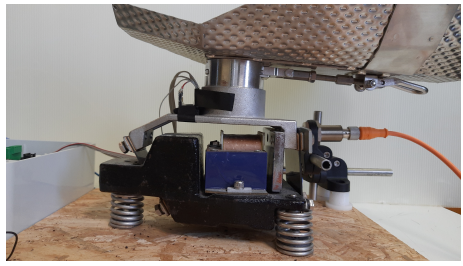
(a) *Alimentazione, da sinistra a destra: convertitore 220V a 110V AC e ponte-H a 160V DC*



(b) *Scheda Microcontroller*



(c) *Scheda pilota ponte-H DC*



(d) *L'accelerometro è attaccato sul piatto con nastro isolante, il sensore di distanza è montato a destra*

Figura 5.3.: *Set sperimentale.*

Tabella 5.1.: Parametri fisici del sistema

Variabile	Descrizione	Valore
f_0	frequenza di risonanza	52.6 Hz
m	massa del piatto	1.164 Kg
A_r	spostamento massimo	$1.24 \cdot 10^{-3}$ m
K_e	rigidezza molla	129990 N/m
K_d	smorzamento molla	4.7333 $N \cdot (s/m)$
U	tensione di alimentazione	160 V
K_p	costante di potenza	$1.5364 \cdot 10^{-6}$ $N/(V\,(rad/s)^2)$
ζ	fattore di smorzamento	0.0061
ω_0	frequenza di risonanza	334.1766 rad/s

Sia $x_{1,ref}$ lo spostamento desiderato o ampiezza dell'oscillazione, di seguito l'equazione che regola il controllore:

$$\begin{aligned} x_{1,ref} &= 1.14e-3 \\ z &= \omega_0^2 x_1^2 + x_2^2 - (x_{1,ref}\,\omega_0)^2 \end{aligned} \tag{5.4}$$

$$\begin{cases} |z| \leq \epsilon \Rightarrow u = 0 \qquad \epsilon = 0.01 \\ |x_2| \leq 0 \Rightarrow u = 160 \\ z > 0, \quad x_2 \geq 0 \Rightarrow u = 160 \\ else \quad u = 0 \end{cases} \tag{5.5}$$

Il sensore inerziale montato sul piatto vibrante, fig. 5.3d, misura l'accelerazione, questa è filtrata mediante un filtro IIR allo scopo di eliminare il tipico fenomeno di deriva di questa categoria di sensori. Questo prefiltraggio non inquina il risultato poiché lo scopo del sistema è oscillare attorno allo zero ad una frequenza nota, quella di risonanza. Il filtro IIR è un passa-banda fra 10 e 200 Hz, di 4th ordine con frequenza di campionamento 1000 Hz, nel grafico 5.4 sono riportate le caratteristiche di guadagno e fase. Di seguito (5.6) le equazioni del filtro in spazio di stato dove u_f è l'accelerazione misurata e y_f è l'accelerazione filtrata.

$$\begin{aligned} \dot{x}_{1,f} &= 0.436597u_f + 1.91155x_{1,f} - 0.915653x_{2,f} \\ \dot{x}_{2,f} &= x_{1,f} \\ \dot{x}_{3,f} &= 0.190617u_f + 0.834577x_{1,f} - 0.836368x_{2,f} + 0.423531x_{3,f} - 0.22571x_{4,f} \\ \dot{x}_{4,f} &= x_{3,f} \\ y_f &= 0.190617u_f + 0.834577x_{1,f} - 0.836368x_{2,f} + 0.423531x_{3,f} - 1.22571x_{4,f} \end{aligned} \tag{5.6}$$

In Appendice B il codice che implementa il filtraggio e controllo. La misura di accelerazione filtrata è integrata per calcolare la velocità e tale stima è l'ingresso del filtro di kalman che fornisce una migliore stima dello stato costituito da posizione e velocità del piatto vibrante. Di seguito le equazioni del filtro di kalman discreto:

$$\begin{aligned} &\hat{x}(k|k-1) = A_d \cdot \hat{x}(k-1) + B_d \cdot u(k-1) \\ &P(k)' = A_d \cdot P(k-1) \cdot A_d^T + Q_w \\ &e(k) = y(k) - C_d \cdot \hat{x}(k|k-1) \\ &K(k) = P(k)' \cdot C_d^T \cdot (C_d \cdot P(k)' \cdot C_d^T + R_v)^{-1} \\ &\hat{x}(k) = \hat{x}(k|k-1) + K(k) \cdot e(k) \\ &P(k) = P(k)' - K(k) \cdot C_d \cdot P(k)' \end{aligned} \tag{5.7}$$

$$Q_w = \begin{bmatrix} 10^{-4} & 0 \\ 0 & 10^{-4} \end{bmatrix} \quad R_w = 1$$

Dove Q_w e R_w sono rispettivamente le matrici di covarianza del rumore di processo e misura, mentre k è l'istante discreto. Allo scopo di valutare l'efficacia

di questo metodo di misura e stima, il set sperimentale è fornito di un sensore di distanza magnetico posto frontalmente al vibratore (fig. 5.3d, cavo arancione), tale segnale è acquisito separatamente dal sistema e confrontato con la stima nei grafici 5.5 e 5.6. Questo studio è in corso di pubblicazione [43] con un focus sull'ottimizzazione temporale del transitorio.

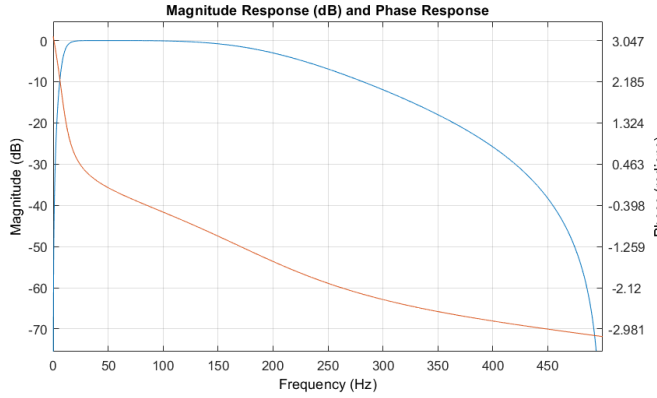


Figura 5.4.: Risposta in fase e guadagno del filtro IIR

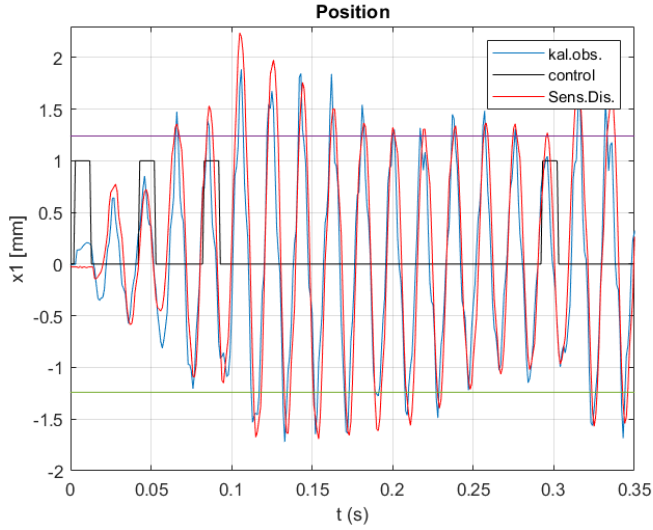


Figura 5.5.: Confronto fra spostamento misurato e stimato

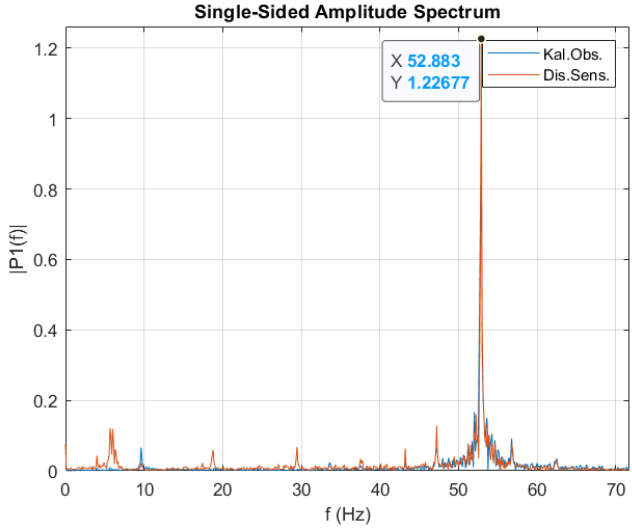


Figura 5.6.: Confronto fra frequenza misurata e stimata

5.1.2. Bilancia

Il sistema di pesatura e costituito da una cella di carico agganciata ad un cestello fig.5.7, un secondo cestello sovrastante lo alimenta con piccole quantità del cibo da pesare, generalmente granulato o simili. Il cestello della cella di carico viene aperto da un sistema terzo che richiede la misura di peso il prima possibile. Tale dispositivo può essere anch'esso modellato con il modello massa-molla-smorzatore. L'equazione differenziale risultante è la seguente:

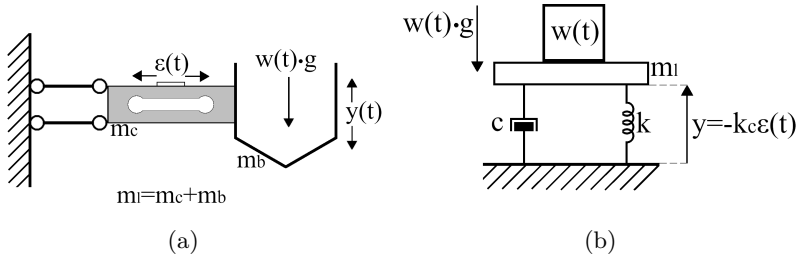


Figura 5.7.: Modello e schema della cella di carico

L'equazione è scritta come segue:

$$[w(t) + M]\left[\ddot{y}(t) + \ddot{y}^{\perp}(t)\right] + c\dot{y}(t) + ky(t) = w(t)g \tag{5.8}$$

Dove:

- w(t) - massa del prodotto
- M - massa del cestello ($M = M_l + M_b$, con M_l = massa della cella, M_b = massa del cestello)
- c - coefficiente di smorzamento
- k - rigidezza della molla
- $y(t)$ - spostamento del cestello
- $y^{\perp}(t)$ - disturbo ambientale sullo spostamento
- $\varepsilon(t)$ - deformazione misurata dalla strain gauge
- K_c - guadagno della strain gauge

Riordinando (5.8) ed omettendo la dipendenza dal tempo si ottiene:

$$\ddot{y} + \frac{c}{[w(t)+M]}\dot{y} + \frac{k}{[w(t)+M]}y = \frac{w(t)g}{[w(t)+M]} - \ddot{y}^{\perp} \tag{5.9}$$

- $M = 2.022kg$
- $c = 34.5714Ns/m$
- $k = 9068.8N/m$

In spazio di stato il sistema SDC, che dipende dal peso del prodotto stesso w, diventa:

$$\begin{aligned} A &= \begin{bmatrix} 0 & 1 \\ \frac{-9068.8}{w+2.022} & \frac{-34.5714}{w+2.022} \end{bmatrix} \quad B = \begin{bmatrix} 0 \\ \frac{9.81}{w+2.022} \end{bmatrix} \\ C &= \begin{bmatrix} \frac{9068.8}{9.81} & 0 \end{bmatrix} \end{aligned} \tag{5.10}$$

In questo caso la necessità è di ridurre i tempi di misura al minimo. A tale scopo sono state utilizzate due tecniche. La prima stima il valore del peso come punto medio fra coppie di minimo e massimo della traiettoria della misura. Calcolando la derivata prima istantanea, si può stimare quando si presenta un punto di minimo o massimo, fra questi si utilizza la media come stima del peso. Ciò è possibile poiché il punto di flesso tende al valore finale del sistema, smorzata completamente l'oscillazione, più rapidamente della traiettoria dello spostamento. In appendice C il codice che implementa il metodo. Questo metodo è in fase di pubblicazione [44]. Il secondo metodo utilizza lo stimatore di kalman SDC con un sistema ampliato. Questa rappresentazione alternativa

non ha ingresso, ma il peso esterno diventa la terza variabile di stato. Il nuovo sistema in spazio di stato è composto dalle seguenti matrici:

$$\begin{aligned} A &= \begin{bmatrix} 0 & 1 & 0 \\ \frac{-9068.8}{w+2.022} & \frac{-34.5714}{w+2.022} & \frac{9.81}{w+2.022} \\ 0 & 0 & 0 \end{bmatrix} \quad B = 0 \\ C &= \begin{bmatrix} \frac{9068.8}{9.81} & 0 & 0 \end{bmatrix} \end{aligned} \tag{5.11}$$

La riga nulla nella matrice della dinamica costituisce un problema poiché non invertibile, di conseguenza per utilizzare la matrice Q proposta, occorre una forma alternativa. Si considera la sottomatrice superiore 2x2 di Q prendendo la sottomatrice superiore corrispondente della matrice A, si completa la terza riga e colonna con una semplice costante sulla diagonale come segue:

$$\begin{aligned} A_s &= \begin{bmatrix} 0 & 1 \\ \frac{-9068.8}{w+2.022} & \frac{-34.5714}{w+2.022} \end{bmatrix} \quad Q_s = (A_s^T A_s)^{-1} \\ Q_a &= \begin{bmatrix} Q_s & \begin{bmatrix} 0 \\ 0 \end{bmatrix} \\ \begin{bmatrix} 0 & 0 \end{bmatrix} & 1 \end{bmatrix} \\ Q_d &= \begin{bmatrix} 10^{-5} & 0 & 0 \\ 0 & 10^{-2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned} \tag{5.12}$$

Questo metodo è senza dubbio più efficace, utilizzando dati acquisiti dal set sperimentale, si può constatare come la stima del peso sia ottimale. Nei grafici 5.8 con un campione di riso di 50 grammi e 5.9 con un campione di 10 grammi, sono confrontati: il segnale acquisito e filtrato con FIR 2° ordine in nero; la stima calcolata sul precedente segnale mediante metodo punto medio fra minimo e massimo in rosso; la stima del peso mediante filtro di kalman SDC con matrice Q_a in blu mentre in viola con matrice Q_d. Non solo il tempo per rientrare nella banda di tolleranza del 4% (in verde) è minimo, ma la sovraelungazione della misura, dovuta all'energia cinetica dell'impatto dal cestello superiore, è stata completamente eliminata. La scelta della matrice Q è determinante a parità della rappresentazione SDC, da questo caso applicativo, la scelta della giusta matrice di peso può portare alla stima ottima semplificando la necessità di una rappresentazione SDC di A tale da portare all'ottimo indipendentemente dalla matrice Q.

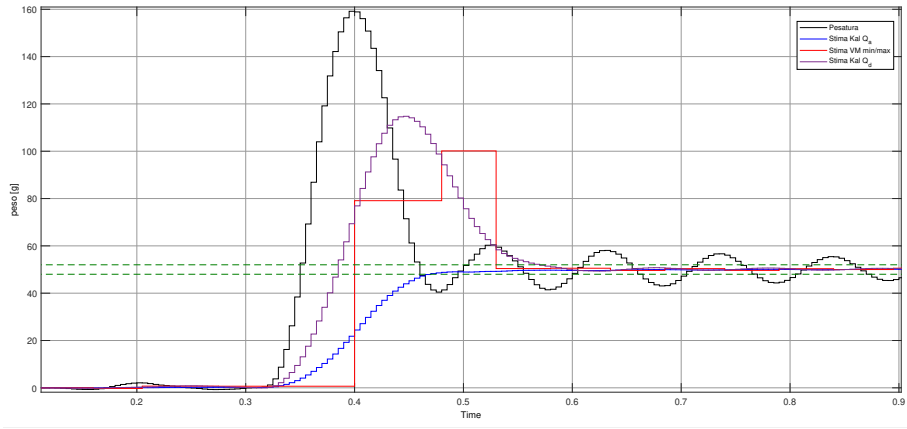


Figura 5.8.: Filtraggio di acquisizione campione di riso 50g

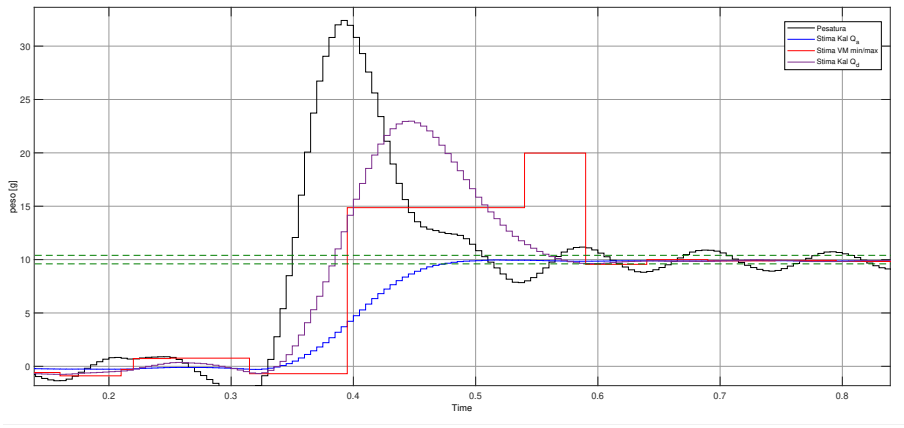


Figura 5.9.: Filtraggio di acquisizione campione di riso 10g

Capitolo 6.

Conclusioni

La tecnica di parametrizzazione SDC presenta vantaggi molto interessanti al netto di svantaggi che oggi possono essere resi meno limitanti. Una notevole duttilità nel rappresentare sistemi non lineari senza perdita di informazione in spazio di stato, come mostrato nel capitolo 3. La possibilità di implementare un controllo robusto, come mostrato nel capitolo 4, sebbene sia difficile valutarne l'efficacia, la soluzione ottima è raggiungibile ma non è presente un metodo semplice ed univoco per ottenerla. Nonostante l'oneroso carico computazionale, esistono molti modi per ridurre il costo senza inficiare il controllo, per esempio la soluzione proposta nel sistema pendolo su carrello 4.3. Nel momento in cui lo stato subisce piccole variazioni, il controllo è assimilabile alla sua forma lineare, il grosso vantaggio di questa tecnica è la possibilità di usare il medesimo controllo per punti di equilibrio differenti, come mostrato nell'esempio del doppio pendolo, la parametrizzazione SDC permette al controllo di stabilizzarsi nel nuovo stato. Sebbene la maggioranza dei sistemi reali ha caratteristiche non lineari, spesso in industria si preferisce un approccio conservativo, linearizzando il sistema o utilizzando tecniche semplici dove permesso anche al costo di una minore efficacia. Tuttavia, è possibile implementare tecniche SDC anche su sistemi industriali guadagnando migliori prestazioni senza modifiche hardware, come visto in 5.1.2. Nello studio di questa tecnica, sono stati catalogati i metodi per trattare le non linearità note in letteratura migliorandone alcune caratteristiche e proponendo una variante, in 3.2. Il punto critico resta trovare la parametrizzazione ottimale, anche analizzando le caratteristiche tipiche di un sistema in spazio di stato è ancora assente un legame matematico certo con le caratteristiche di $A(x)$ ed il controllo. La rappresentazione della raggiungibilità dipendente dalla matrice $A(x)$ proposta in 3.3 traccia un percorso per sviluppare un metodo di scelta e valutazione della matrice parametrica. La forma dipendente dallo stato ha un impatto sulla complessità di implementazione del controllo, come mostrato in 4.1, tuttavia è possibile trascurare gli effetti delle piccole variazioni (annullando le derivate di matrici) ed utilizzare la formula nota nel caso lineare risolvendola in ogni istante. Sebbene la scelta della matrice della dinamica abbia un impatto sul controllo, come mostrato

con un semplice esempio, e nonostante l'utilizzo di una formulazione approssimata, una scelta della matrice di peso dello stato Q può essere determinante per trovare la soluzione di controllo ottima. La forma proposta in 4.2 sembra permettere un controllo ottimo fintanto che la matrice della dinamica $A(x)$ soddisfa le condizioni locali imposte dal controllo, come invertibilità e raggiungibilità. Sviluppi futuri si devono concentrare sul colmare la carenza di metodi di scelta di $A(x)$, valutare l'effetto sul controllo nel trascurare le variazioni delle matrici dipendenti dallo stato, verificare se le considerazioni ottenute sulla raggiungibilità possono essere estese anche al caso della osservabilità. In ultimo, occorre approfondire se l'efficacia della matrice Q proposta possa essere verificata per ogni sistema rappresentabile in spazio di stato mediante struttura State-Dependent Coefficient.

Appendice A.

Implementazione Pendolo su carrello

In questa sezione vengono riportate le parti fondamentali del codice che implementa il MET-SDRE sul dispositivo di test pendolo su carrello. Il primo estratto è l'inizializzazione delle matrici e l'impostazione dei task paralleli.

```
    short Ad_row = 5, Ad_col = 5;
    float Ad[] =   {1.0,            0.02,              0.0,   0.0,
0.0,
                    0.0,   0.9992816048, -0.004650865606,   0.0, -0.01158468366,
                    0.0,            0.0,              1.0, 0.02,
0.0,
                    0.0, 0.002324822555,    0.6497542728,   1.0,  0.03748957961,
                    0.0,            0.0,              0.0,   0.0,     0.72899729};

    short Bd_row = 5, Bd_col = 1;
    float Bd[] = {     0.0,
            0.02023728389,
                      0.0,
           -0.06549054661,
             0.4734146341};

    short Cd_row = 2, Cd_col = 5;
    float Cd[] = {1.0, 0.0, 0.0, 0.0, 0.0,
                  0.0, 0.0, 1.0, 0.0, 0.0};

    // Controllo ottimo R && Q

    float R = 1;

    short Q_row = 5, Q_col = 5;
    float Q[] = { 1.0e3,    0.0,   0.0,    0.0,    0.0,
                    0.0,  1.0e0,   0.0,    0.0,    0.0,
                    0.0,    0.0,  5.0e3,   0.0,    0.0,
                    0.0,    0.0,   0.0,    1.0e2,  0.0,
                    0.0,    0.0,   0.0,    0.0,    1.0e2};

    short P_row = 5, P_col = 5;
    float P[] =   {1.0, 0.0, 0.0, 0.0, 0.0,
                   0.0, 1.0, 0.0, 0.0, 0.0,
                   0.0, 0.0, 1.0, 0.0, 0.0,
                   0.0, 0.0, 0.0, 1.0, 0.0,
                   0.0, 0.0, 0.0, 0.0, 1.0};

    short K_row = 1, K_col = 5;
```

```
float K[] = {0.0, 0.0, 0.0, 0.0, 0.0};

// Kalman covariance Rf && Qf

short Rf_row = 2, Rf_col = 2;
float Rf[] = {2.4655e-06,       0.0,
                      0.0,      6.4831e-09};

short Qf_row = 5, Qf_col = 5;
float Qf[] = {1e-6,  0.0,  0.0,  0.0,  0.0,
                0.0, 1e-6, 0.0,  0.0,  0.0,
                0.0, 0.0,  1e-6, 0.0,  0.0,
                0.0, 0.0,  0.0,  1e-6, 0.0,
                0.0, 0.0,  0.0,  0.0,  1e-6};

interface encoder_i enc_i;
interface motor_i mot_i;
interface sonar_i  son_i;

/*************************************************************
 *                                                           *
 *                          LQR                              *
 *                                                           *
 *     P = Q + Ad'*P*Ad - Ad'*P*Bd*inv(R+Bd'*P*Bd)*Bd'*P*Ad  *
 *                                                           *
 *************************************************************/
timer t;
int i=0;
unsigned int inizio=0, fine=0;
t :> inizio;
for (i=0; i<1000; i++) {
    riccatiEQ(Ad, Ad_row*Ad_col, Ad_row, Ad_col,
              Bd, Bd_row*Bd_col, Bd_row, Bd_col,
              Cd, Cd_row*Cd_col, Cd_row, Cd_col,
              Q,  Q_row*Q_col, Q_row, Q_col,
              P,  P_row*P_col, P_row, P_col, R);
}
t :> fine;
printf("tempo %.3fms\n",(float)(fine-inizio)/100000.0);

/*************************************************************
 *                                                           *
 *                          LQR                              *
 *                                                           *
 *                K = inv(R+Bd'*P*Bd)*Bd'*P*Ad               *
 *                                                           *
 *************************************************************/

gainK(Ad, Ad_row*Ad_col, Ad_row, Ad_col,
      Bd, Bd_row*Bd_col, Bd_row, Bd_col,
      P, P_row*P_col, P_row, P_col,
      K, K_row*K_col, K_row, K_col, R);

for (int i=0; i < K_col*K_row; i++){
          printf("%f ", K[i]);
      }
printf("\n");

par {
    encoder_server(enc_i, P_enc);
    sonar_server(son_i, P_sonar);
```

```
        motor_driver(mot_i, P_motor);
        controller(son_i, enc_i, mot_i,
                Ad, Ad_row*Ad_col, Ad_row, Ad_col,
                Bd, Bd_row*Bd_col, Bd_row, Bd_col,
                Cd, Cd_row*Cd_col, Cd_row, Cd_col,
                 Q, Q_row*Q_col, Q_row, Q_col,
                 P, P_row*P_col, P_row, P_col,
                Qf, Qf_row*Qf_col, Qf_row, Qf_col,
                Rf, Rf_row*Rf_col, Rf_row, Rf_col,
                K, K_row*K_col, K_row, K_col);
    }
```

Di seguito il codice che implementa il calcolo delle matrici SDC ed operazioni utili:

```
    /*
 * lqg.xc
 *
 *  Created on: 09/gen/2019
 *      Author: Omar Cocchiarella
 */
#include "lqg.h"
#include <stdio.h>


/* P = Q + Ad'*P*Ad - Ad'*P*Bd*inv(R+Bd'*P*Bd)*Bd'*P*Ad */
void riccatiEQ(float Ad[Ad_L], unsigned Ad_L, short Ad_row, short Ad_col,
        float Bd[Bd_L], unsigned Bd_L, short Bd_row, short Bd_col,
        float Cd[Cd_L], unsigned Cd_L, short Cd_row, short Cd_col,
        float Q[Q_L], unsigned Q_L, short Q_row, short Q_col,
        float P[P_L], unsigned P_L, short P_row, short P_col, float R){

    short P_Ad_row = 5, P_Ad_col = 5;
    float P_Ad[5*5];

    short BdT_row = 1, BdT_col = 5;
    float BdT[1*5];

    short BdT_P_Ad_row = 1, BdT_P_Ad_col = 5;
    float BdT_P_Ad[1*5];

    short P_Bd_row = 5, P_Bd_col = 1;
    float P_Bd[5*1];

    short BdT_P_Bd_row = 1, BdT_P_Bd_col = 1;
    float BdT_P_Bd[1*1];

    short SumR_row = 1, SumR_col = 1;
    float SumR[1*1];

    short AdT_row = 5, AdT_col = 5;
    float AdT[5*5];

    short AdT_P_Bd_row = 5, AdT_P_Bd_col = 1;
    float AdT_P_Bd[5*1];

    short AdT_P_Ad_row = 5, AdT_P_Ad_col = 5;
    float AdT_P_Ad[5*5];

    // Adt_P_Ad_sott_Adt_P_Bd___inv___Bdt_P_Ad
    short SumPar_row = 5, SumPar_col = 5;
    float SumPar[5*5];
```

```
// temp_inv__Bdt_P_Ad
short term1_row = 1, term1_col = 5;
float term1[1*5];

// temp_Adt_P_Bd__inv__Bdt_P_Ad
short term2_row = 5, term2_col = 5;
float term2[5*5];

// P*Ad => result @temp_P_Ad 4x4
matMul(P, P_row*P_col, P_row, P_col,
       Ad, Ad_row*Ad_col, Ad_row, Ad_col,
       P_Ad, P_Ad_row*P_Ad_col, P_Ad_row, P_Ad_col);

// Bd' => result @temp_Bdt 1x4
matTr(Bd, Bd_row*Bd_col, Bd_row, Bd_col,
      BdT, BdT_row*BdT_col, BdT_row, BdT_col);

// Bd'*P*Ad => result @temp_Bdt_P_Ad 1x4
matMul(BdT, BdT_row*BdT_col, BdT_row, BdT_col,
       P_Ad, P_Ad_row*P_Ad_col, P_Ad_row, P_Ad_col,
       BdT_P_Ad, BdT_P_Ad_row*BdT_P_Ad_col, BdT_P_Ad_row, BdT_P_Ad_col);

// P*Bd => result @temp_P_Bd 4x1
matMul(P, P_row*P_col, P_row, P_col,
       Bd, Bd_row*Bd_col, Bd_row, Bd_col,
       P_Bd, P_Bd_row*P_Bd_col, P_Bd_row, P_Bd_col);

// Bd'*P*Bd => result @temp_Bdt_P_Bd 1x1
matMul(BdT, BdT_row*BdT_col, BdT_row, BdT_col,
       P_Bd, P_Bd_row*P_Bd_col, P_Bd_row, P_Bd_col,
       BdT_P_Bd, BdT_P_Bd_row*BdT_P_Bd_col, BdT_P_Bd_row, BdT_P_Bd_col);

// R+Bd'*P*Bd => result @temp_Bdt_P_Bd__plus_R 1x1
SumR[0] = R + BdT_P_Bd[0];

// inv(R+Bd'*P*Bd) => result @temp_inv 1x1 (1x1 matrix -> inv = 1/matrix)
SumR[0] = 1/SumR[0];

// Ad' => result @temp_Adt 4x4
matTr(Ad, Ad_row*Ad_col, Ad_row, Ad_col,
      AdT, AdT_row*AdT_col, AdT_row, AdT_col);

// Ad'*P*Bd => result @temp_Adt_P_Bd 4x1
matMul(AdT, AdT_row*AdT_col, AdT_row, AdT_col,
       P_Bd, P_Bd_row*P_Bd_col, P_Bd_row, P_Bd_col,
       AdT_P_Bd, AdT_P_Bd_row*AdT_P_Bd_col, AdT_P_Bd_row, AdT_P_Bd_col);

// Ad'*P*Ad => result @temp_Adt_P_Ad 4x4
matMul(AdT, AdT_row*AdT_col, AdT_row, AdT_col,
       P_Ad, P_Ad_row*P_Ad_col, P_Ad_row, P_Ad_col,
       AdT_P_Ad, AdT_P_Ad_row*AdT_P_Ad_col, AdT_P_Ad_row, AdT_P_Ad_col);

// inv(R+Bd'*P*Bd) * Bd'*P*Ad => result @temp_inv__Bdt_P_Ad 1x4
matMul(SumR, SumR_row*SumR_col, SumR_row, SumR_col,
       BdT_P_Ad, BdT_P_Ad_row*BdT_P_Ad_col, BdT_P_Ad_row, BdT_P_Ad_col,
       term1, term1_row*term1_col, term1_row, term1_col);

// Ad'*P*Bd * inv(R+Bd'*P*Bd)*Bd'*P*Ad =>
// result @temp_Adt_P_Bd__inv__Bdt_P_Ad 4x4
matMul(AdT_P_Bd, AdT_P_Bd_row*AdT_P_Bd_col, AdT_P_Bd_row, AdT_P_Bd_col,
       term1, term1_row*term1_col, term1_row, term1_col,
```

```
            term2, term2_row*term2_col, term2_row, term2_col);

    // Ad'*P*Ad - Ad'*P*Bd*inv(R+Bd'*P*Bd)*Bd'*P*Ad =>
    // result @ temp_Adt_P_Ad_sott_Adt_P_Bd__inv__Bdt_P_Ad
    matSum(AdT_P_Ad, AdT_P_Ad_row*AdT_P_Ad_col, AdT_P_Ad_row, AdT_P_Ad_col,
            term2, term2_row*term2_col, term2_row, term2_col,
            SumPar, SumPar_row*SumPar_col, SumPar_row, SumPar_col, -1);

    // P = Q + Ad'*P*Ad - Ad'*P*Bd*inv(R+Bd'*P*Bd)*Bd'*P*Ad =>
    // result @P parameters on function call
    matSum(Q, Q_row*Q_col, Q_row, Q_col,
            SumPar, SumPar_row*SumPar_col, SumPar_row, SumPar_col,
            P, P_row*P_col, P_row, P_col, 1);

}

/* Feedback controller LQG
 *  K = inv(R+Bd'*P*Bd)*Bd'*P*Ad */
void gainK(float Ad[Ad_L], unsigned Ad_L, short Ad_row, short Ad_col,
        float Bd[Bd_L], unsigned Bd_L, short Bd_row, short Bd_col,
        float P[P_L], unsigned P_L, short P_row, short P_col,
        float K[K_L], unsigned K_L, short K_row, short K_col, float R){

        short P_Ad_row = 5, P_Ad_col = 5;
        float P_Ad[5*5];

        short BdT_row = 1, BdT_col = 5;
        float BdT[1*5];

        short BdT_P_Ad_row = 1, BdT_P_Ad_col = 5;
        float BdT_P_Ad[1*5];

        short P_Bd_row = 5, P_Bd_col = 1;
        float P_Bd[5*1];

        short BdT_P_Bd_row = 1, BdT_P_Bd_col = 1;
        float BdT_P_Bd[1*1];

        short SumR_row = 1, SumR_col = 1;
        float SumR[1*1];

    // P*Ad => result @temp_P_Ad 4x4
        matMul(P, P_row*P_col, P_row, P_col,
                Ad, Ad_row*Ad_col, Ad_row, Ad_col,
                P_Ad, P_Ad_row*P_Ad_col, P_Ad_row, P_Ad_col);

        // Bd' => result @temp_Bdt 1x4
        matTr(Bd, Bd_row*Bd_col, Bd_row, Bd_col,
               BdT, BdT_row*BdT_col, BdT_row, BdT_col);

        // Bd'*P*Ad => result @temp_Bdt_P_Ad 1x4
        matMul(BdT, BdT_row*BdT_col, BdT_row, BdT_col,
                P_Ad, P_Ad_row*P_Ad_col, P_Ad_row, P_Ad_col,
                BdT_P_Ad, BdT_P_Ad_row*BdT_P_Ad_col, BdT_P_Ad_row, BdT_P_Ad_col);

        // P*Bd => result @temp_P_Bd 4x1
        matMul(P, P_row*P_col, P_row, P_col,
                Bd, Bd_row*Bd_col, Bd_row, Bd_col,
                P_Bd, P_Bd_row*P_Bd_col, P_Bd_row, P_Bd_col);

        // Bd'*P*Bd => result @temp_Bdt_P_Bd 1x1
```

```
        matMul(BdT, BdT_row*BdT_col, BdT_row, BdT_col,
                P_Bd, P_Bd_row*P_Bd_col, P_Bd_row, P_Bd_col,
                BdT_P_Bd, BdT_P_Bd_row*BdT_P_Bd_col, BdT_P_Bd_row, BdT_P_Bd_col);

        // R+Bd'*P*Bd => result @temp_Bdt_P_Bd___plus_R 1x1
        SumR[0] = R + BdT_P_Bd[0];

        // inv(R+Bd'*P*Bd) => result @temp_inv 1x1 (1x1 matrix -> inv = 1/matrix)
        SumR[0] = 1/SumR[0];

    // K = inv(R+Bd'*P*Bd)*Bd'*P*Ad
    matMul(SumR, SumR_row*SumR_col, SumR_row, SumR_col,
           BdT_P_Ad, BdT_P_Ad_row*BdT_P_Ad_col, BdT_P_Ad_row, BdT_P_Ad_col,
           K, K_row*K_col, K_row, K_col);
}

/* Matrix multiplication A*B=C */
short matMul(float A[A_L], unsigned A_L, short A_row, short A_col,
        float B[B_L], unsigned B_L, short B_row, short B_col,
        float C[C_L], unsigned C_L, short C_row, short C_col) {
    short err=0;
    int i=0,j=0,k=0;
    // verify matrix multriplication constraint
    if (A_col == B_row && C_col == B_col && C_row == A_row) {
        // navigate each A row
        for (i=0; i < A_row; i++){
            // navigate each B column
            for (j=0; j < B_col; j++){
                // initialize C value
                C[i*C_col + j] = 0.0;
                // perform A-row per B-column
                for (k=0; k < A_col; k++){
                    C[i*C_col + j] = C[i*C_col + j]
                        + (float)((double)A[i*A_col + k] * (double)B[k*B_col + j]);
                }
            }
        }
    } else {
        err = 1;
    }
    return err;
}

/* Matrix traspose At=B */
short matTr(float A[A_L], unsigned A_L, short A_row, short A_col,
        float B[B_L], unsigned B_L, short B_row, short B_col) {
    short err = 0;
    int i=0, j=0;
    // verify B matrix is proper
    if (A_row == B_col && A_col == B_row) {
        // perform trasposition
        for (i=0; i < A_row; i++){
             for (j=0; j < A_col; j++){
                B[j*B_col + i] = A[i*A_col + j];
             }
        }
    } else { err = 1; }
    return err;
}

/* Matrix sum - A + sign*B = C - sign={1,-1} */
```

```
short matSum(float A[A_L], unsigned A_L, short A_row, short A_col,
        float B[B_L], unsigned B_L, short B_row, short B_col,
        float C[C_L], unsigned C_L, short C_row, short C_col, short sign) {
    int i=0, j=0;
    short err=0;
    if (sign != 1) {
        if (sign != -1) {
            sign = 1;
            err = 1;
        }
    }
    if (A_col == B_col && A_row == B_row) {
    for(i=0; i < C_row; i++){
            for(j=0; j < C_col; j++){
                C[i*C_col + j] = A[i*A_col + j] + sign*B[i*B_col + j];
            }
        }
    } else { err = 1; }
    return err;
}
```

In ultimo il ciclo del controllore ed il calcolo della matrice $A(x)$:

```
    void controller(client interface sonar_i son_i,
        client interface encoder_i enc_i,
        client interface motor_i mot_i,
        float Ad[Ad_L], unsigned Ad_L, short Ad_row, short Ad_col,
        float Bd[Bd_L], unsigned Bd_L, short Bd_row, short Bd_col,
        float Cd[Cd_L], unsigned Cd_L, short Cd_row, short Cd_col,
        float Q[Q_L], unsigned Q_L, short Q_row, short Q_col,
        float P[P_L], unsigned P_L, short P_row, short P_col,
        float Qf[Qf_L], unsigned Qf_L, short Qf_row, short Qf_col,
        float Rf[Rf_L], unsigned Rf_L, short Rf_row, short Rf_col,
        float K[K_L], unsigned K_L, short K_row, short K_col){

    timer t,t1;
    float init_enc = 0.0, init_position = 0.0;
    short i=0, initFlag = 0, resFlag = 0;
    short distFlag=0, distDel=0, distLen=0;
    unsigned int now = 0, time = 0, del = 0, uno,due;
    // selettore simulazione 1 fault motore 0 angolo non zero
    short selettore=0;

    float R=1;
    float Ad_lin[5*5];
    for (i=0; i<25; i++){
        Ad_lin[i] = Ad[i];
    }
    float Bd_lin[5*1];
    for (i=0; i<5; i++){
        Bd_lin[i] = Bd[i];
    }

    short U_row = 1, U_col = 1;
    float U[1*1] = {0.0};

    short X_row = 5, X_col = 1;
    float X[5*1]={0.0,0.0,0.0,0.0,0.0};
    if (!selettore) {
       float X[5*1]={0.0,0.0,10.0*3.1415/180,0.0,0.0};
    }
```

```
short Pf_row = 5, Pf_col = 5;
float Pf[5*5]= {1e-6,  0.0,  0.0,   0.0,   0.0,
                0.0,   1e-6, 0.0,   0.0,   0.0,
                0.0,   0.0,  1e-6,  0.0,   0.0,
                0.0,   0.0,  0.0,   1e-6,  0.0,
                0.0,   0.0,  0.0,   0.0,   1e-6};

// Letture sensori
short Y_row = 2, Y_col = 1;
float Y[2*1] = {0.0, 0.0};

if (!selettore) {
// Update Ad Bd
sdc_Ad(Ad, Ad_row*Ad_col, Ad_row, Ad_col,
       Bd, Bd_row*Bd_col, Bd_row, Bd_col,
        X, X_row*X_col, X_row, X_col);

// Update P
for (i=0; i<100; i++) {
riccatiEQ(Ad, Ad_row*Ad_col, Ad_row, Ad_col,
          Bd, Bd_row*Bd_col, Bd_row, Bd_col,
          Cd, Cd_row*Cd_col, Cd_row, Cd_col,
          Q,  Q_row*Q_col, Q_row, Q_col,
          P,  P_row*P_col, P_row, P_col, R);
}
}

if (initFlag != 1) {
    init_position = 0.5;
    init_enc = 1.613 - 0.0016;//90*3.1415/180 - 0.0016;
    initFlag = 1;
    mot_i.setVolt(0.0);
    t :> del;
    del += 1000000000;
    t when timerafter(del) :> void;
}
t :> time;

while(1){
    select{
        case !resFlag => t when timerafter(time) :> void:
                t:> uno;
            if (time >= MAXTIME - Tc) {
                time = time - (MAXTIME - Tc);
                resFlag = 1;
                t :> now;
            } else {
            time += Tc;
            }

        // data read
        Y[0] = - (son_i.getPos() - init_position);
        Y[1] = enc_i.getAngle() - init_enc;


        kalmanFilter(Ad, Ad_row*Ad_col, Ad_row, Ad_col,
                     Bd, Bd_row*Bd_col, Bd_row, Bd_col,
                     Cd, Cd_row*Cd_col, Cd_row, Cd_col,
                     Y, Y_row*Y_col, Y_row, Y_col,
                     X, X_row*X_col, X_row, X_col,
                     Pf, Pf_row*Pf_col, Pf_row, Pf_col,
```

```
                        Qf, Qf_row*Qf_col, Qf_row, Qf_col,
                        Rf, Rf_row*Rf_col, Rf_row, Rf_col,
                        U, U_row*U_col, U_row, U_col);

            // Update Ad Bd
            sdc_Ad(Ad, Ad_row*Ad_col, Ad_row, Ad_col,
                   Bd, Bd_row*Bd_col, Bd_row, Bd_col,
                    X, X_row*X_col, X_row, X_col);

            // Update P
            for (i=0; i<20; i++) {
            riccatiEQ(Ad, Ad_row*Ad_col, Ad_row, Ad_col,
                      Bd, Bd_row*Bd_col, Bd_row, Bd_col,
                      Cd, Cd_row*Cd_col, Cd_row, Cd_col,
                      Q,  Q_row*Q_col, Q_row, Q_col,
                      P,  P_row*P_col, P_row, P_col, R);
            }

            // Update K
            gainK(Ad, Ad_row*Ad_col, Ad_row, Ad_col,
                  Bd, Bd_row*Bd_col, Bd_row, Bd_col,
                  P, P_row*P_col, P_row, P_col,
                  K, K_row*K_col, K_row, K_col, R);

/*          for (int i=0; i < K_col*K_row; i++){
                        printf("%f ", K[i]);
                    }
                printf("\n");
*/
            U[0] = 0.0;
            for(i=0; i < K_col; i++){
                    U[0] = U[0] - K[i] * X[i];
            }

            U[0] = - U[0]; //temp invertire cavi
                // Saturation
                if (U[0] > 12.0) {
                    U[0] = 12.0; }
                if (U[0] < -12.0) {
                    U[0] = -12.0; }

                printf("Y0: %.3f ", Y[0]);
                printf("u: %.1f ", U[0]);
                printf("x: %.3f ", X[0]);
                printf("dx: %.3f ", X[1]);
                printf("th: %.3f ", X[2]*180/3.1415);
                printf("dth: %.3f \n", X[3]);
/* */
                // pwm with current iteration value + disturbe
                if (selettore) {
                if (!distFlag) {
                    distDel++;
                    if (distDel >= 10*50) {
                        if (X[1]>0 && X[2]>-0.01 && X[2]<0.01 && X[3]<0) {
                            distFlag = 1;
                        }
                    }
                } else {printf("dist on ");}
                if (distFlag && distLen < 1) {
                    distLen++;
                    mot_i.setVolt(0.0);
```

```
                } else {
                    mot_i.setVolt(U[0]);
                }
                } else {
                mot_i.setVolt(U[0]);
                }
                U[0] = - U[0]; //temp invertire cavi HW

                /*// deadzone per kalman
                if (U[0] < -0.8 && U[0] > 0.8) {U[0]=0;}
                if (U[0] < -0.8) {U[0]=U[0]+0.8;}
                if (U[0] >  0.8) {U[0]=U[0]-0.8;}
                */
                xscope_float(0, X[0]*100);
                xscope_float(1, X[1]);
                xscope_float(2, X[2]*180/3.1415);
                xscope_float(3, X[3]);
                xscope_float(4, U[0]);
                xscope_float(5, Y[0]);
                xscope_float(6, Y[1]*180/3.1415);

                t :> due;
                printf("t %.3fms\n",(float)(due-uno)/100000.0);
            break;

            case resFlag => t1 when timerafter(now+100) :> now:
                if (now < time) {
                    resFlag = 0;
                }
                break;
        }
    }
}

void sdc_Ad(float Ad[Ad_L], unsigned Ad_L, short Ad_row, short Ad_col,
            float Bd[Bd_L], unsigned Bd_L, short Bd_row, short Bd_col,
            float X[X_L], unsigned X_L, short X_row, short X_col) {

        double cos_x3   = cos(X[2]);
        double cos_x3_2 = pow(cos_x3,2.0);
        double sin_x3   = sin(X[2]);
        double den = (0.00041616*cos_x3_2 - 0.0179658585);

        Ad[1*Ad_col+1] = 0.00001260762/den + 1.0;
        if(X[2] < 0.001 && X[2] > -0.001){
            Ad[1*Ad_col+2] = (0.0000816214608*cos_x3)/(den);
            Ad[3*Ad_col+2] = -(0.0114029982)/(den);
        }else{
            Ad[1*Ad_col+2] = (0.0000816214608*cos_x3*sin_x3)/(X[2]*den);
            Ad[3*Ad_col+2] = -(0.0114029982*sin_x3)/(X[2]*den);
        }
        Ad[1*Ad_col+3] = -(0.00000257195448*X[3]*sin_x3)/den;
        Ad[1*Ad_col+4] = 0.0002033077202/den;
        Ad[3*Ad_col+1] = -(0.0000408*cos_x3)/den;
        Ad[3*Ad_col+3] = (0.0000083232*X[3]*cos_x3*sin_x3)/den + 1.0;
        Ad[3*Ad_col+4] = -(0.0006579318686*cos_x3)/den;
        Bd[1] = -0.000357/den;
        Bd[3] =  (0.001155*cos_x3)/den;
}
```

Appendice B.

Implementazione filtraggio e controllo piatto vibrante

Di seguito il codice che implementa il filtraggio IIR/Kalman ed il controllo bang-off su scheda STM del piatto vibrante.

```
int16_t myAccX, myAccY;
float myAccX_dec, myAccY_dec, myAccZ_dec;
float myAccX_conv, myAccY_conv, myAccZ_conv;
float myAccY_prec = 0;
float Acc_offset = 0.6472554692;  //accelerometer offset
float myVelY;   //velocity estimated
float myVelY_prec = 0;
float myVelY_lin;
float myVelY_lin_prec = 0;
float myOsc;   //position estimated
float myOsc_prec = 0;
float myOsc_lin;
float myOsc_lin_prec = 0;

float Yk, x1k_1, x2k_1, Uk_1 = 0;
float mem[1000], sum=0, offset=0;
float myAccY_lin=0, myAccY_lin_prec=0;
int im=0, flag=0, on=1, dzero=0;
float x1=0,x2=0,x3=0,x4=0,x1p=0,x2p=0,x3p=0,x4p=0,u1=0;
int myTk2, myTk2_1;

int myTk, myTk_1;   //measure clock
int myTk1 = 0;      //control clock

float z;   //switching function

float Ts = 0.001;  //sampling time
float g = 9.81;  // m/s^2
//mechanical resonance frequency of conveyor
float w0 = 334.1766;  //resonance frequency in rad/s
float A = 1.04e-3 * 334.1766;
float Eps = 0.01; //--switching function tolerance
float Eps_V = 0.0;   //velocity tolerance
int u = 0;  //control variable

char serial_data[100];   //array used to send data to pc

void user_loop() {
        uint32_t t;
        myTk = HAL_GetTick();
```

```
if (myTk >= myTk_1 + 1){      //--function works every 1 ms

        //raw accelerometer data in accelerometer unit (scale is set to 24g)
        myAccX_dec = (float)(Acc_getX());
        myAccY_dec = (float)(Acc_getY());
        myAccZ_dec = (float)(Acc_getZ());

        //raw accelerometer data converted in m/s^2
        myAccX_conv = (myAccX_dec *24 *g) / 32767;
        myAccY_conv = ((myAccY_dec *24 *g) / 32767);
        myAccZ_conv = (myAccZ_dec *24 *g) / 32767;

        //A simple moving average filter can be used to reduce the noise in t
        //acceleration acquired
        t = HAL_GetTick();
        if (t<10000){
        sum = sum - mem[im];
        mem[im] = myAccY_conv;
        sum = sum + mem[im];
        im++;
        if (im >= 1000) {
                im = 0;
        }
        offset = sum/1000;
        }
        myAccY_conv = myAccY_conv-offset;

        if(t > 10000){

        // ---------- filtro passa banda 10 200 hz ord 4
        u1 = myAccY_conv;

        x1p = 0.436597*u1 + 1.91155*x1 - 0.915653*x2;
    x2p = x1;
        x3p = 0.190617*u1 + 0.834577*x1 - 0.836368*x2 + 0.423531*x3 - 0.22571
x4p = x3;

        myAccY_lin = 0.190617*u1 + 0.834577*x1 - 0.836368*x2 + 0.423531*x3 -
        x1 = x1p;
        x2 = x2p;
        x3 = x3p;
        x4 = x4p;

    //------GET VELOCITY WITH NUMERICAL INTEGRATION
        myVelY = myVelY + myAccY_lin*Ts;


        //------KALMAN OBSERVER--2x2-------------------------------
        Yk = myVelY;
        x1k_1 = myOsc_lin_prec;
        x2k_1 = myVelY_lin_prec;

        myOsc_lin = 8.488e-7*Uk_1 - 0.00797*Yk + 0.009093*x1k_1 + 0.008837*x2
        myVelY_lin = 3.36e-7*Uk_1 + 0.9928*Yk - 0.3524*x1k_1 + 0.003143*x2k_1
        myOsc_lin_prec = myOsc_lin;
        myVelY_lin_prec = myVelY_lin;

        //-----------------------------------------------------

        }
```

```
                myTk1 = myTk1 + 1;  //---update control clock

                //-----SWITCHING FUNCTION-----------------------------------------------------
                z = w0*w0*myOsc_lin*myOsc_lin + myVelY_lin*myVelY_lin - A*A;

                //-----CONTROL ALGORITHM------------------------------------------------------
                if(t > 20000 && t < 30000){             //set working time
                        float ptemp = (myOsc_lin < 0) ? -myOsc_lin : myOsc_lin;
                        if (myTk1 >= 8){        //--define sample time for control
                                float vtemp = (myVelY_lin < 0) ? -myVelY_lin : myV
                                float ztemp = (z < 0) ? -z : z;
                                if (ztemp < Eps) {
                                        u = 0;
                                }
                                else if (vtemp < 0.004) {
                                        u = 1;
                                }
                                else if (z<0){
                                        if(myVelY_lin>=Eps_V){
                                                u = 1;
                                        }
                                        else{
                                                u = 0;
                                        }
                                }
                                else if (z>0){
                                        if(myVelY_lin<=-Eps_V){
                                                u = 0;
                                        }
                                        else{
                                                u = 0;
                                        }
                                }
                                else{
                                        u = 0;
                                }
                                myTk1 = 0;    //--reset control clock

                                if (ptemp > 1.5e-4 && on) {
                                        on = 0;
                                }
                        }
                        HAL_GPIO_WritePin(CMD_1_GPIO_Port, CMD_1_Pin, u);
//--apply control input to move conveyor

                }
                else{
                        u = 0;
                        if (t > 30000) {
                        //--turn off conveyor
                        HAL_GPIO_WritePin(CMD_2_GPIO_Port, CMD_2_Pin, 0);
                        }
                }

                Uk_1 = 160*u;

                //-----------END CONTROL ALGORITHM-------------------------------------------


                //---------SYMBOLIC CONVERSION OF FLOAT DATA FOR sprintf()-----------
                //-----data are expressed in the form: sign/integer part/decimal pa
```

```
        //----acceleration----------------------------------------------------------
        char *tmpSignY = (myAccY_conv < 0) ? "-" : "";
        float tmpValY = (myAccY_conv < 0) ? -myAccY_conv : myAccY_conv;
        int tmpInt1Y = tmpValY;                    // Get the integer (678).
        float tmpFracY = tmpValY - tmpInt1Y;       // Get fraction (0.0123).
        int tmpInt2Y = trunc(tmpFracY * 10000);  // Turn into integer (123).

        //----velocity with dc blocker---------------------------------------------
        char *tmpSignV = (myVelY_lin < 0) ? "-" : "";
        float tmpValV = (myVelY_lin < 0) ? -myVelY_lin : myVelY_lin;
        int tmpInt1V = tmpValV;                    // Get the integer (678).
        float tmpFracV = tmpValV - tmpInt1V;       // Get fraction (0.0123).
        int tmpInt2V = trunc(tmpFracV * 100000);  // Turn into integer (123)

        //----position with dc blocker---------------------------------------------
        char *tmpSignO = (myOsc_lin < 0) ? "-" : "";
        float tmpValO = (myOsc_lin < 0) ? -myOsc_lin : myOsc_lin;
        int tmpInt1O = tmpValO;                    // Get the integer (678).
        float tmpFracO = tmpValO - tmpInt1O;       // Get fraction (0.0123).
        int tmpInt2O = trunc(tmpFracO * 100000000);  // Turn into integer (1

        //----switching function---------------------------------------------------
        char *tmpSignZ = (z < 0) ? "-" : "";
        float tmpValZ = (z < 0) ? -z : z;
        int tmpInt1Z = tmpValZ;                    // Get the integer.
        float tmpFracZ = tmpValZ - tmpInt1Z;       // Get fraction.
        int tmpInt2Z = trunc(tmpFracZ * 100000);  // Turn into integer. //ag
        //------END DATA CONVERSION-----------------------------------------------

        //-----put data in array---------------------------------------------------
        sprintf (serial_data, "%s%d.%04d %s%d.%05d %s%d.%08d %s%d.%05d %d\n\r
                        tmpSignY, tmpInt1Y, tmpInt2Y,
                        tmpSignV, tmpInt1V, tmpInt2V, tmpSignO, tmpInt1O, tm
                        tmpSignZ, tmpInt1Z, tmpInt2Z, u);  //tmpSignZ, tmpInt

        //send data to the PC through serial port
        Debug_print(serial_data, strlen(serial_data));
        //-------------------------------------------------------------------------

        //save actual time for next cycle
        myTk_1 = myTk;
    }

}
```

Appendice C.

Implementazione filtraggio metodo min/max e kalman per la bilancia

Il metodo min/max è molto semplice, nota la derivata prima del segnale è possibile identificare un punto di minimo e massimo dove si annulla. Memorizzando il valore del segnale nel medesimo istante, si procede stimando la misura del peso come punto medio fra gli ultimi due valori così trovati.

```
function [y, dw_n, w1_n] = stimaD(w,dw,y_old,dw_old,w1)

if sign(dw) ~= sign(dw_old)
    if w > w1
        y = w1+(w-w1)/2;
    else
        y = w+(w1-w)/2;
    end
    w1_n = w;
else
    y = y_old;
    w1_n = w1;
end

dw_n = dw;

end
```

Il filtro di kalman SDC è più complesso ma molto più efficace. Di seguito il codice che lo implementa.

```
function [Xk,Pk] = Kalman_sdc(Yk,Xk_1,Pk_1)

w = Xk_1(3);

A = [
[                    0,                    1.0, 0]
[ -9068.8/(w + 2.022), -34.5714/(w + 2.022), 9.81/(w+2.022)]
[0, 0, 0]
];

B = 0;

C = [9068.8/9.81 0 0];

% Sampling time
Ts = 0.005; %sec
```

```
%Calculate discrete-time state-space matrices
Ad = (eye(3)+Ts*A);
Bd = Ts*B;
Cd = C;

Uk_1 = 0;
%% measure noise
Rv = 0;

%% process noise
Ac = A(1:2,1:2);
Q2 = inv(Ac'*Ac);
Qw = [Q2, [0;0];
      [0,0], 1];

%% Kalman filtering (Observer)

    %%%%% new prediction step
    xk_es    = Ad * Xk_1 + Bd * Uk_1;
    Pk_temp  = Ad * Pk_1 * Ad' +  Qw;

    %%%% new update step
    ek = Yk - Cd * xk_es;
    Kk = Pk_temp * Cd' * inv(Cd*Pk_temp*Cd' + Rv);
    Xk = xk_es + Kk * ek; % correcting the state estimate x_ec
    Pk = Pk_temp - Kk*Cd*Pk_temp;

end
```

Appendice D.

Pubblicazioni su attività collaterali

Nel corso dell'attività di dottorato è stato modellato un drone DuctedFan composto da due eliche intubate contrapposte, dove l'assetto è regolato da due flap incrociati posti all'uscita del flusso. Lo studio del modello ha portato alla pubblicazione [45], dove il sistema è stato realizzato al fine di testare tecniche di controllo non lineari come quella in oggetto di studio in questa tesi.

Bibliografia

[1] JD Pearson, "Approximation methods in optimal control i. sub-optimal control," *International Journal of Electronics*, vol. 13, no. 5, pp. 453–469, 1962.

[2] Tayfun Çimen, "Systematic and effective design of nonlinear feedback controllers via the state-dependent riccati equation (sdre) method," *Annual Reviews in control*, vol. 34, no. 1, pp. 32–51, 2010.

[3] Tayfun Cimen, "Survey of state-dependent riccati equation in nonlinear optimal feedback control synthesis," *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 4, pp. 1025–1047, 2012.

[4] Andreas Wernli and Gerald Cook, "Suboptimal control for the nonlinear quadratic regulator problem," *Automatica*, vol. 11, no. 1, pp. 75–84, 1975.

[5] James R Cloutier, Christopher N D'Souza, and Curtis P Mracek, "Nonlinear regulation and nonlinear h∞ control via the state-dependent riccati equation technique: Part 1, theory," in *Proceedings of the international conference on nonlinear problems in aviation and aerospace*. Embry Riddle University, 1996, pp. 117–131.

[6] Kelly Douglas Hammett, *Control of nonlinear systems via state feedback state-dependent Riccati equation techniques*, Air Force Institute of Technology, 1997.

[7] James R Cloutier, Donald T Stansbery, and Mario Sznaier, "On the recoverability of nonlinear state feedback laws by extended linearization control techniques," in *Proceedings of the 1999 American Control Conference (Cat. No. 99CH36251)*. IEEE, 1999, vol. 3, pp. 1515–1519.

[8] Evrin Bilge Erdem, *Analysis and real-time implementation of state-dependent Riccati equation controlled systems*, University of Illinois at Urbana-Champaign, 2001.

[9] James Cloutier and Donald Stansbery, "All-aspect acceleration-limited homing guidance," in *Guidance, Navigation, and Control Conference and Exhibit*, 1999, p. 4063.

[10] Neil F Palumbo and Todd D Jackson, "Integrated missile guidance and control: A state dependent riccati differential equation approach," in *Proceedings of the 1999 IEEE International Conference on Control Applications (Cat. No. 99CH36328)*. IEEE, 1999, vol. 1, pp. 243–248.

[11] David K Parrish and D Brett Ridgely, "Attitude control of a satellite using the sdre method," in *Proceedings of the 1997 American Control Conference (Cat. No. 97CH36041)*. IEEE, 1997, vol. 2, pp. 942–946.

[12] Kelly D Hammett, Christopher D Hall, and D Brett Ridgely, "Controllability issues in nonlinear state-dependent riccati equation control," *Journal of guidance, control, and dynamics*, vol. 21, no. 5, pp. 767–773, 1998.

[13] Donald T Stansbery and James R Cloutier, "Position and attitude control of a spacecraft using the state-dependent riccati equation technique," in *Proceedings of the 2000 American Control Conference. ACC (IEEE Cat. No. 00CH36334)*. IEEE, 2000, vol. 3, pp. 1867–1871.

[14] Evrin B Erdem and Andrew G Alleyne, "Experimental real-time sdre control of an underactuated robot," in *Proceedings of the 40th IEEE conference on decision and control (Cat. No. 01CH37228)*. IEEE, 2001, vol. 3, pp. 2986–2991.

[15] David K Parrish and D Brett Ridgely, "Control of an artificial human pancreas using the sdre method," in *Proceedings of the 1997 American Control Conference (Cat. No. 97CH36041)*. IEEE, 1997, vol. 2, pp. 1059–1060.

[16] Jie Yu, Ali Jadbabaie, James Primbs, and Yun Huang, "Comparison of nonlinear control design techniques on a model of the caltech ducted fan," *Automatica*, vol. 37, no. 12, pp. 1971–1978, 2001.

[17] Paulina Superczyńska, Oskar Lindenau, Marcin Wa, et al., "Sdre-based suboptimal controller for manipulator control," in *2019 12th International Workshop on Robot Motion and Control (RoMoCo)*. IEEE, 2019, pp. 21–25.

[18] Pouria Razzaghi, Ehab Al Khatib, and Shide Bakhtiari, "Sliding mode and sdre control laws on a tethered satellite system to de-orbit space debris," *Advances in Space Research*, vol. 64, no. 1, pp. 18–27, 2019.

[19] Insu Chang and Soon-Jo Chung, "Exponential stability region estimates for the state-dependent riccati equation controllers," in *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*. IEEE, 2009, pp. 1974–1979.

[20] Robbin van Hoek, Mohsen Alirezaei, Antoine Schmeitz, and Henk Nijmeijer, "Vehicle state estimation using a state dependent riccati equation," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 3388–3393, 2017.

[21] Eduardo Gallestey, Peter Al-Hokayem, Mr Giampaolo Torrisi, and Mr Dario Paccagnan, "Nonlinear systems and control," *Department of Information Technology and Electrical Engineering, Swiss Federal Institute of Technology*, 2015.

[22] Yew-Wen Liang and Li-Gang Lin, "Analysis of sdc matrices for successfully implementing the sdre scheme," *Automatica*, vol. 49, no. 10, pp. 3120–3124, 2013.

[23] Ahmed Mohamed Abdelhady Khamis, *Advanced tracking strategies for linear and nonlinear control systems: Theory and applications*, Ph.D. thesis, Idaho State University, 2014.

[24] Ahmed Khamis and D Subbaram Naidu, "Real-time algorithm for nonlinear systems with incomplete state information using finite-horizon optimal control technique," in *2014 7th International Symposium on Resilient Control Systems (ISRCS)*. IEEE, 2014, pp. 1–6.

[25] Ahmed Khamis, Cheng-Hung Chen, and D Subbaram Naidu, "Tracking of a robotic hand via sd-dre and sd-dve strategies," in *2016 UKACC 11th International Conference on Control (CONTROL)*. IEEE, 2016, pp. 1–6.

[26] Francesco Topputo, Martino Miani, and Franco Bernelli-Zazzera, "Optimal selection of the coefficient matrix in state-dependent control methods," *Journal of Guidance, Control, and Dynamics*, vol. 38, no. 5, pp. 861–873, 2015.

[27] Engin H Copur, Ahmet C Arican, Sinan Ozcan, and Metin U Salamci, "An update algorithm design using moving region of attraction for sdre based control law," *Journal of the Franklin Institute*, vol. 356, no. 15, pp. 8388–8413, 2019.

[28] Fernando Ornelas-Tellez, J Jesus Rico-Melgoza, Elisa Espinosa-Juarez, and Edgar N Sanchez, "Optimal and robust control in dc microgrids," *IEEE Transactions on Smart Grid*, vol. 9, no. 6, pp. 5543–5553, 2017.

[29] Cagatay Cebeci, Michael Grimble, Luis Recalde-Camacho, and Reza Katebi, "Sdre preview control for a lpv modelled autonomous vehicle," *IFAC-PapersOnLine*, vol. 52, no. 28, pp. 114–119, 2019.

[30] Thien Van Nguyen, Ana-Maria Bordei, Trang Minh Nguyen, and Achim Ionita, "Using pid controller and sdre methods for tracking control of

spacecrafts in closed-rendezvous process," *INCAS Bulletin*, vol. 11, no. 1, pp. 139–150, 2019.

[31] Li-Gang Lin and Ming Xin, "Nonlinear control of two-wheeled robot based on novel analysis and design of sdre scheme," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 3, pp. 1140–1148, 2019.

[32] Li-Gang Lin and Ming Xin, "Guaranteed continuity and computational improvement in sdre controllers for cancer treatment analysis," *Journal of Dynamic Systems, Measurement, and Control*, vol. 142, no. 4, pp. 041005, 2020.

[33] Saeed Rafee Nekoo, "Tutorial and review on the state-dependent riccati equation," *Journal of Applied Nonlinear Dynamics*, vol. 8, no. 2, pp. 109–166, 2019.

[34] Andrea Bonci, Sauro Longhi, and Giuseppe Antonio Scala, "Towards an all-wheel drive motorcycle: Dynamic modeling and simulation," *IEEE Access*, vol. 8, pp. 112867–112882, 2020.

[35] Alberto Isidori, J van Schuppen, E Sontag, M Thoma, and M Krstic, "Communications and control engineering," in *Nonlinear control systems*. Springer, 1995.

[36] PK Menon, T Lam, LS Crawford, and VHL Cheng, "Real-time computational methods for sdre nonlinear control of missiles," in *Proceedings of the 2002 American Control Conference (IEEE Cat. No. CH37301)*. IEEE, 2002, vol. 1, pp. 232–237.

[37] Pritpal Dang and Frank L Lewis, "Controller for swing-up and balance of single inverted pendulum using sdre-based solution," in *31st Annual Conference of IEEE Industrial Electronics Society, 2005. IECON 2005.* IEEE, 2005, pp. 6–pp.

[38] Andrea Bonci, Sauro Longhi, Giacomo Nabissi, and Giuseppe Antonio Scala, "Execution time of optimal controls in hard real time, a minimal execution time solution for nonlinear sdre," *IEEE Access*, vol. 8, pp. 158008–158025, 2020.

[39] Andrea Bonci, Giacomo Nabissi, and Giuseppe Antonio Scala, "Hard real time embedded solution for execution time analysis of non-linear control laws," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2020, vol. 1, pp. 1289–1292.

[40] Chandrasekar Jaganath, Aaron Ridley, and Dennis S Bernstein, "A sdre-based asymptotic observer for nonlinear discrete-time systems," in *Proceedings of the 2005, American Control Conference, 2005.* IEEE, 2005, pp. 3630–3635.

[41] Mule PALA PRASAD Reddy and Jeevamma Jacob, "Vibration control of flexible link manipulator using sdre controller and kalman filtering," *Stud. Inform. Control*, vol. 2, pp. 143–150, 2017.

[42] Aleksandar I. Ribic and Željko V Despotovic, "High-performance feedback control of electromagnetic vibratory feeder," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 9, pp. 3087–3094, 2010.

[43] Andrea Bonci, Sauro Longhi, and Giuseppe Antonio Scala, "Time-optimal feedback control of industrial vibratory feeder," In fase di sottomissione.

[44] Andrea Bonci, Sauro Longhi, and Giuseppe Antonio Scala, "Fast dynamic weighing estimation of load cells using averaged inflection points technique," In fase di sottomissione.

[45] Andrea Bonci, Alice Cervellieri, Sauro Longhi, Giacomo Nabissi, and Giuseppe Antonio Scala, "The double propeller ducted-fan, an uav for safe infrastructure inspection and human-interaction," in *2020 25th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*. IEEE, 2020, vol. 1, pp. 727–733.