



**UNIVERSITÀ DEGLI STUDI DI MESSINA  
DIPARTIMENTO DI INGEGNERIA**

**Corso di Laurea in Ingegneria Elettronica e Informatica (L-8)**

---

**Slow Feature Analysis per Sistemi Lineari e Non Lineari**

Tesi di laurea di:  
**Gabriele Rinaldi**

Relatore:  
**Chiar.ma Prof. Maria Gabriella Xibilia**

---

**Anno Accademico 2022/23**

# Sommario

1	Capitolo: Introduzione .....	4
2	Capitolo: Teoria .....	5
2.1	Dynamic mode Decomposition.....	5
2.1.1	Descrizione Matematica.....	5
2.1.2	Algoritmo DMD.....	7
2.2	Dynamic Mode Decomposition with Control.....	8
2.2.1	Descrizione matematica .....	8
2.2.2	Algoritmo DMDc .....	14
2.2.3	Implementazione Python.....	15
2.3	Multi Resolution Dynamic Mode Decomposition .....	17
2.3.1	Descrizione matematica .....	17
2.3.2	Algoritmo mrDMD .....	20
2.4	Multi Resolution Dynamic Mode Decomposition with Control.....	21
2.4.1	Algoritmo mrDMDc .....	21
2.4.2	Implementazione Python.....	27
3	Capitolo: Casi di studio.....	33
3.1	Dataset Industriale: Surful Recovery Unit (SRU).....	33
3.2	Dati sintetici .....	35
3.3	Vehicle to Grid (V2G) .....	37
4	Capitolo: Risultati .....	39
4.1	Key Performance Indicator .....	39
4.2	DMDc.....	41
4.2.1	SRU .....	41
4.2.1	Sistema Sintetico con 5 Ingressi con ritardi dal Sistema SRU .....	44
4.2.2	Sistema sintetico con ingresso n.5 non ritardato dai dati SRU .....	48
4.2.3	V2G .....	51
4.3	mrDMDc .....	64
4.3.1	Ricostruzione a passo 1 .....	64
4.3.2	Ricostruzione a passo step .....	74
4.4	Confronto DMDc Vs. mrDMDc .....	125
4.4.1	SRU .....	125
4.4.2	Sistema Sintetico con 5 Ingressi con ritardi dal Sistema SRU .....	125
4.4.3	Sistema sintetico con ingresso n.5 non ritardato dai dati SRU .....	125
4.4.4	V2G .....	125
5	Conclusioni .....	127
6	Bibliografia .....	128



# 1 Capitolo: Introduzione

Negli ultimi anni, la crescente disponibilità di dati in vari settori ha aperto nuove prospettive per l'analisi dei dati. L'approccio orientato ai dati, che si basa sull'utilizzo delle informazioni disponibili, ha assunto un ruolo cruciale nel fornire soluzioni intelligenti per la gestione dei dati.

Questa tesi si focalizza sull'applicazione delle metodologie orientate ai dati nel contesto industriale, esplorando come i dati possano essere elaborati e impiegati per ottenere risultati utili al miglioramento dei processi industriali coinvolti. I dati, provenienti da diverse fonti, come sensori e analisi di laboratorio, offrono informazioni preziose che, se analizzate correttamente, possono contribuire all'ottimizzazione dei sistemi, all'individuazione di anomalie, alla previsione delle prestazioni e altro ancora. L'utilizzo delle metodologie orientate ai dati consente di massimizzare il valore di questa vasta quantità di informazioni disponibili.

Nel corso dello sviluppo di questa tesi, verranno esaminati sia gli aspetti teorici che quelli pratici dell'approccio orientato ai dati. Saranno presi in oggetto i seguenti algoritmi:

- Dynamic Mode Decomposition (DMD),
- Dynamic Mode Decomposition with Control (DMDc),
- Multi-Resolution Dynamic Mode Decomposition (mrDMDc).

Il fine di questa tesi è proporre un'evoluzione del DMDc andando ad integrare l'analisi Multi-Resolution, definendo così un nuovo algoritmo denominato Multi-Resolution Dynamic Mode Decomposition with Control (mrDMDc). Chiaramente per avanzare una proposta di implementazione dell'algoritmo mrDMDc è necessario analizzare a fondo il comportamento del DMDc e dell'mrDMD, che verranno discussi nel Capitolo 2.

Vengono riassunti brevemente i punti della tesi:

- Nel Capitolo 2 andremo a discutere la teoria legata agli algoritmi DMD, DMDc, mrDMD e mrDMDc, ed in particolare per l'mrDMDc verrà spiegato passo-passo ogni punto dell'algoritmo, riportando poi anche l'implementazione nel Capitolo 4 che è possibile trovare anche sulla piattaforma GitHub[12].
- Nel Capitolo 3 saranno discussi i casi di studio che saranno poi presi in considerazione per il training ed il test degli algoritmi DMDc e mrDMDc.
- Nel Capitolo 4 verranno presentati i risultati applicati ai casi di studio.

## 2 Capitolo: Teoria

In questo capitolo si discute la teoria legata agli algoritmi presentati nell'indice.

### 2.1 Dynamic mode Decomposition

L'algoritmo DMD, introdotto per la prima volta da Peter Schmid nel 2008, permette di estrarre delle caratteristiche di un sistema dinamico, analizzando delle serie temporali di dati del sistema stesso [1]. Le caratteristiche dinamiche estratte dal DMD risultano lineari ma questo non comporta che il sistema dinamico di cui si vogliono estrarre le caratteristiche sia anch'esso necessariamente lineare. Chiaramente rappresenta un vantaggio in quanto possiamo studiare anche sistemi dinamici non lineari ed ottenere una modellizzazione lineare.

Inoltre, la conoscenza del modello di un sistema permette anche di prevedere le prestazioni del sistema stesso; dunque, ne migliora il controllo e l'affidabilità.

#### 2.1.1 Descrizione Matematica

Come è stato detto in precedenza il DMD analizza serie temporali di dati per estrarre le dinamiche di un sistema, dinamiche che generalmente sono non lineari, esprimibili con un'equazione del tipo[2]:

$$x_{k+1} = F(x_k)$$

*Equazione 2.1*

Consideriamo allora una serie temporale di stati del sistema, cioè matrici le cui righe rappresentano degli snapshots temporali e le cui colonne rappresentano gli stati del sistema

$$\mathbf{X} = \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_{m-1} \\ | & | & & | \end{bmatrix}$$

*Equazione 2.2*

Il DMD ci fornisce una previsione di stati futuri del nostro sistema dinamico, nell'Equazione 2.1 corrispondono agli  $x_{k+1}$ . Consideriamo allora la matrice  $X'$  come la matrice degli stati futuri, strutturata come la X, con le righe che rappresentano gli snapshots temporali e le colonne gli stati:

$$\mathbf{X}' = \begin{bmatrix} | & | & & | \\ \mathbf{x}_2 & \mathbf{x}_3 & \cdots & \mathbf{x}_m \\ | & | & & | \end{bmatrix}$$

*Equazione 2.3*

Si nota come gli elementi di  $X'$  risultano un campione in avanti rispetto ad X, proprio perché la nostra  $X'$  indica lo stato futuro del sistema dinamico.

Nell'introduzione del DMD è stato sottolineato come sia possibile fare l'analisi anche di sistemi dinamici non lineari, ma in ogni caso la modellizzazione viene fatta in modo lineare, quindi, tramite DMD ci aspettiamo un modello di questo tipo:

$$X' \approx AX$$

Equazione 2.4

Dove ogni elemento sarà così determinato:

$$x_{k+1} = Ax_k$$

Equazione 2.5

Vedendo l'Equazione 2.4 si nota chiaramente come l'operatore lineare A possa essere trovato semplicemente considerando la Moore-Penrose pseudo-inversa di X [3]:

$$A = X'X^\dagger$$

Equazione 2.6

Per calcolare la matrice A basterebbe quindi avere una serie temporale di stati con  $k = 0, \dots, m$  e considerare per  $X'$  gli istanti  $k = 1, \dots, m$ , mentre per  $X$  gli istanti  $k = 0, \dots, m - 1$ .

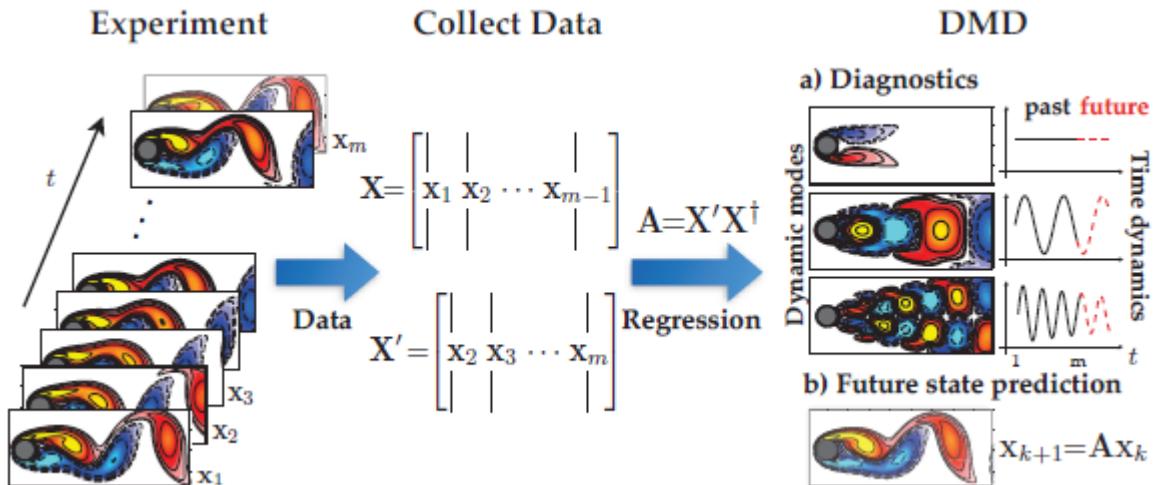


Figura 2.1

## 2.1.2 Algoritmo DMD

Il problema della risoluzione tramite l'Equazione 2.6 spesso è legato alle dimensioni della matrice X, poiché potrebbe risultare troppo grande per essere ritenuta calcolabile, dunque, è necessario operare in modo tale che non ci sia la necessità di considerare tutta la matrice X.

Per tale motivo l'algoritmo DMD sfrutta la Singular Value Decomposition (SVD) [4], che permette una riduzione a basso rango della matrice X.

Vediamo passo-passo come opera il DMD sfruttando la SVD:

1. Calcolare SVD di X, troncando se necessario:

$$X = U\Sigma V^*$$

Nota:  $V^*$  indica la trasposta coniugata di  $V$

*Equazione 2.7*

2. Calcolare  $\tilde{A}$ , la proiezione della matrice A su U:

$$\tilde{A} = U^*AU = U^*X'V\Sigma^{-1}$$

*Equazione 2.8*

3. Calcolare gli autovalori  $\lambda_i$  e gli autovettori  $w_i$  di  $\tilde{A}$ :

$$\tilde{A}W = W\Lambda$$

*Equazione 2.9*

4. Ricostruzione della decomposizione di A tramite W e  $\Lambda$ :

$$\Phi = X'V\Sigma^{-1}W$$

*Equazione 2.10*

## 2.2 Dynamic Mode Decomposition with Control

Il DMDc rappresenta un'evoluzione del DMD, poiché questo nuovo algoritmo è capace di modellizzare anche sistemi dinamici complessi nel quale intervengono non solo le dinamiche interne del sistema ma anche input esterni, analisi che non era possibile fare dal DMD in quanto non teneva conto dei possibili ingressi esogeni[2].

Il metodo DMDc aiuta a scoprire le dinamiche sottostanti senza l'effetto confondente del controllo esterno. Inoltre, il metodo quantifica anche l'effetto degli ingressi di controllo sulle misurazioni del sistema, mantenendo quelli che sono i vantaggi del DMD come la modellizzazione dei sistemi dinamici di controllo.

### 2.2.1 Descrizione matematica

Il sistema dinamico che ci permette di predire lo stato futuro adesso deve tener conto non solo dello stato corrente del sistema ma anche degli ingressi esogeni; possiamo dunque prendere in considerazione la canonica equazione di stato di un sistema di controllo dinamico:

$$x_{k+1} = F(x_k) + G(u_k)$$

*Equazione 2.11*

Dove:

- $x_{k+1}$  è la previsione dello stato:

$$\mathbf{X}' = \begin{bmatrix} | & | & \cdots & | \\ \mathbf{x}_2 & \mathbf{x}_3 & \cdots & \mathbf{x}_m \end{bmatrix}$$

*Equazione 2.12*

- $x_k$  è lo stato attuale:

$$\mathbf{X} = \begin{bmatrix} | & | & \cdots & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_{m-1} \end{bmatrix}$$

*Equazione 2.13*

- $u_k$  è l'ingresso attuale:

$$\mathbf{U} = \begin{bmatrix} | & | & \cdots & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \cdots & \mathbf{u}_{m-1} \end{bmatrix}$$

*Equazione 2.14*

La modellizzazione del DMDc sarà di questo tipo:

$$X' = AX + BY$$

Equazione 2.15

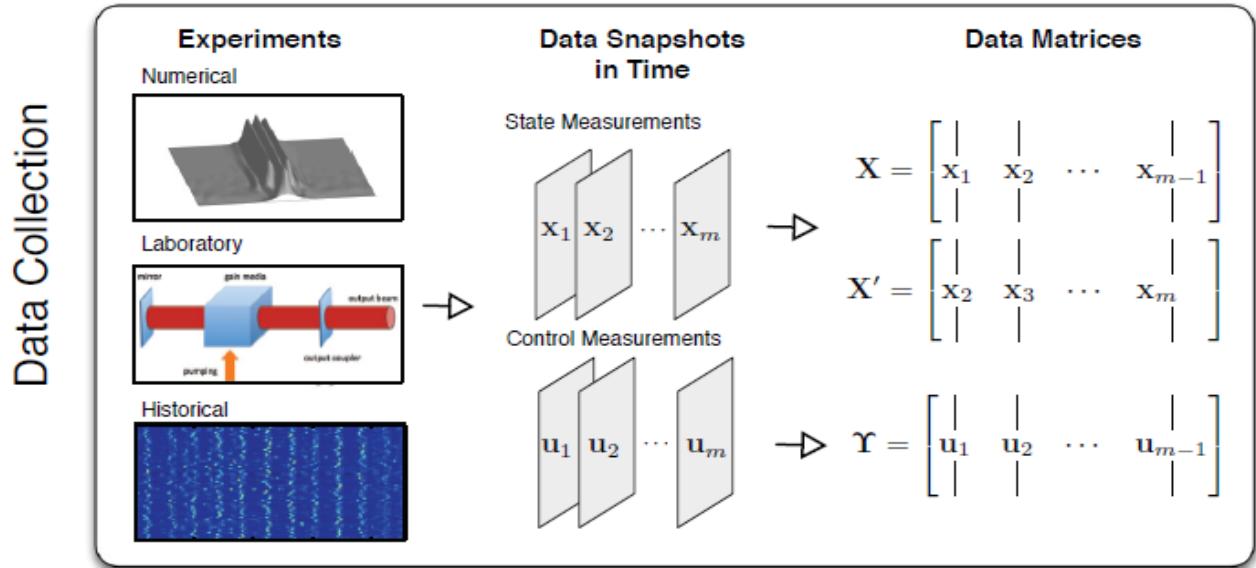


Figura 2.2

Considerando le matrici  $X'$ ,  $X$  e  $Y$  possiamo trovare le approssimazioni per le matrici  $A$  e  $B$ , oppure supporre di conoscere  $B$  e trovarci solo  $A$ . Vediamo più dettagliatamente questi due casi.

1. Caso in cui  $B$  è nota:

Si considera che la matrice  $B$  sia una buona approssimazione che permette di descrivere come l'ingresso influisce sul sistema dinamico, se non venisse stimata correttamente allora la previsione dello stato non sarebbe corretta.

Ipotizzando che la  $B$  sia stata stimata correttamente possiamo calcolarci la matrice  $A$ , dunque autovalori e modi dinamici del sistema in esame.

1. Come primo passo andiamo ad isolare la matrice  $A$ :

$$X' - BY \approx AX$$

Equazione 2.16

2. Applichiamo la SVD alla  $X$  (2.1.7) e sostituiamo:

$$X' - BY = A(U\Sigma V^*)$$

Equazione 2.17

3. Dalla quale otteniamo:

$$A = (X' - BY)V\Sigma^{-1}U^*$$

*Equazione 2.18*

4. In caso di troncamento di X durante la SVD avremo  $X = \tilde{U}\tilde{\Sigma}\tilde{V}^*$ , ottenendo:

$$A \approx \bar{A} = (X' - BY)\tilde{V}\tilde{\Sigma}^{-1}\tilde{U}^*$$

*Equazione 2.19*

Con i passaggi sopra descritti riusciamo ad ottenere la matrice A, che ci permette di calcolare l'Equazione 2.16, quindi di conoscere lo stato futuro.

Nel caso in cui volessimo ottenere un modello maggiormente troncato, per avere maggiore efficienza a livello computazione, potremmo considerare una trasformazione dove  $\tilde{x} = U^*x$ , quindi si ottiene:

$$\tilde{A} = U^*(X' - BY)V\Sigma^{-1}$$

*Equazione 2.20*

$$\tilde{B} = U^*B$$

*Equazione 2.21*

Sia nel caso di A che di  $\tilde{A}$  possiamo considerare la decomposizione in autovalori  $\lambda_i$  ed autovettori  $w_i$ , dalla quale possiamo estrarre i modi dinamici tramite questa equazione:

$$\Phi = (X' - BY)V\Sigma^{-1}W$$

*Equazione 2.22*

2. Caso in cui B non è nota:

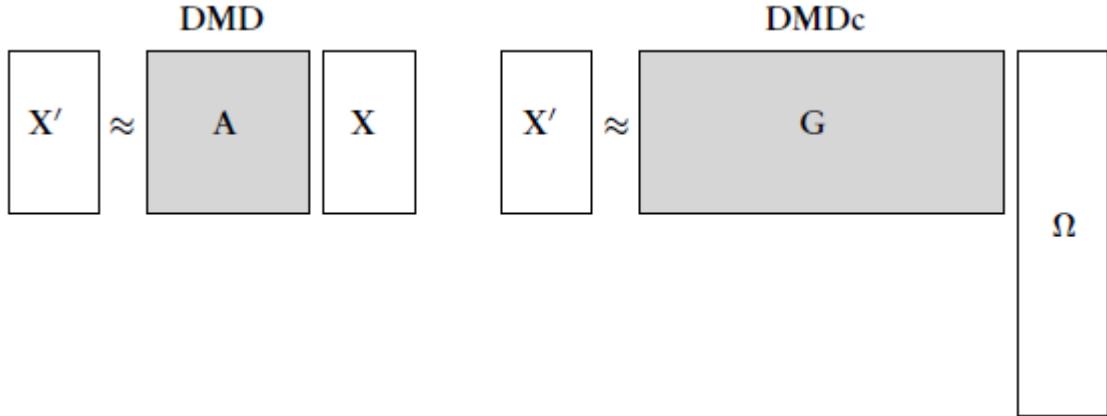
Nel caso in cui non si conosca la B, è possibile calcolarla a partire dalle serie temporali degli stati e le serie temporali degli input. Le nostre incognite, considerando l'equazione 2.2.2, adesso sono sia la A sia la B, ed il nostro obiettivo sarà, a questo punto, quello di stimare non solo quali sono le dinamiche interne del sistema, ma anche di capire come il controllo influisce sullo stato.

Partendo dall'Equazione 2.15 è possibile riformulare il problema in questo modo:

$$X' \approx [A \ B] \begin{bmatrix} X \\ \gamma \end{bmatrix} = G\Omega$$

*Equazione 2.23*

Si nota già da subito che l'equazione appena ottenuta, utilizzando gli operatori G e  $\Omega$ , non è dissimile dall'equazione con la quale andiamo a descrivere il DMD (Equazione 2.4), il principale cambiamento sta nelle dimensioni delle matrici, come possiamo denotare da questa figura:



*Figura 2.3*

A questo punto ci siamo sostanzialmente ricondotti alla risoluzione già vista per il DMD.

1. Isoliamo l'operatore G:

$$G = X'\Omega^\dagger$$

*Equazione 2.24*

$$[A \ B] = X' \begin{bmatrix} X \\ \gamma \end{bmatrix}^\dagger$$

*Equazione 2.25*

2. A questo punto possiamo applicare la SVD a  $\Omega$  troncando per aumentare l'efficienza computazionale:

$$G = X'\tilde{V}\widetilde{\Sigma}^{-1}\tilde{U}^*$$

*Equazione 2.26*

3. Possiamo approssimare la A e la B splittando la U in questo modo:

$$\tilde{U}^* = [\tilde{U}_1^* \ \tilde{U}_2^*]$$

*Equazione 2.27*

Con  $\tilde{U}_1^* \in \mathbb{R}^{n \times \tilde{r}}$  e  $\tilde{U}_2^* \in \mathbb{R}^{q \times \tilde{r}}$

Ottenendo:

$$[A, \ B] \approx [X' \tilde{V} \tilde{\Sigma}^{-1} \tilde{U}_1^*, \ X' \tilde{V} \tilde{\Sigma}^{-1} \tilde{U}_2^*]$$

*Equazione 2.28*

Anche in questo caso ci potrebbe essere la necessità di troncare maggiormente nel caso in cui l'ordine di grandezza di A e B fosse eccessivo.

Qui, cerchiamo ancora un modello ad ordini ridotti di rango  $r \neq n$ , e considerando la trasformazione  $\tilde{x} = U^*x$  si ottiene:

$$\begin{aligned}\tilde{A} &= \hat{U}^* \bar{A} \hat{U} = \hat{U}^* X' \tilde{V} \tilde{\Sigma}^{-1} \tilde{U}_1^* \hat{U} \\ \tilde{B} &= \hat{U}^* \bar{B} = \hat{U}^* X' \tilde{V} \tilde{\Sigma}^{-1} \tilde{U}_2^*\end{aligned}$$

*Equazione 2.29*

Con  $\tilde{A} \in \mathbb{R}^r$  e  $\tilde{B} \in \mathbb{R}^{(r \times l)}$

Otteniamo quindi l'equazione di ordine ridotto:

$$\tilde{x}_{k+1} = \tilde{A} \tilde{x}_k + \tilde{B} u_k$$

*Equazione 2.30*

E le dinamiche:

$$\Phi = X' \tilde{V} \tilde{\Sigma}^{-1} \tilde{U}_1^* \hat{U} W$$

*Equazione 2.31*

## Model Reduction

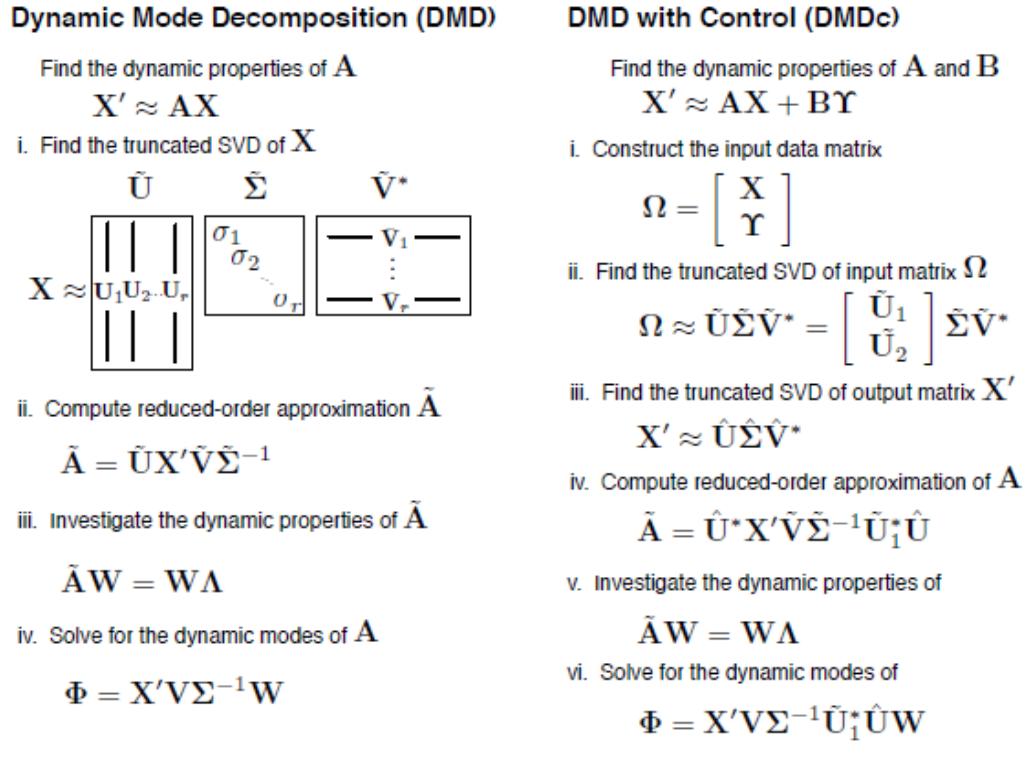


Figura 2.4

## Applications

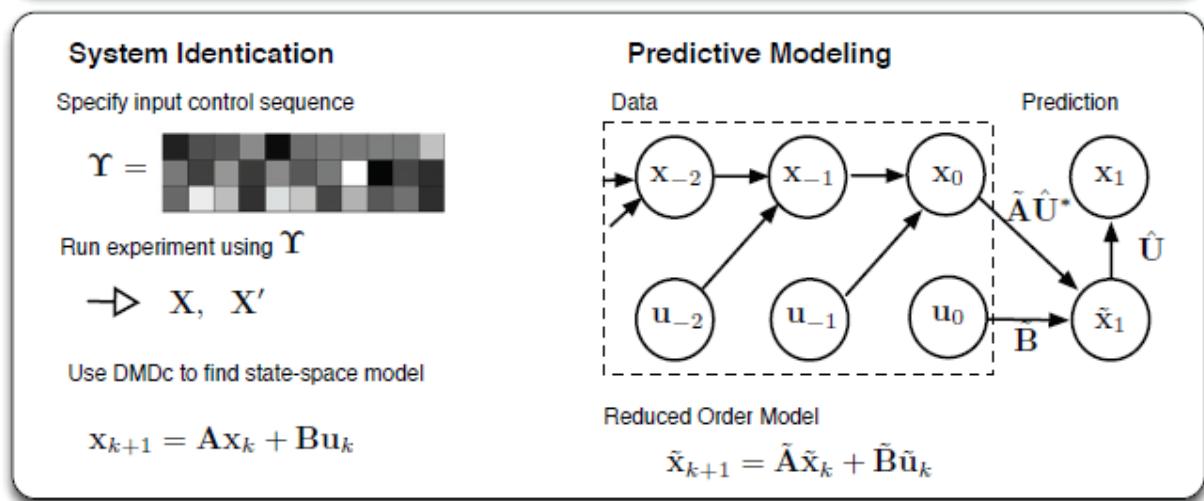


Figura 2.5

## 2.2.2 Algoritmo DMDc

Diamo una breve descrizione di come viene sviluppato l'algoritmo per il calcolo del DMDc.

1. Vengono caricate le serie temporali di  $X'$ ,  $X$  e  $Y$ . Le matrici di dati  $X$  e  $Y$  verranno utilizzate per costruire la matrice  $\Omega$ .
2. Viene calcolata la SVD di  $\Omega$ :

$$\Omega \approx \tilde{U} \tilde{\Sigma} \tilde{V}^* = \begin{bmatrix} \tilde{U}_1 \\ \tilde{U}_2 \end{bmatrix} \tilde{\Sigma} \tilde{V}^*$$

*Equazione 2.32*

3. Viene calcolata la SVD di  $X'$ :

$$X' \approx \hat{U} \hat{\Sigma} \hat{V}^*$$

*Equazione 2.33*

4. Viene calcolata l'approssimazione di  $G$ , cioè  $\tilde{A} \tilde{B}$ :

$$\begin{aligned} \tilde{A} &= \hat{U}^* X' \tilde{V} \tilde{\Sigma}^{-1} \tilde{U}_1^* \hat{U}, \\ \tilde{B} &= \hat{U}^* X' \tilde{V} \tilde{\Sigma}^{-1} \tilde{U}_2^*: \end{aligned}$$

*Equazione 2.34*

5. Viene calcolata la decomposizione di  $\tilde{A}$ :

$$\tilde{A}W = W\Lambda$$

*Equazione 2.35*

6. Vengono calcolati i modi dinamici:

$$\Phi = X' V \Sigma^{-1} \tilde{U}_1^* \hat{U} W$$

*Equazione 2.36*

### 2.2.3 Implementazione Python

Viene presentata una breve descrizione di come viene eseguito l'algoritmo DMDc su un selezionato dataset (algoritmo in parte riutilizzato per l'implementazione del mrDMDc)

Per selezionare un dataset è necessario modificare la variabile di percorso per il caricamento dei dati sperimentali di train e test che dovranno essere passati al DMDc:

```
#insert path in which to load .mat files
#load the training experimental file
path_for_load_experimental_train = r'C:\Use
#load the test experimental file
path_for_load_experimental_test = None#'C:\\\\

#insert path in which to save the csv files
#save the training reconstructed file
path_for_save_reconstructed_train = r'C:\Us
#save the test reconstructed file
path_for_save_reconstructed_test = None#'C:\\\\
```

Codice 4.1

Per selezionare uno stato da vedere è necessario modificare il parametro ‘*column\_to\_show*’. Lo stato che viene presentato è pari a ‘*column\_to\_show*’ + 1.

```
#this parameter is used to decide which column to show
column_to_show = 0
```

Codice 4.2

Per decidere se eseguire un troncamento per Y e  $\Omega$  (Equazione 2.32), è necessario modificare il parametro ‘*svd\_rank\_set*’.

```
'''param svd_rank: the rank for the truncation; If 0, the method computes the optimal rank and uses it for truncation;
if positive interger, the method uses the argument for the truncation; if float between 0 and 1, the rank is the number
of the biggest singular values that are needed to reach the 'energy' specified by `svd_rank`; if -1, the method does
not compute truncation.'''
svd_rank_set = -1
```

Codice 4.3

Per tutte le prove è stato impostato il parametro *svd rank* = −1, questo significa che per l'identificazione del modello DMDc sul dataset non vi è stato alcun troncamento per Y o  $\Omega$ .

A questo punto possiamo presentare la modalità con la quale viene calcolato il DMDc sul dataset. Viene utilizzata la libreria PyDMD[7I] con la classe DMDc e due metodi, ‘*fit*’ e ‘*reconstructed\_data*’, gli stessi che sono stati descritti nella teoria del mrDMDc:

```
U_train = U_train[:,1:]
dmdc = DMDc(svd_rank = svd_rank_set)
dmdc.fit(D_train,U_train)
DMDc_train_reconstructed = dmdc.reconstructed_data()
Codice 4.4
```

Notiamo come al metodo *fit* passiamo un campione in avanti di  $U_{train}$  rispetto alla  $D_{train}$ , questo perché il metodo vuole che venga un passato un campione in avanti di  $U_{train}$ . Il motivo di ciò è che per la ricostruzione chiaramente il controllo non interviene nelle condizioni iniziali del sistema (considerato come primo campione) ma dal campione successivo in poi.

La matrice ricostruita è la ‘*DMDc\_train\_reconstructed*’, risultato del metodo *reconstructed\_data*.

Per il test viene utilizzato nuovamente il metodo ‘*reconstructed\_data*’, passando in ingresso una  $U$  (serie temporale di ingressi) che ci servirà per capire se il modello calcolato dal DMDc è adeguato, poiché verrà fatto il confronto tra:

- I dati di test ottenuti in maniera sperimentale con il medesimo ingresso.
- I dati ricostruiti utilizzando il modello calcolato dal DMDc.

Di seguito il codice che ci permette di eseguire il test:

```
#reconstruction test
DMDc_test_reconstructed = dmdc.reconstructed_data(U_test)[0]
Codice 4.5
```

Da attenzionare che per la ricostruzione dei dati di test è stato necessario passare una matrice di controllo  $U_{test}$  che avesse lo stesso numero di colonne della  $U_{train}$ .

### 2.3 Multi Resolution Dynamic Mode Decomposition

L'algoritmo mrDMD si basa sulla possibilità di distinguere modi dinamici lenti (slow) e veloci (fast) di un dataset tramite analisi DMD. Questa capacità permette di eliminare in modo ricorsivo il contenuto a bassa frequenza da una serie temporale di snapshot degli stati. Il numero di snapshot dal quale si calcolano i modi deve essere tale da garantire che, i modi per l'appunto, siano sia slow sia fast.

Per tale motivo il numero di snapshot viene scelto inizialmente pari al dataset, e quelle che si trovano sono le stesse dinamiche che troveremo andando ad eseguire una singola volta il DMD sull'intero dataset. Nella prima iterazione abbiamo dunque ottenuto  $m$  dinamiche di cui  $m_1$  sono dinamiche slow e  $m_2$  sono dinamiche fast. A questo punto andiamo a rimuovere le dinamiche  $m_1$  dal dataset e facciamo uno split in due parti, nelle quali in ognuna eseguiamo lo stesso procedimento

Nella seconda iterazione considereremo metà dataset, troveremo i modi dinamici tramite analisi DMD, troviamo le dinamiche slow e fast, ed andiamo a rimuovere quelle slow, per poi di nuovo splittare il dataset e rieseguire l'algoritmo.

### 2.3.1 Descrizione matematica

Come abbiamo scritto in precedenza, l'algoritmo mrDMD elimina ricorsivamente le dinamiche slow trovate dall'analisi DMD della serie temporale degli stati. Dunque, il primo passo sarà quello di analizzare le dinamiche DMD riguardanti la prima iterazione del codice:

### *Equazione 2.37*

Nella prima riga sono presenti le  $m$  dinamiche dell'analisi DMD effettuata su una serie temporale di stati composta da  $M$  snapshot. Nella seconda riga vengono divise le dinamiche in slow ( $m_1$ ) e fast ( $m_2$ ), e le dinamiche slow vengono rimosse dal dataset.

A questo punto si esegue un'ulteriore analisi DMD solo sulle dinamiche fast e si ottiene:

$$X_{m/2} = \sum_{k=m_1+1}^m b_k(0) \phi_k^{(1)} \exp(\omega_k t)$$

### *Equazione 2.38*

Come prima posso scomporre le dinamiche in slow e fast, ottenendo:

$$\mathbf{X}_{m/2} = \mathbf{X}_{m/2}^{(1)} + \mathbf{X}_{m/2}^{(2)}$$

### *Equazione 2.39*

La prima riga contiene i modi slow e la seconda quelli fast; vengono rimossi i modi slow e si analizza tramite DMD i modi fast.

Questo processo continua ricorsivamente fino a quando non si raggiunge il livello desiderato, andando a rimuovere di volta in volta componenti a bassa frequenza, arrivando ad ottenere un risultato pari al seguente:

$$\mathbf{x}_{\text{mrDMD}}(t) = \sum_{k=1}^{m_1} b_k^{(1)} \phi_k^{(1)} \exp(\omega_k^{(1)} t) + \sum_{k=1}^{m_2} b_k^{(2)} \phi_k^{(2)} \exp(\omega_k^{(2)} t) + \sum_{k=1}^{m_3} b_k^{(3)} \phi_k^{(3)} \exp(\omega_k^{(3)} t) + \dots$$

Equazione 2.40

Gli  $m_k$  indicano il numero di modi slow che vengono trovati ad ogni iterazione del codice, con il relativo calcolo delle dinamica per ogni modo slow identificato. In particolare,  $\phi_k^{(k)}$  e  $\omega_k^{(k)}$  rappresentano i modi dinamici e gli autovalori DMD al  $k$ -esimo livello di decomposizione, e  $b_k^{(k)}$  sono le proiezioni iniziali dei dati sull'intervallo di tempo considerato.

A questo punto possiamo definire in maniera più chiara il risultato dell'Equazione 2.40, ci basta considerare i seguenti parametri:

- Livello di decomposizione:  $L$
- Numero di intervalli temporali per ogni livello:  $J$
- Numero di modi:  $m_l$

Gli indici per questi parametri sono i seguenti:

- $l = 1, 2, \dots, L$  serve per indicare il numero di livelli di decomposizione
- $j = 1, 2, \dots, J$  serve per indicare il numero degli intervalli di tempo per livello  $J = 2^{(l-1)}$
- $k = 1, 2, \dots, m_l$  serve per indicare il numero di modi estratti al livello  $l$

La soluzione formale richiede la seguente funzione indicatrice:

$$f^{\ell,j}(t) = \begin{cases} 1, & t \in [t_j, t_{j+1}] \\ 0, & \text{elsewhere} \end{cases}, \quad \text{with } j = 1, 2, \dots, J \text{ and } J = 2^{(l-1)},$$

Equazione 2.41

A questo punto è possibile scrivere la soluzione mrDMD come segue:

$$\mathbf{x}_{\text{mrDMD}}(t) = \sum_{\ell=1}^L \sum_{j=1}^J \sum_{k=1}^{m_L} f^{\ell,j}(t) b_k^{(\ell,j)} \phi_k^{(\ell,j)} \exp(\omega_k^{(\ell,j)} t)$$

Equazione 2.42

L'utilità di utilizzare mrDMD rispetto a DMD sta nel fatto che grazie alle iterazioni del codice vi è la possibilità di sfruttare varie dinamiche DMD per rappresentare caratteristiche multi-resolution. Questo significa che la SVD non viene calcolata per l'intero dataset una volta sola, ma viene calcolata ogni qual volta si itera il codice, adattando le dinamiche all'intervallo del dataset considerato.

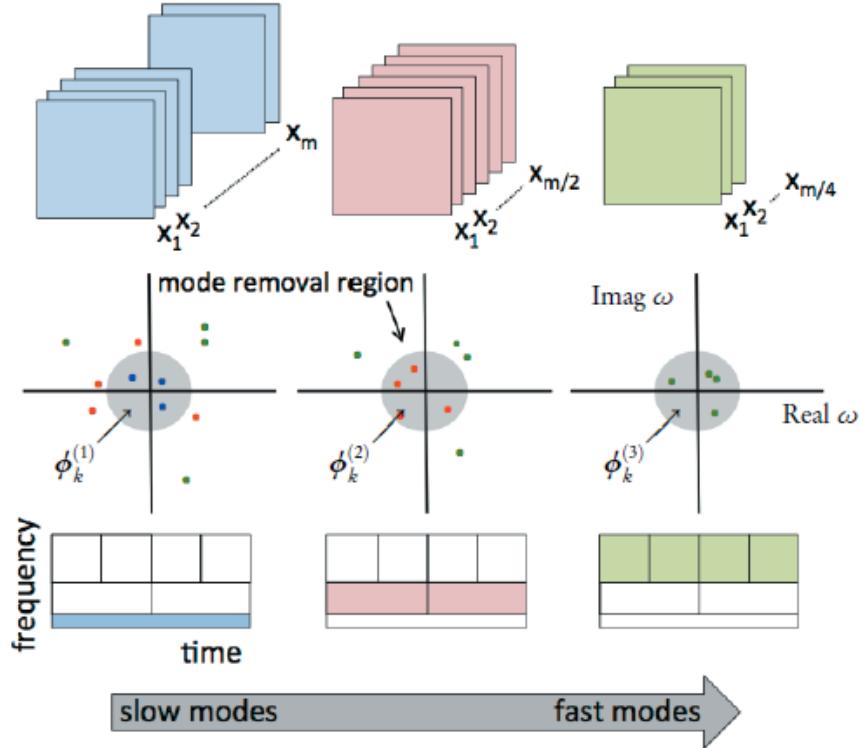


Figura 2.6

La figura mostra come avviene la decomposizione tramite mrDMD. In questo caso i livelli identificati sono tre: la scala più lenta è quella in blu dove si considera tutta la finestra temporale degli stati, la scala intermedia è quella in rosso dove si considerano due finestre temporali e la scala veloce è quella in verde dove si considerano quattro finestre temporali (metà delle due precedenti). In ogni scala vengono estratte le dinamiche slow.

Per avere una migliore efficienza computazionale possiamo considerare gli  $M$  snapshot, nel quale eseguire in maniera ricorsiva l'mrDMD, come un sotto campionamento della serie temporale degli stati; questo significa che l'analisi DMD non la facciamo lungo tutto il dataset, ma solo su dei campioni prelevati da esso. Il numero di campioni deve comunque essere tale da comprendere la presenza delle dinamiche slow e fast.

### 2.3.2 Algoritmo mrDMD

Come abbiamo detto in precedenza l'algoritmo mrDMD performa l'algoritmo DMD in una finestra di campioni del dataset.

Vediamo passo-passo come viene eseguito l'algoritmo:

1. Campionamento del dataset.
2. Calcolo dei modi tramite analisi DMD dei dati sotto campionati, con possibilità di troncamento.
3. Estrazione dei modi slow, con calcolo delle relative dinamiche.
4. Sottrazione tra il dataset iniziale e le dinamiche slow calcolate al punto 3.
5. Split dei dati a metà, nella quale vengono rieseguiti i passaggi precedenti per entrambe le parti.

Chiaramente la procedura per eseguire il punto 5 è ricorsiva, quindi, l'algoritmo richiamerà sé stesso fino ad un determinato livello, oppure fino a quando vi è la possibilità di dividere il segnale e di campionare.

Per il troncamento, discusso nel punto 2, è possibile utilizzare la *Singular Value Hard Threshold* (SVHT)[5] che calcolerà in maniera ottimale la riduzione del rango.

Per il calcolo delle dinamiche invece andiamo ad utilizzare il calcolo di b ottimale (spDMD)[6].

## 2.4 Multi Resolution Dynamic Mode Decomposition with Control

Dalle sperimentazioni eseguite, e da quanto descritto in precedenza, si può facilmente comprendere come l'algoritmo mrDMD sia un'evoluzione del DMD, poiché chiaramente ne migliora le prestazioni.

L'evoluzione del DMDc sarà quindi un nuovo algoritmo che include un'analisi multi-resolution: mrDMDc.

L'algoritmo di per sé performa come un mrDMD, con l'aggiunta però dell'analisi degli input.

Per quanto riguarda l'implementazione dell'mrDMDc sono state utilizzate parti dell'algoritmo mrDMD e parti dell'algoritmo DMDc. Difatto l'algoritmo mrDMDc non è altro che una fusione tra mrDMD e DMDc, con opportune modifiche.

### 2.4.1 Algoritmo mrDMDc

- Input:
    - $D^{n \times m}$  con  $n$  stati ed  $m$  snapshot (serie di snapshot degli stati)
    - $U^{p \times m}$  con  $p$  ingressi ed  $m$  snapshot (serie di snapshot degli ingressi)
    - level = 0 (parametro che tiene traccia dei livelli di ricorsione dell'algoritmo)
    - bin num = 0 (parametro che tiene traccia del numero di finestre temporali)
    - offset = 0 (parametro che tiene traccia dello scostamento tra finestre temporali)
    - max cycles = 10 (parametro che indica il numero massimo di oscillazioni di un modo dinamico per essere considerato slow nella finestra temporale (variabile *bin size*) analizzata dall'algoritmo nell'attuale iterazione)
  - Output:
    - Lista contenente oggetti appartenenti alla classe DMDc, chiamata nodi, che serve per la ricostruzione per livelli, di cui discuteremo successivamente.
1. Campionamento del dataset:
    - Definizione ed inizializzazione del parametro:
$$nyq = 4 * \text{max cycles}$$
Equazione 2.44
    - indica la frequenza di campionamento con la quale sotto campioniamo la finestra temporale selezionata.
    - Definizione ed inizializzazione del parametro *bin size* posto pari al numero di colonne di  $D$ , indica la finestra temporale analizzata dall'algoritmo in quella iterazione.

- Definizione ed inizializzazione del parametro:

$$step = \frac{bin\ size}{nyq} \text{ (divisione intera)}$$

Equazione 2.45

indica ogni quanto bisogna prendere un campione di D, in questo caso ogni step. Si esce dall'algoritmo se  $bin\ size < nyq$ , e quindi quando  $step = 1$ .

Presentazione di un grafico che mostra l'andamento tipico dei parametri sopra elencati:

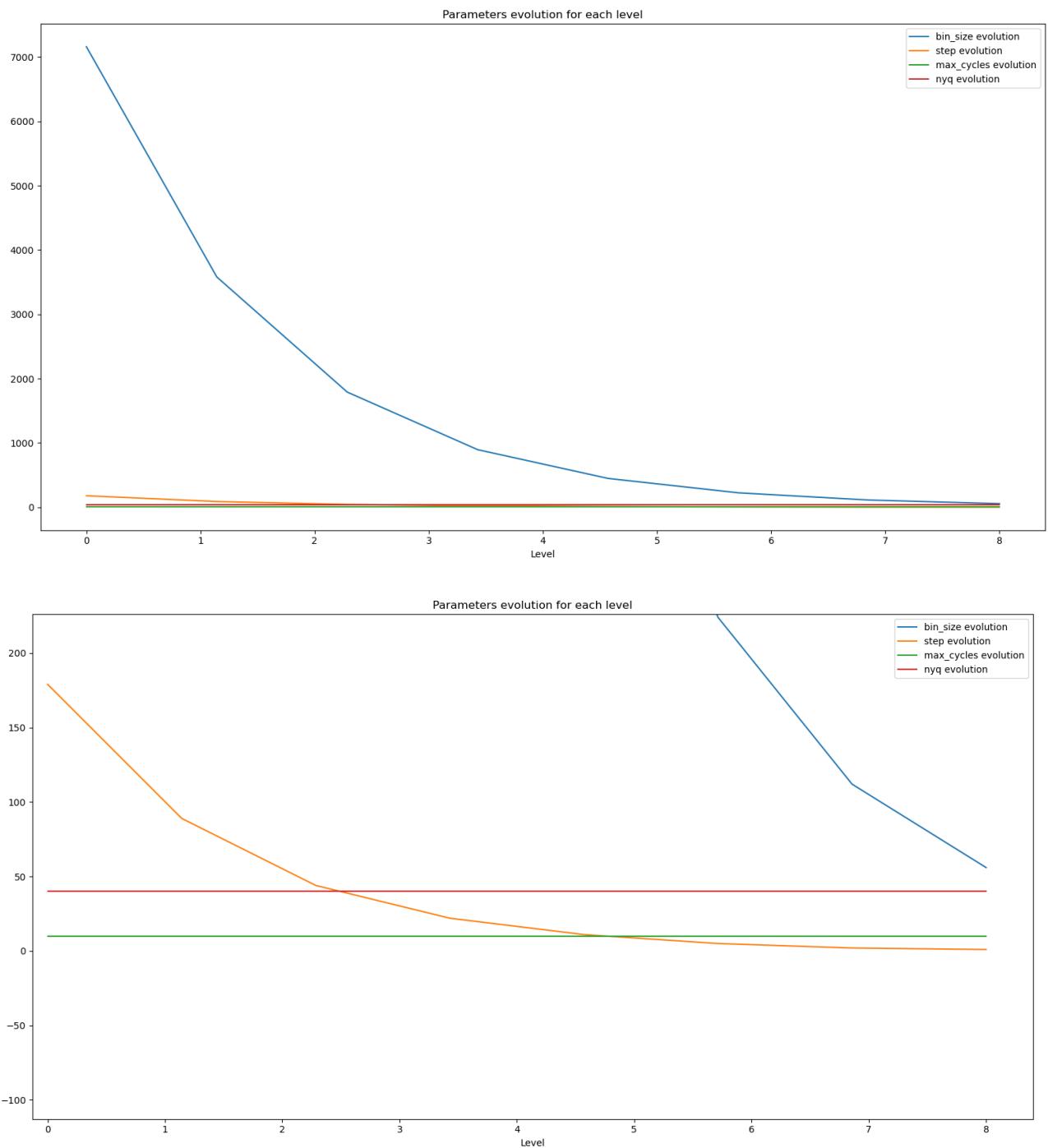


Figura 2.7

- Adesso è possibile sotto campionare D ed U prendendo un campione ogni step. Definiamo dunque due variabili  $D_{sub}$  ed  $U_{sub}$  che contengono le matrici sotto campionate lungo l'asse temporale.

## 2. Calcolo del modello tramite analisi DMDc:

- Inizializzazione di un oggetto appartenente alla classe DMDc [7], nella quale passiamo un parametro, definito *svd rank*, che ci permette di decidere se troncare  $\Omega$  e Y (determinati successivamente all'interno del metodo *fit*) oppure no.  
*svd rank* può assumere tre valori:
  - Un qualsiasi valore intero per troncare a nostro piacimento
  - 0: per troncare in maniera ottimale secondo hard threshold[5]
  - -1 per non troncare.
- Dichiariamo anche un array di nodi che ci servirà per salvare i vari oggetti DMDc (che vedremo contengono la matrice ricostruita per quella determinata finestra temporale) che creiamo in tutte le iterazioni.
- Salviamo dei parametri utili nell'oggetto DMDc.
- Applichiamo all'oggetto dichiarato il metodo *fit* a cui passiamo  $D_{sub}$  ed  $U_{sub}$  (per la  $U_{sub}$  dobbiamo partire un campione avanti rispetto ad  $D_{sub}$ , perché per la previsione ad 1 passo  $k + 1_{esimo}$ , si utilizzano gli ingressi al passo  $k + 1_{esimo}$ , ipotizzando di poterli misurare all'istante di interesse). Ciò serve per l'identificazione del sistema discretizzato a passo step, determinando:
  - Autovalori  $\Lambda^r$  con  $r = n$  in caso di *svd rank* = -1 (nessun troncamento) o  $r \leq n$  in caso di *svd rank* ≠ -1 (troncando).
  - Modi  $\Phi^{n \times r}$  con  $r \leq n$ .
  - Evoluzione temporale  $\Psi^{r \times m}$  con  $r \leq n$ .
  - $\tilde{A}^{r \times r}$  con  $r \leq n$ .
  - $\tilde{B}^{r \times p}$  e  $B^{n \times p}$  con  $r \leq n$
- Possiamo estrarre gli autovalori  $\mu$  e i modi  $\phi$  trovati dall'oggetto DMDc alla quale abbiamo applicato il metodo *fit*. Questi saranno rappresentativi del sistema discreto a passo step.

## 3. Filtrazione delle slow feature:

- Calcoliamo il raggio per la filtrazione, che sarà dato da questa formula:

$$\rho = \lambda * \frac{\max cycles}{bin size}$$

*Equazione 2.46*

Dove  $\lambda$  è un parametro da modificare a proprio piacimento in base alla percentuale di filtrazione che si vorrà ottenere.  $\rho$  verrà salvato all'interno dell'oggetto DMDc.

- A questo punto tramite questo confronto:

$$\left\| \frac{\ln(\mu)}{2 * \pi * step} \right\| \leq \rho$$

*Equazione 2.47*

Determiniamo quali sono gli autovalori slow, in quanto quelli minori o uguali di  $\rho$  saranno considerati slow, tutti gli altri saranno considerati fast.

- Estraiamo i modi e gli autovalori slow e li teniamo in due variabili chiamate  $\mu_{slow}$  e  $\phi_{slow}$ , per le quali calcoliamo le dinamiche slow mentre per le dinamiche fast consideriamo la loro ricostruzione nulla.

#### 4. Calcolo delle dinamiche slow:

- Calcoliamo la matrice A del sistema discreto a passo step:

$$A_{step} = \Phi_{slow} * M_{slow} * (\Phi_{slow})^\dagger$$

*Equazione 2.48*

Dove:

- $\Phi^{n \times f}$  con  $f \leq r$ .
- $M^f$  con  $f \leq r$ .

Da queste segue che  $A_{step}$  ha dimensioni  $n \times n$ .

- La matrice  $B_{step}$  è stata calcolata dal metodo *fit*, con dimensioni  $n \times p$ .
- Definizione di una matrice  $C^{n \times n}$  tale che  $n$  sia pari al numero di righe di  $A_{step}$ , mentre  $m$  sia pari al numero di righe di  $B_{step}$ . La prima colonna è posta ad 1, il resto 0.
- Definizione di una matrice  $D^{n \times p}$  tale che  $n$  sia pari al numero di righe di  $A_{step}$ , mentre  $m$  sia pari al numero di colonne di  $B_{step}$ . La matrice è tutta nulla.
- Con queste matrici definiamo un sistema discreto con passo step, che dobbiamo ricondurre ad un sistema con passo 1 per la successiva ricostruzione. Il sistema discreto viene definito utilizzando la variabile *sys disc step*. La libreria con il quale dichiariamo il sistema è la libreria Harold[8].
- Il sistema discreto a passo step per essere ricondotto a passo 1 deve essere prima ricondotto nel continuo; quindi, utilizziamo il metodo *undescretize* della libreria Harold, e salviamo il sistema continuo in una variabile definita *sys cont*. Estraiamo anche gli autovalori della matrice  $A_{cont}$  per vedere i poli nel continuo.

- Per visualizzare la mappa poli zeri di ogni nodo (quindi per ogni ricorsione del codice) utilizziamo un'altra libreria nominata Control [9], in quanto all'interno della libreria Harold non è presente questo metodo.
- Con la libreria Harold facciamo una nuova conversione dal sistema continuo (*sys cont*) al sistema discreto con passo 1. Il sistema discreto con passo 1 lo nominiamo *sys disc 1*.
- Da questo sistema estraiamo le matrici A e B che chiamiamo  $A_{disc\ 1}$  e  $B_{disc\ 1}$  a questo punto saranno calcolate con passo 1. Dalla matrice  $A_{disc\ 1}$  possiamo calcolare gli autovalori che avranno passo 1.

## 5. Ricostruzione per ogni nodo:

Per la ricostruzione sono stati seguiti due approcci:

1. Ricostruzione con le matrici del sistema discreto a passo step ( $A_{disc\ step}$   $B_{disc\ step}$ ).
2. Ricostruzione con le matrici del sistema discreto a passo step ( $A_{disc\ 1}$   $B_{disc\ 1}$ ).

I passaggi per entrambe le modalità sono i seguenti:

- Definiamo la variabile nella quale effettuiamo la ricostruzione nominandola  $D\ dmdc$  che poniamo uguale alla prima colonna di D che ci rappresenta le condizioni iniziali per il sistema. La dichiariamo come una lista che andremo a riempire al prossimo passaggio.
- Dichiariamo un indice  $i$ , con il quale ci andiamo a scorrere la lista  $D\ dmdc$  e la matrice di controllo  $U$  (preleviamo una colonna di U ad iterazione), andando a calcolare per ogni iterazione un parametro definito come  $x_{i+1}$ :

$$x_{i+1} = A_{disc\ 1\ o\ step} * D\ dmdc_i + B_{disc\ 1\ o\ step} * U_{i+1}$$

*Equazione 2.49*

Il valore  $x_{i+1}$  viene inserito nella lista  $D\ dmdc$ . Perciò ogni volta andiamo ad eseguire una nuova iterazione andiamo a considerare l'ultimo valore inserito  $x_i$  per calcolare il nuovo  $x_{i+1}$ . Finito lo scorrimento trasformiamo la lista in array omonimo.

- L'array  $D\ dmdc$  a questo punto indica la matrice ricostruita per questo nodo (nodo che opera in una certa finestra temporale). A questo punto salviamo all'interno dell'oggetto DMDc questa matrice e la andiamo anche a rimuovere da D.

## 6. Ricorsione:

- Se non si è superato il massimo numero dei livelli si esegue uno split di D e di U e si richiama l'algoritmo due volte con i seguenti parametri:
  - D split sinistra
  - U split sinistra
  - $level = level + 1$
  - $bin\ num = 2 * bin\ num$
  - $offset = offset + split$
  - $max\ levels = max\ levels$
  - $max\ cycles = max\ cycles$
- Dopo aver descritto l'algoritmo per la decomposizione, dalla quale abbiamo ottenuto dei nodi contenenti ognuno una matrice che ricostruisce una certa finestra temporale, possiamo passare alla ricostruzione dei livelli, e vedere come ha performato l'algoritmo di decomposizione.

- Dopo aver descritto l'algoritmo per la decomposizione, dalla quale abbiamo ottenuto dei nodi contenenti ognuno una matrice che ricostruisce una certa finestra temporale, possiamo passare alla ricostruzione dei livelli, e vedere come ha performato l'algoritmo di decomposizione.
- 7. Dichiariamo una matrice nominata  $D\ mrDMDc$  che poniamo inizialmente tutta a 0 poiché la andremo a sommare insieme alle ricostruzioni di ogni livello.
- 8. Facciamo un ciclo dove con un iteratore intero andiamo a selezionare tutti i livelli generati durante la decomposizione, da 0 fino al livello massimo (determinato tramite una funzione). All'interno del ciclo estraiamo i nodi del livello selezionato dall'iteratore e in una variabile nominata  $D\ mrDMDc\ level$  concateniamo le matrici ricostruite in orizzontale (aumentando il numero di colonne).
- 9. Ogni matrice  $D\ mrDMDc\ level$  calcolata ad ogni iterazione viene sommata alla variabile  $D\ mrDMDc$ .

## 2.4.2 Implementazione Python

Di seguito viene mostrata l'implementazione dell'algoritmo mrDMDc seguendo lo schema logico affrontato nella sezione ‘Algoritmo mrDMDc 2.4.1’.

- Input:

```
def mrdmdc(D, U, level=0, bin_num=0, offset=0, max_levels=20, max_cycles=20):
    """Compute the multi-resolution DMDc on the dataset `D`, returning a list of nodes
    in the hierarchy. Each node represents a particular "time bin" (window in time) at
    a particular "level" of the recursion (time scale). The node is an object consisting
    of the various data structures generated by the DMDc at its corresponding level and
    time bin. The `level`, `bin_num`, and `offset` parameters are for record keeping
    during the recursion and should not be modified.

    The `max_levels` parameter controls the maximum number of levels. The `max_cycles`
    parameter controls the maximum number of mode oscillations in any given time scale
    that qualify as 'slow'."""
    pass
```

Codice 4.6

- Output:

```
nodes = mrdmdc(D_train, U_train)
```

Codice 4.7

1. Campionamento del dataset:

```
# 4 times nyquist limit to capture cycles
nyq = 4 * max_cycles

#bin_size is equal to number of columns of D
bin_size = D.shape[1]

#the condition for exit from algorithm
if (bin_size) < (nyq):
    return []

# extract subsamples, take a value every step for D and U
step = bin_size // nyq
D_sub_step = D[:,::(step)]
U_sub_step = U[:,::(step)]
```

Codice 4.8

## 2. Calcolo del modello tramite analisi DMDc:

```
#declaration of DMDc object, it will be used to calculate eigenvalues and modes,
#also keeps track of the algorithm iterations and allow to save the reconstructed matrix for each iteration
dmdc = DMDc(svd_rank=1)

#save the DMDc object into an array
nodes = [dmdc]

dmdc.level = level          # level of recursion
dmdc.bin_num = bin_num       # time bin number
dmdc.bin_size = bin_size     # time bin size
dmdc.start = offset          # starting index
dmdc.stop = offset + bin_size # stopping index
dmdc.step = step             # step size
dmdc.nyq = nyq

#fitting model from data and input passed to algorithm, take the snapshot at k+1
dmdc.fit(D_sub_step , U_sub_step[:,1:])

#extract eigenvalues and modes from DMDc object
mu = dmdc.eigs
Phi = dmdc.modes
```

*Codice 4.9*

## 3. Filtrazione delle slow feature:

```
#Slow filtration
rho = slow_feauture_scale*max_cycles / bin_size
dmdc.rho = rho

# consolidate slow eigenvalues (as boolean mask)
slow = (np.abs(np.log(mu) / (2 * pi * step))) <= rho

# number of slow modes
n = sum(slow)

#calculate the percentage of filtration 100 : mu = percentage_of_filtration : n
dmdc.percentage_of_filtration = 100 - ((100/len(mu)) * n)

# extract slow modes (perhaps empty)
mu_SLOW_step = mu[slow]
Phi_SLOW_step = Phi[:,slow]

#reassignments of A,B are made in case no SLOW features are found
#fast modes
D_dmdc_A = np.zeros([D.shape[0], D.shape[1]], dtype='complex')
D_dmdc_B = np.zeros([D.shape[0], D.shape[1]], dtype='complex')

D_dmdc = np.zeros([D.shape[0], D.shape[1]], dtype='complex')
```

*Codice 4.10*

#### 4. Calcolo delle dinamiche slow:

```

#if found slow mode...
if n > 0:

    '''Definition of StateSpace discrete system with dt = step'''
    #Calculate A with slow features, they were calcuted with dt = step
    A_disc_step = np.linalg.multi_dot([
        [Phi_SLOW_step, np.diag(mu_SLOW_step), np.linalg.pinv(Phi_SLOW_step)] 
    ])

    #Extract B from dmdc object, also calculate with dt = step by method .fit (when we call dmdc.fit)
    B_disc_step = dmdc.B

    #Considerate two matrices C and D
    #matrix C with a ones column and the rest with 0 to extract the first state (that is not delayed)
    C_disc = np.zeros([A_disc_step.shape[0],B_disc_step.shape[0]], dtype = 'complex')
    C_disc[:,0] = 1

    #matrix D with 0 because the inputs don't influence directly the exit (strictly own system)
    D_disc = np.zeros([A_disc_step.shape[0], B_disc_step.shape[1]], dtype = 'complex')

    #declaration of StateSpace system, helps to convert discrete system with dt = step in continuous and see eigenvalues in continuous
    sys_disc_step = harold.State(A_disc_step, B_disc_step , C_disc, D_disc, dt = (step))

'''Defintion of StateSpace continuous system'''
#to see eigenvalues in continuous used the method of the Harold library "undiscretize"
#that allow to pass from discrete system to continuous system
sys_cont = harold.undiscretize(sys_disc_step, method='tustin')

#extract matrices from continuous system
A_cont = sys_cont.a
B_cont = sys_cont.b
C_cont = sys_cont.c
D_cont = sys_cont.d

#calculate eigenvalues from A matrix
[dmdc.mu_SLOW_cont,eigenvectors] = eig(A_cont)

#to print pzmap (pole-zero map) about system,
#there is the necessity to use another library because Harold library don't support the pzmap method
#the library that allow to see pzmap is the "Control Library for Python" doc: https://python-control.readthedocs.io/en/0.9.4/index.html
sys_cont_control = ct.StateSpace(A_cont, B_cont, C_cont, D_cont)
#ct.pzmap(sys_cont_control, title= 'Pole Zero Map node: ' + str(number_node) + ' level: ' + str(level) + ' step: ' + str(step))
#plt.show()

'''Definition of StateSpace discrete system with dt = 1'''
#for the reconstruction there is a necessity to reconvert system in discrete state,
#but with dt = 1 because every sample must be rescaled from distance step to 1
sys_disc_1 = harold.discretize(sys_cont, dt = 1, method = 'tustin')

#extract matrices, they are used to reconstruct
A_disc_1 = sys_disc_1.a
B_disc_1 = sys_disc_1.b
C_disc_1 = sys_disc_1.c
D_disc_1 = sys_disc_1.d

#extraction of eigenvalues with dt = 1
[dmdc.mu_SLOW_1,eigenvectors] = eig(A_disc_1)

```

Codice 4.11

## 5. Ricostruzione per ogni nodo:

```
...
Now the algorithm have the matrices A and B to compute the reconstructed matrix for the node (DMDc object).
The reconstruction for one node starts from the initial conditions (declared as an array).
This array will become a matrix after the reconstruction code block
"""

#parameters of the reconstruction
D_dmdc = [D[:, 0]].copy()

D_dmdc_A = D_dmdc.copy()      #these are used for the test (work in progress)
D_dmdc_B = []

#number of rows expected after the reconstruction
expected_shape = D_dmdc[0].shape

#the reconstruction continue with the calculation about data and inputs like DMDC algorithm
#but with the difference that the reconstruction is with all inputs and not only subsampled inputs
#data[i] represents the i-th instant of the state data, u represents the i-th instant of the input data
```

Per la ricostruzione sono stati seguiti due approcci:

### 1. Ricostruzione con le matrici del sistema discreto a passo step ( $A_{disc\ step}$ $B_{disc\ step}$ ):

```
'''reconstruction with A and B with dt = step'''
if reconstruction == 1:
    for i, u in enumerate(U[:,1:]):
        d_succ = A_disc_step.dot(D_dmdc[i]) + B_disc_step.dot(u)

        if d_succ.shape != expected_shape:
            raise ValueError(
                f"Invalid shape: expected {expected_shape}, got {d_succ.shape}"
            )
        D_dmdc.append(d_succ)
        D_dmdc_A.append(A_disc_step.dot(D_dmdc[i]))
        D_dmdc_B.append(B_disc_step.dot(u))

D_dmdc = np.array(D_dmdc).T
D_dmdc_A = np.array(D_dmdc_A).T
D_dmdc_B = np.array(D_dmdc_B).T
```

Codice 4.12

### 2. Ricostruzione con le matrici del sistema discreto a passo step ( $A_{disc\ 1}$ $B_{disc\ 1}$ ):

```
'''Reconstruction with A and B with dt = 1'''
if reconstruction == 2:
    for i, u in enumerate(U[:,1:]):
        d_succ = ((A_disc_1.dot(D_dmdc[i]) + B_disc_1.dot(u)))

        if d_succ.shape != expected_shape:
            raise ValueError(
                f"Invalid shape: expected {expected_shape}, got {d_succ.shape}"
            )
        D_dmdc.append(d_succ)
        D_dmdc_A.append(A_disc_1.dot(D_dmdc[i]))
        D_dmdc_B.append(B_disc_1.dot(u))

D_dmdc = np.array(D_dmdc).T
D_dmdc_A = np.array(D_dmdc_A).T
D_dmdc_B = np.array(D_dmdc_B).T
```

Codice 4.13

Rimozione delle dinamiche slow e salvataggio della matrice  $D$   $dmdc$ :

```
# remove influence of slow modes
D = D - D_dmdc

#save matrix for the reconstruction
dmdc.dato = D_dmdc
```

*Codice 4.14*

## 6. Ricorsione:

```
#the code will iterate splitting the data and input, stopping if level = max_levels or return when bin_size < nyq
if level < max_levels:
    split = ceil(bin_size / 2) # where to split           ## ceil(x) approximate by excess
    nodes += mrdmdc(
        D[:, :split],
        U[:, :split],
        level=level+1,
        bin_num=2*bin_num,
        offset=offset,
        max_levels=max_levels,
        max_cycles=max_cycles,
    )
    nodes += mrdmdc(
        D[:, split:],
        U[:, split:],
        level=level+1,
        bin_num=2*bin_num+1,
        offset=offset+split,
        max_levels=max_levels,
        max_cycles=max_cycles,
    )
#With every iteration of the algorithm there is an addition of a node (that is a DMDc object) in an array
#At the end the algorithm return the array with nodes processed in all iterations
return nodes
```

*Codice 4.15*

Ricostruzione del dataset che comprende i punti 7, 8, 9:

```
#declare variable that filled up in the reconstruction
D_mrdmdc = np.zeros([D_train.shape[0], D_train.shape[1]], dtype = complex)
D_mrdmdc_A = np.zeros([D_train.shape[0], D_train.shape[1]], dtype = complex)
D_mrdmdc_B = np.zeros([D_train.shape[0], D_train.shape[1]], dtype = complex)

for level in range(0 , iteration_level(nodes) + 1):

    # extract and sort relevant nodes of single level
    nodes_level = [n for n in nodes if n.level == level]
    nodes_level = sorted(nodes_level, key=lambda n: n.bin_num)

    #save step and nyq values, they can be seen in the plots
    nyq = nodes_level[0].nyq
    step = nodes_level[0].step

    #horizontal stack of reconstructed matrices of nodes with same level (stack reconstructed snapshots in order)
    D_mrdmdc_level_reconstruction = np.hstack([n.dato for n in nodes_level])

    #sum every reconstructed level
    D_mrdmdc += D_mrdmdc_level_reconstruction
```

*Codice 4.16*

Per lo studio dell'algoritmo mrDMDc, come descritto, sono state implementate due possibili ricostruzioni:

- Una eseguita utilizzando le matrici A e B con passo  $dt = 1$ , quindi dopo aver eseguito il passaggio da discreto a passo step a continuo e di nuovo discreto a passo 1 (punto 4 Calcolo delle dinamiche slow)
- L'altra eseguita utilizzando le matrici A e B con passo  $dt = step$

Si noterà come le ricostruzioni con A e B a passo  $dt = 1$  generino dei segnali che hanno un'ampiezza molto elevata, e ciò è chiaramente un assurdo fisico.

Di seguito, considerando come caso di studio il dataset SRU, presentiamo un confronto tra la ricostruzione con A e B a passo  $dt = 1$  e con A e B a passo  $dt = step$ .

Nota: è necessario fare un appunto per il parametro *max cycles*, che, come è stato descritto in precedenza, esso rappresenta il numero massimo di oscillazioni di un modo dinamico per essere considerato slow nella finestra temporale selezionata. Questo parametro generalmente è selezionato in maniera casuale, cercando di trovare il valore che ne migliora le prestazioni a parità di *svd rank* e scala di filtrazione. Capita talvolta che selezionato un valore questo comporti la non convergenza della SVD per un determinato nodo, impedendo così la modellizzazione del sistema dinamico. Per tale motivo nell'applicazione dell'algoritmo ai vari casi di studio tale parametro varia.

# 3 Capitolo: Casi di studio

## 3.1 Dataset Industriale: Surful Recovery Unit (SRU)

Una volta generati e testati dei dati sintetici, bisognava fare lo step successivo, ovvero utilizzare un dataset industriale reale. Per l'estrazione di questo dataset ci siamo basati sul Sulfur Recovery Unit (SRU), ovvero un processo per estrarre dati da processi industriali.

L'articolo da cui prenderemo le informazioni sarà il seguente [11]. In particolare, in questo articolo sono stati sviluppati due SS (Soft Sensor) per una raffineria in Sicilia, Italia. In particolare, è stato utilizzato per l'estrazione di zolfo dai gas acidi, questo perché il recupero di zolfo in questi tipi di processi è un processo chiave perché permette di rimuovere gli inquinanti ambientali dai vapori generati dal gas acido che vengono rilasciati nell'atmosfera. L'SRU preso in considerazione è composto da quattro sotto-unità identiche (ovvero linee di zolfo) che lavorano in parallelo, ognuna in grado di estrarre zolfo dai gas acidi ad una velocità di 100 tonnellate al giorno, qui di seguito in figura uno schema a blocchi che ne descrive il processo:

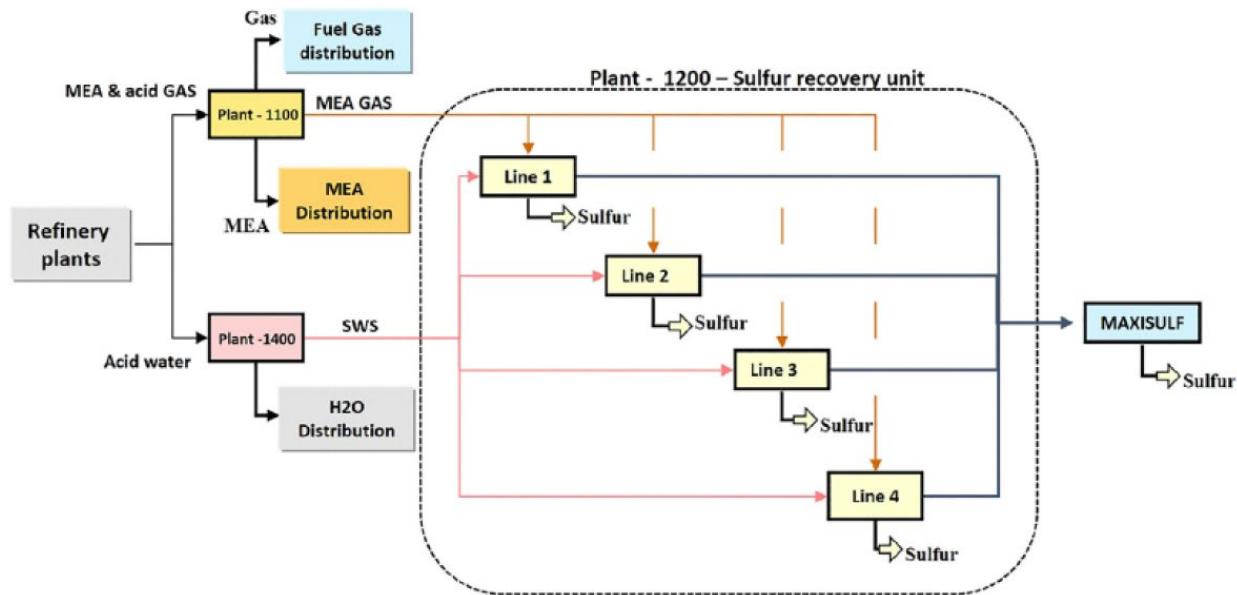


Figura 3.1

Ogni linea di zolfo riceve due tipi di gas acidi in ingresso: gas MEA, proveniente dalle centrali di lavaggio dei gas, e gas, SWS, ricco di H<sub>2</sub>S e NH<sub>3</sub>, proveniente dalla centrale di stripping dell'acqua acida. L'estrazione dello zolfo avviene in reattori, dove l'H<sub>2</sub>S viene sottoposto a una reazione di ossidazione parziale con l'aria. Il gas residuo viene alimentato nella centrale di Maxisulf per una fase di estrazione diversa. I gas che fuoriescono (gas di coda) dall'SRU contengono H<sub>2</sub>S e SO<sub>2</sub>, residui che devono essere controllati. L'aria, che fornisce ossigeno per la reazione, è essenziale per la conversione dei gas acidi e responsabile della composizione del gas di coda. Il processo è difficile da controllare perché un flusso d'aria eccessivo aumenta la concentrazione di SO<sub>2</sub> rispetto a H<sub>2</sub>S, mentre un flusso d'aria basso fa l'opposto. Attualmente, viene utilizzato un analizzatore online per misurare la concentrazione di H<sub>2</sub>S e SO<sub>2</sub> nel gas che fuoriesce da ciascuna linea di zolfo. Misura anche il valore [H<sub>2</sub>S] – 2[SO<sub>2</sub>] (dove le parentesi indicano la concentrazione) per monitorare le prestazioni del processo di conversione e controllare il rapporto aria-alimentazione nell'SRU per un'eccellente estrazione dello zolfo. Il valore desiderato di [H<sub>2</sub>S] – 2[SO<sub>2</sub>] è zero, il che indica l'assenza di questi

inquinanti nel gas che fuoriesce o che i reagenti sono in proporzione stechiometrica, il che è ottimale per la rimozione totale dello zolfo.

Il controllo viene migliorato da un algoritmo a ciclo chiuso che regola un flusso d'aria aggiuntivo (AIR MEA 2) in base all'analisi della composizione del gas di scarico. I gas acidi danneggiano i sensori, che richiedono una manutenzione frequente. Quando l'analizzatore è offline per la manutenzione, questo circuito di controllo non può funzionare e le prestazioni dell'SRU peggiorano. Pertanto, è necessaria una SS per sostituire l'analizzatore offline.

Inoltre, la SS fornisce una stima ridondante delle variabili di interesse per scopi di rilevamento dei guasti. L'SRU è spesso utilizzato come punto di riferimento per la progettazione di sistemi di stima. La maggior parte dei risultati disponibili in letteratura si applica alla linea 4 dello zolfo. In questo articolo, inizialmente sono stati considerati i dati acquisiti dalla linea 2. Per analizzare ulteriormente l'applicazione della metodologia proposta, vengono valutati anche i dati relativi alla linea 4 dello zolfo, confrontando i risultati ottenuti con alcune architetture alternative disponibili in letteratura. Per la riproducibilità degli approcci proposti, i dati di input e output normalizzati, utilizzati per lo sviluppo del SS della linea 2 dello zolfo, sono forniti nel materiale supplementare, mentre i dati relativi alla linea 4 sono disponibili in [12k]. Uno schema semplificato relativo a una singola linea di processo dell'SRU è rappresentato in figura:

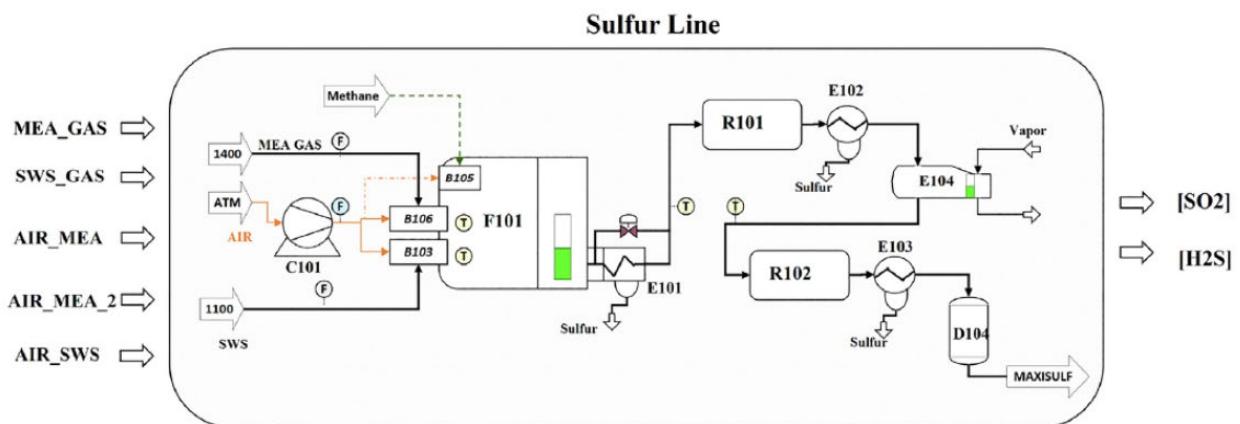


Figura 3.2

I cinque ingressi sono i gas MEA e SWS, i rispettivi flussi d'aria (AIR MEA) e (AIR SWS) e l'ulteriore flusso d'aria di ingresso (AIR MEA 2), che viene determinato utilizzando il sistema di controllo a ciclo chiuso basato sull'analisi del gas di scarto.

Nella linea considerata, vengono sviluppati due SS, uno per il SO<sub>2</sub> e l'altro per l'H<sub>2</sub>S. Oltre alla precisione della previsione, viene valutato anche [H<sub>2</sub>S] - 2[SO<sub>2</sub>]. La presenza di picchi nelle variabili rappresenta una condizione critica che, se prevista correttamente, consente un miglioramento del controllo del processo. Sulla base dei suggerimenti forniti dagli esperti del settore, la soglia critica nell'analisi dei picchi è il 30% del valore medio.

Come caso di studio, SRU è stato trattato considerando lo stato e gli ingressi ritardati 40 volte. Gli ingressi sono i seguenti:

1° ingresso	MEA GAS
2° ingresso	AIR MEA
3° ingresso	SWS GAS
4° ingresso	AIR SWS
5° ingresso	AIR MEA 2

## 3.2 Dati sintetici

Sono stati creati dei dati sintetici tramite uno script di Matlab, che essendo predeterminati ci hanno dato la possibilità di studiare le prestazioni dei vari modelli. In particolare, sono stati creati due dataset, il primo costituito da elementi reali, il secondo da elementi complessi.

Gli script sono disponibili in [10], e li riportiamo di seguito:

```
realpole=false;
To=0;
Tend=20;
samples=7159;
Ts=1;
n=10;

if realpole==true
    poles=[-3000 -2000 -250 -200 -50 -40 -0.1 -0.2 -0.01 -0.02];
else
    poles=[-8+80j -8-80j -2+20j -2-20j -0.5+10j -0.5-10j -0.2+2j -0.2-2j -0.01+0.5j -0.01-0.5j];
end
% A=diag(poles);
%B=ones(10,1);
%C=ones(1,10);
%D=0;
[A,B,C,D] = zp2ss([],poles,1)
%B = zeros(10,1)
csys=ss(A,B*10^8,C,D);
dcgain(csys)
pzmap(csys)

opt = c2dOptions('Method','Tustin');
dsys = c2d(csys,Ts,opt);
ev=eig(dsys.A)

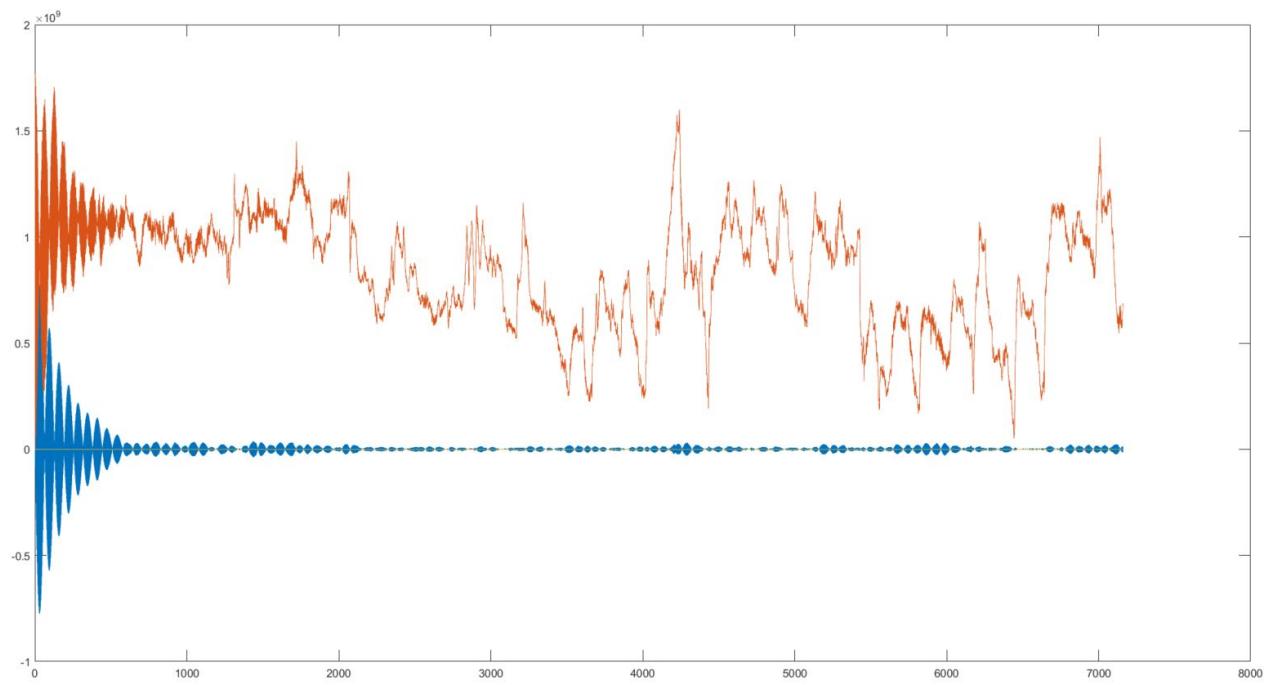
%[y,tOut,x]=initial(dsys,n*ones(10,1),samples);
load 'XU_DMDc.mat' U
u = U(161,:);
t = 1:size(u,2);
[y,tOut,x] = lsim(dsys,u,t,n*ones(10,1));
figure, plot(y)
figure, plot(u)
zplane(zpole);
figure, plot(tOut,x)
if realpole==true
    writematrix(x,'real_eig_timeseries.csv')
    x = x'; %%faccio la trasposta per avere le righe come stati e gli snapshot nelle colonne
    save real_eig_timeseries.mat x
else
    writematrix(x,'complex_eig_timeseries.csv')
    x = x'; %%faccio la trasposta per avere le righe come stati e gli snapshot nelle colonne
    save complex_eig_timeseries.mat x
end
```



Codice 3.1

La selezione del dataset complesso o reale dipende dalla variabile *realpole*, se questa viene impostata su *false* allora avremo selezionato il sistema a poli complessi, mentre se viene impostata su *true* allora avremo selezionato il sistema a poli reali.

Grafico dell’evoluzione temporale degli stati del sistema ad autovalori complessi:



Zoom:

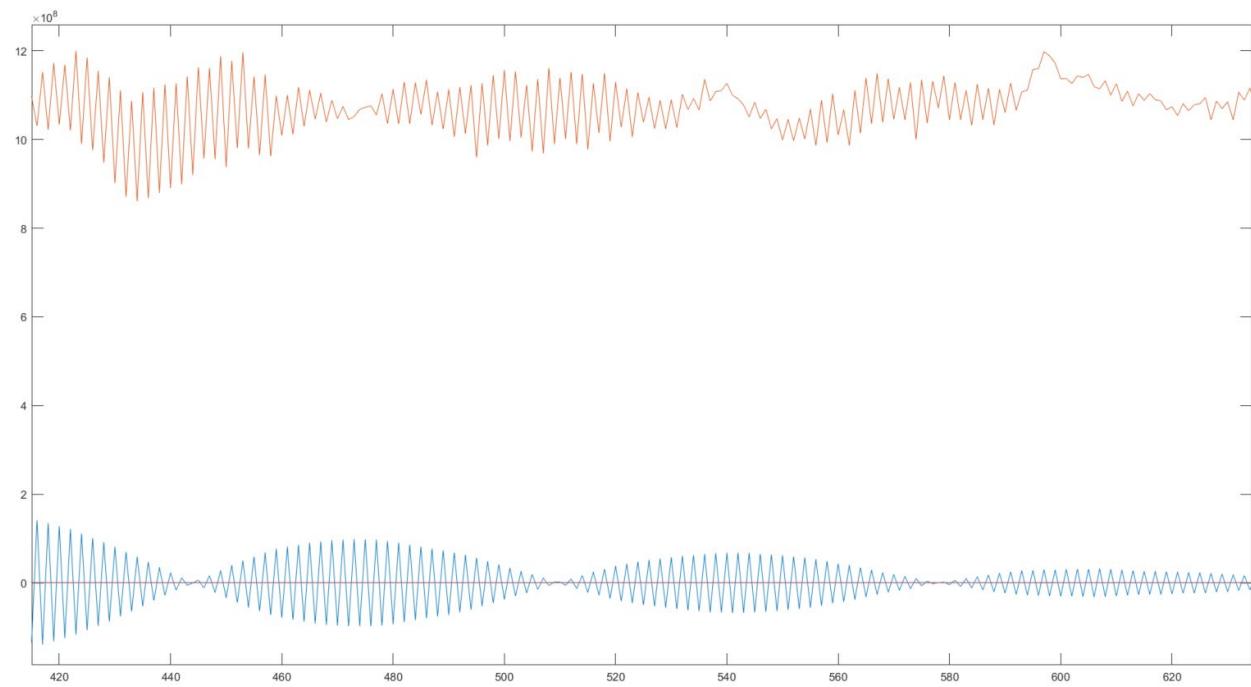


Figura 3.3

### 3.3 Vehicle to Grid (V2G)

Le informazioni riportate di seguito sono state prelevate dall'articolo [13].

Il Vehicle Energy Dataset (VED) [15] è un set di dati ad open - access contenente informazioni relative alla quantità di carburante e all'energia raccolta da 383 singoli veicoli ad Ann Arbor, Michigan, USA. Questo set di dati ad accesso aperto include record GPS dei percorsi dei veicoli e dati storici sul consumo di carburante, consumo di energia, velocità e utilizzo di energia ausiliaria. Il set di dati copre un'ampia gamma di veicoli:

- 264 motori a combustione interna (ICE),
- 92 veicoli elettrici ibridi (HEV),
- 27 veicoli elettrici ibridi plug-in/veicoli elettrici (PHEV/EV)

operando in condizioni reali da novembre 2017 a novembre 2018.

In questo lavoro, un insieme parziale delle funzionalità contenute nel file del set di dati VED è stato elaborato per ottenere la AAC ed altre informazioni aggregate per le applicazioni V2G. Nello specifico, sono state selezionate le seguenti funzionalità:

- DayNum,
- VehId,
- Trip,
- Timestamp,
- Latitudine,
- Longitudine

con tempo di campionamento di 3s.

Il Daynum rappresenta la differenza in giorni tra ciascun record e la data del 1° novembre 2017; VehID è un identificatore del veicolo; Trip è un identificatore per il viaggio del veicolo; Il timestamp rappresenta il tempo trascorso dall'inizio di ciascun viaggio; Latitudine e longitudine rappresenta l'informazione GPS espressa in gradi. Inoltre, viene fornito lo stato di carica della batteria HV (SoC[%]) per PHEV ed EV con un tempo di campionamento di 60 s.

I dati meteorologici sono stati estratti dal MeteoStat database utilizzando l'API Python [16]. Le informazioni estratte sono state le precipitazioni, la temperatura e la velocità del vento su base oraria per il periodo in esame.

La chiusura degli uffici statali in occasione delle festività statali è regolata dal Department of Civil Service Regulation del Michigan. Il Public Act 124 del 1865 è la legge governativa del Michigan che regola le festività statali [17], [18]. Giorni non lavorativi, come nella tabella riportata:

Non-Business Day	Date
Weekends	Saturdays and Sundays
Thanksgiving Day	23 Nov 2017
Day after Thanksgiving	24 Nov 2017
Christmas (Eve and Day)	24-25 Dec 2017
New Year (Eve and Day)	31 Dec 2017 - 1 Jan 2018
Martin Luther King, Jr. Day	15 Jan 2018
President's Day	19 Feb 2018
Memorial Day	28 May 2018
Juneteenth	19 Jun 2018
Independence Day	4 Jul 2018
Labor Day	3 Sept 2018
Columbus Day	8 Oct 2018
Veterans Day	12 Nov 2018

sono stati considerati, insieme alle informazioni del fine settimana, tali da ottenere un quadro completo delle festività da integrare nel modello.

Sono stati utilizzati i viaggi e la mappatura delle fermate per creare un dataset costituito da una serie temporale di dati con un intervallo di 30 minuti ad ogni campione relativo all' $hub_1$  V2G da utilizzare per la previsione dei dati. Il  $SoC_v$  reale o simulato viene utilizzato per determinare la capacità disponibile di un veicolo di AC (Available Capacity). È definita come la capacità di ciascun veicolo di fornire energia alla rete in un periodo di mezz'ora (hh) dove BC è la capacità della batteria. Viene considerata la seguente equazione per i veicoli disponibili:

$$AC_v^{hh} = \text{Max}(SoC_v^{hh-1} - SoC_{min}, 0) * BC$$

*Equazione 3.1*

L'aggregazione nello spazio si riferisce al presupposto che i veicoli parcheggiati entro un raggio specifico  $r$  rispetto ad un hub V2G selezionato si collegherebbero ad esso. Un veicolo parcheggiato entro un raggio  $r$  dall'hub in un intervallo di 30 minuti e rispettando il requisito  $SoC_{min}$ , è considerato disponibile ( $av^r$ ) a condividere energia nel sistema V2G in quell'intervallo.

La caratteristica da prevedere è la Available Aggregated Capacity ( $ACC_{hub^r}^{hh}$ ) in un intervallo di mezz'ora ed a distanza  $r$  dall'hub V2G. L'equazione per la previsione è la seguente:

$$ACC_{hub^r}^{hh} = \sum_{av^r} AC_v^{hh}$$

*Equazione 3.2*

Ulteriori serie temporali aggregate considerate sono le seguenti:

- Distanza media dall'hub ( $MHD_{hub^c}^{hh}$ ):  
la distanza media dal centro hub C di tutti i veicoli parcheggiati nel globale area di interesse nei dati VED.
- Intera durata arresto ( $SDI_{hub^r}^{hh}$ ):  
la somma del parcheggio sosta dei mezzi disponibili nell'arco dei 30 minuti intervallo.

Come caso di studio, V2G è stato trattato considerando lo stato e gli ingressi ritardati per un giorno intero (ove specificata la presenza dei ritardi). I dati sono stati divisi in train e test al 50%, per ogni mese 2 settimane per il train e 2 settimane per il test.

# 4 Capitolo: Risultati

In quest'ultimo capitolo andremo ad analizzare i risultati che si ottengono applicando i vari algoritmi ai dataset discussi nel capitolo precedente.

All'interno del GitHub[12] dedicato a questo lavoro sono presenti tutti i dataset ed i metodi che di seguito vengono utilizzati.

## 4.1 Key Performance Indicator

Prima di procedere con la presentazione dei risultati, una breve descrizione di quali KPI (Key Performance Indicators) sono stati utilizzati:

- MSE (Mean Squared Error):

$$\text{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2$$

*Equazione 4.1*

- MAPE (Mean Absolute Percentage Error):

$$\text{MAPE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} \frac{|y_i - \hat{y}_i|}{\max(\epsilon, |y_i|)}$$

*Equazione 4.2*

Nota:  $\epsilon$  è un numero arbitrario piccolo ma strettamente positivo per evitare risultati indefiniti quando  $y$  è zero.

- MAE (Mean Absolute Error):

$$\text{MAE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} |y_i - \hat{y}_i|$$

*Equazione 4.3*

- RMSE (Root Mean Squared Error):

$$\text{RMSE}(y, \hat{y}) = \sqrt{\text{MSE}(y, \hat{y})}$$

*Equazione 4.4*

- R2:

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

*Equazione 4.5*

Per l'elaborazione dei KPI è stata impiegata la libreria Scikit-learn [14].

Si sottolinea che il calcolo dei KPI è stato eseguito confrontando tutta la matrice sperimentale con tutta la matrice ricostruita, e non per uno stato solo, come si può evincere dall'implementazione del calcolo dei KPI [12].

## 4.2 DMDc

Presentazione dei risultati dei casi di studio applicati all'algoritmo DMDc.

### 4.2.1 SRU

Sono stati selezionati D (serie temporale di stati) ed U (serie temporale di ingressi) dal dataset SRU, con ritardi di 40 campioni per lo stato e per gli ingressi.

Rappresentazione dell'evoluzione del primo stato nel dataset sperimentale in figura:

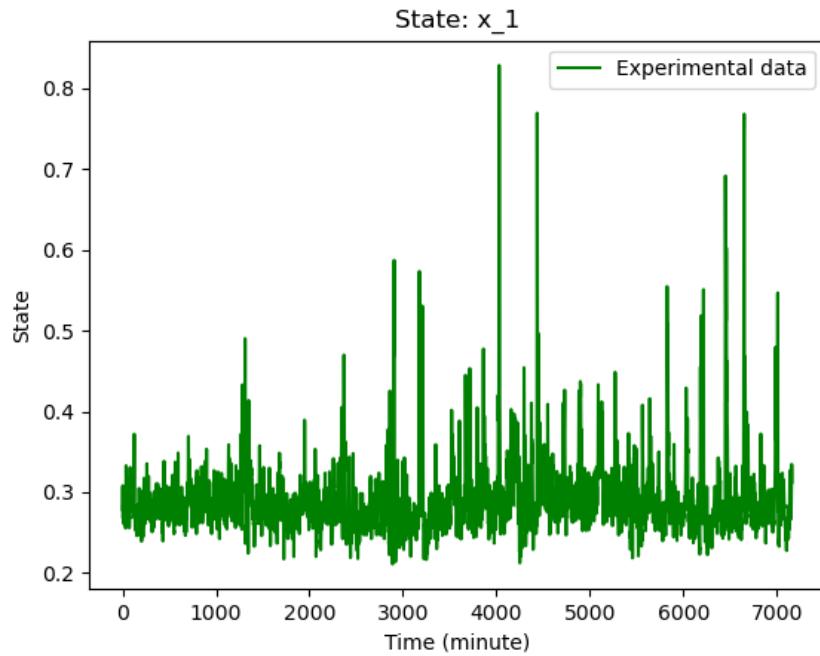
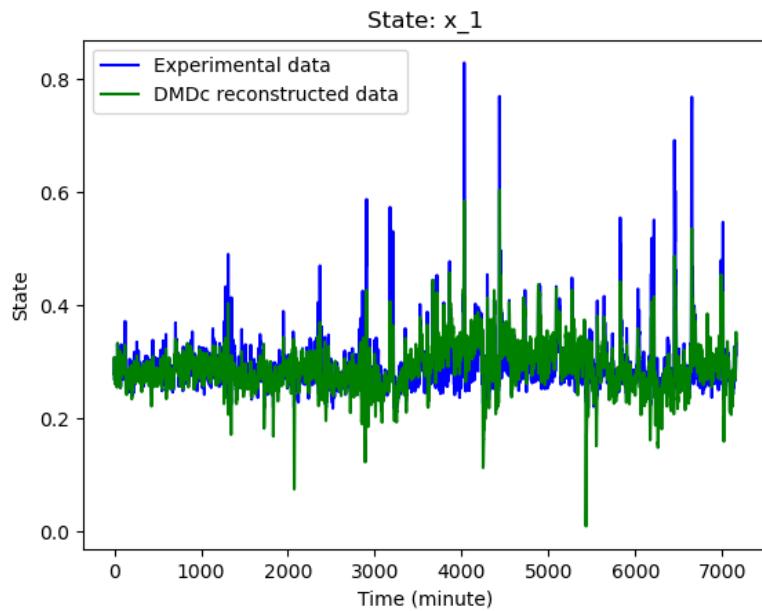
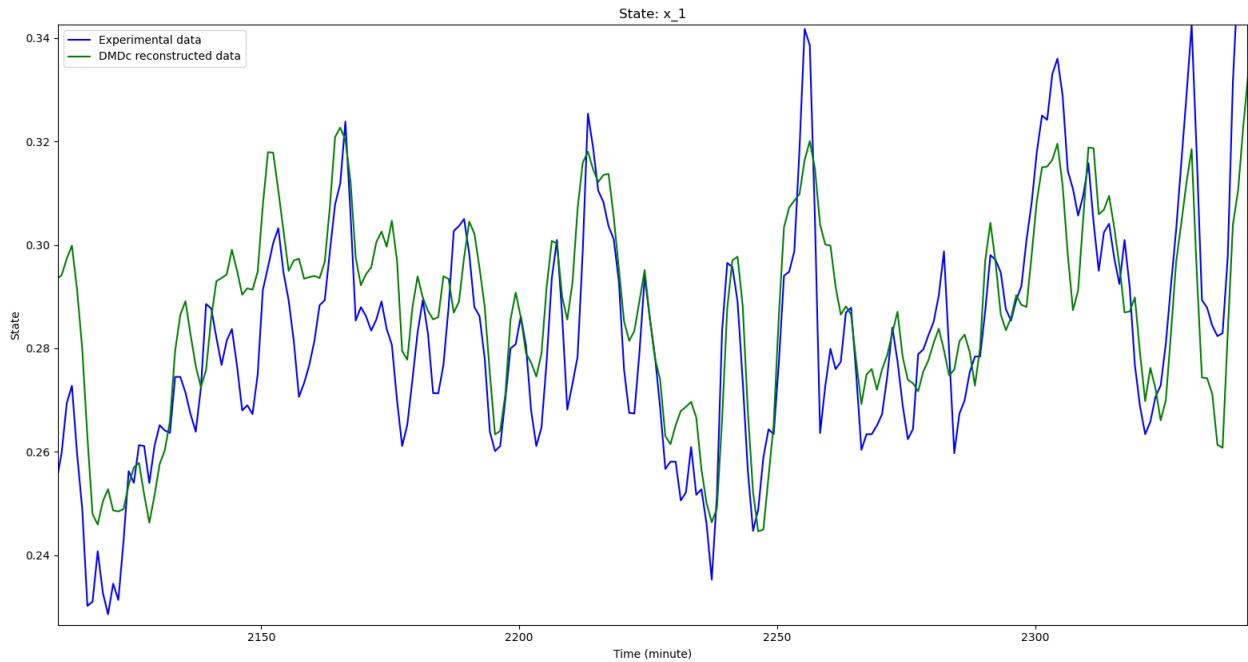


Figura 4.1

Confronto tra il primo stato del sistema originale con il primo stato del sistema ricostruito in figura:



Zoom in figura :



Ulterioro zoom in figura:

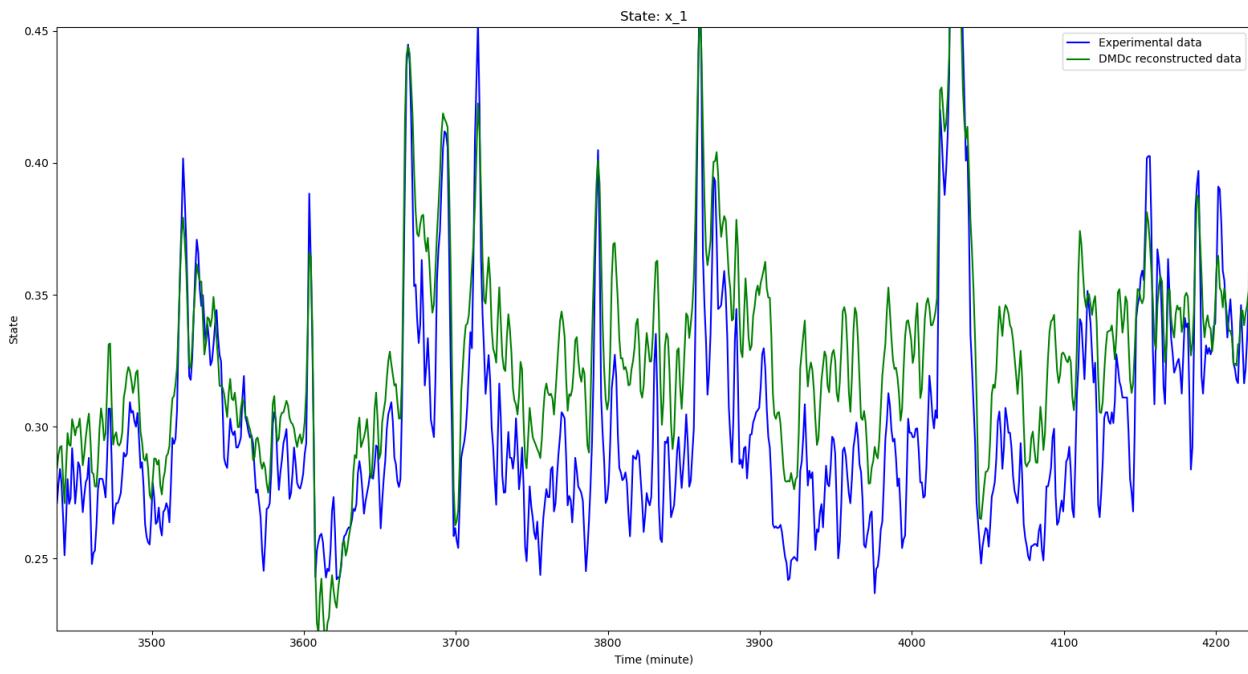


Figura 4.2

Vediamo il grafico dell'errore in figura:

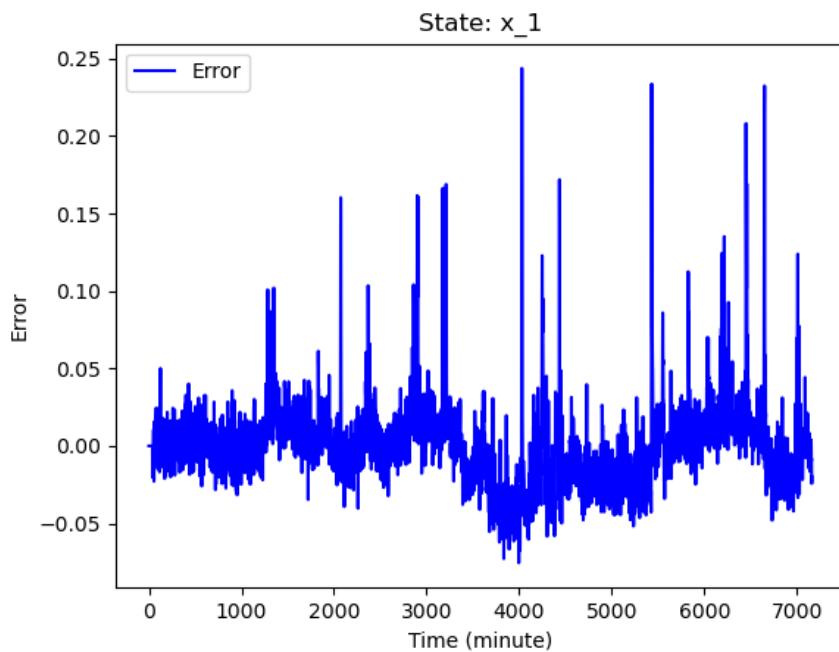


Figura 4.3

Vediamo i risultati dei KPI, calcolati considerando gli interi sistemi e non solo una colonna:

MSE	6.95e-04
MAPE	6.84e-02
MAE	1.80e-02
RMSE	2.64e-02
R <sup>2</sup>	0.63

#### 4.2.1 Sistema Sintetico con 5 Ingressi con ritardi dal Sistema SRU

Sono stati selezionati:

- D (serie temporale di stati) dal dataset generato dal sistema sintetico ad autovalori complessi descritto nel capitolo 3.
- U (serie temporale di ingressi) dal dataset SRU con ritardi di 40 campioni per gli ingressi.

Vediamo la rappresentazione della serie temporale degli stati del dataset sperimentale in figura:

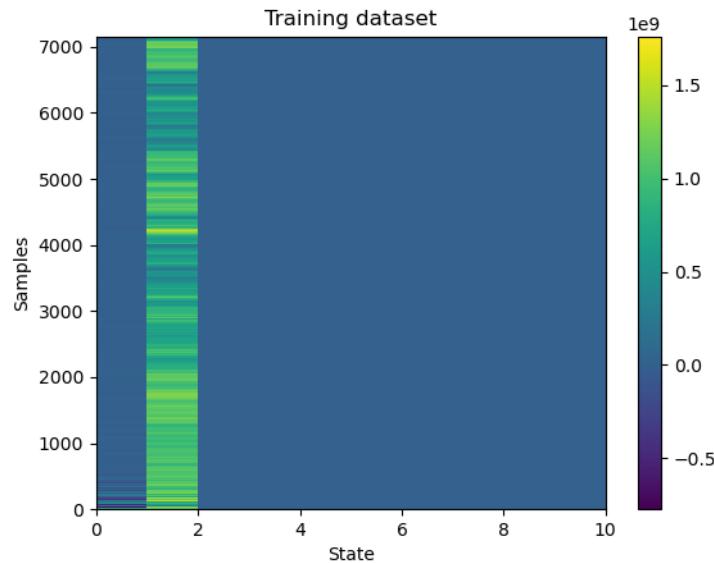


Figura 4.4

E vediamo anche la rappresentazione dell'evoluzione del primo stato nel dataset sperimentale in figura:

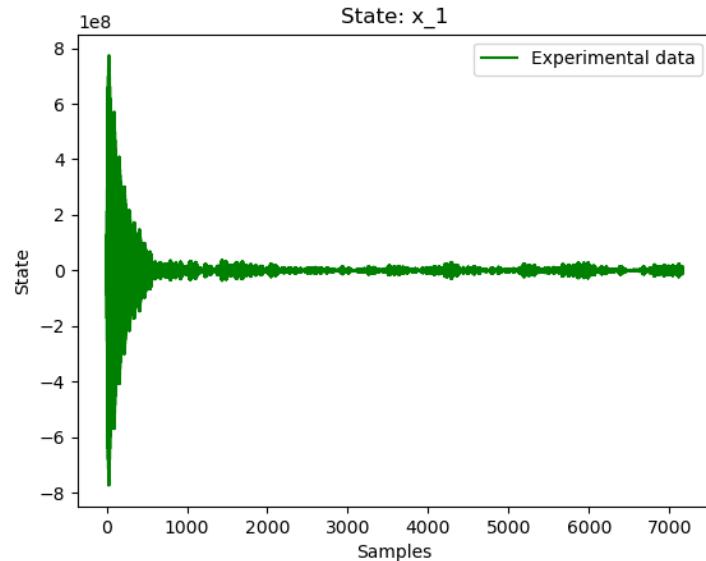


Figura 4.5

Presentiamo un confronto tra i dati sperimentali con i dati ricostruiti dal DMDc in figura:

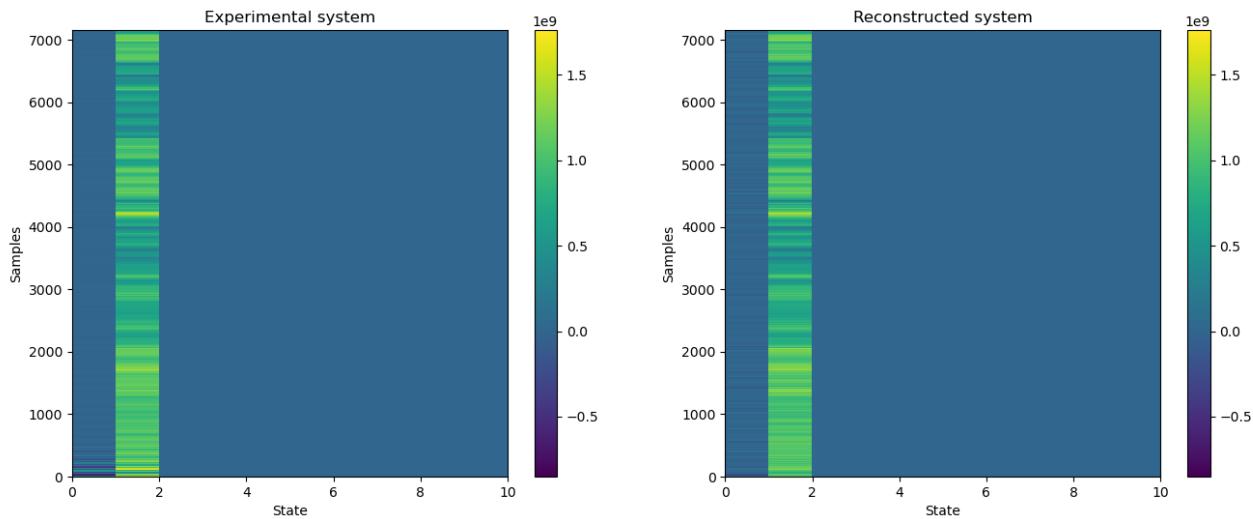
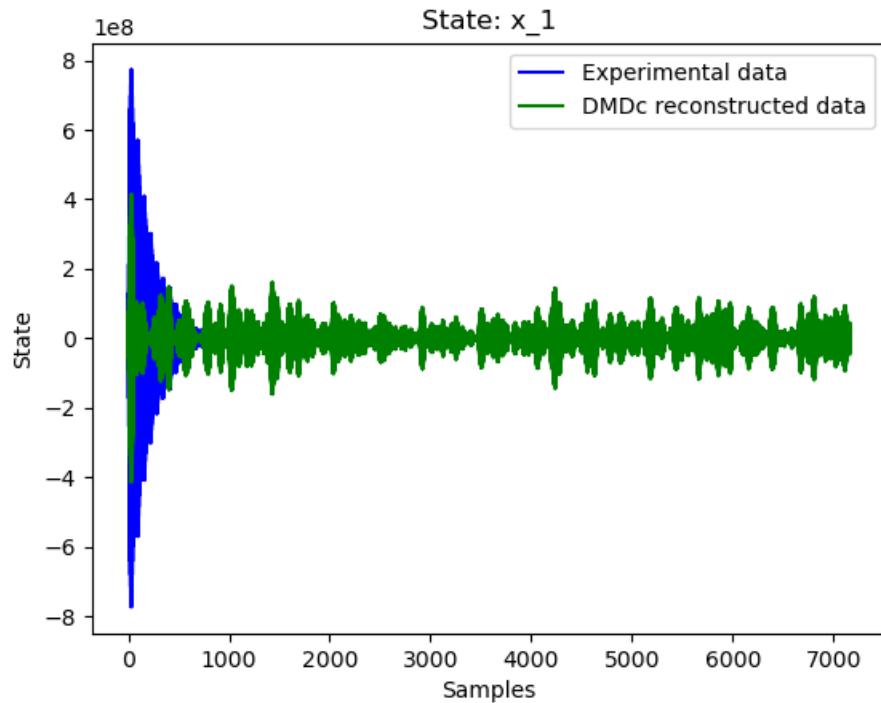
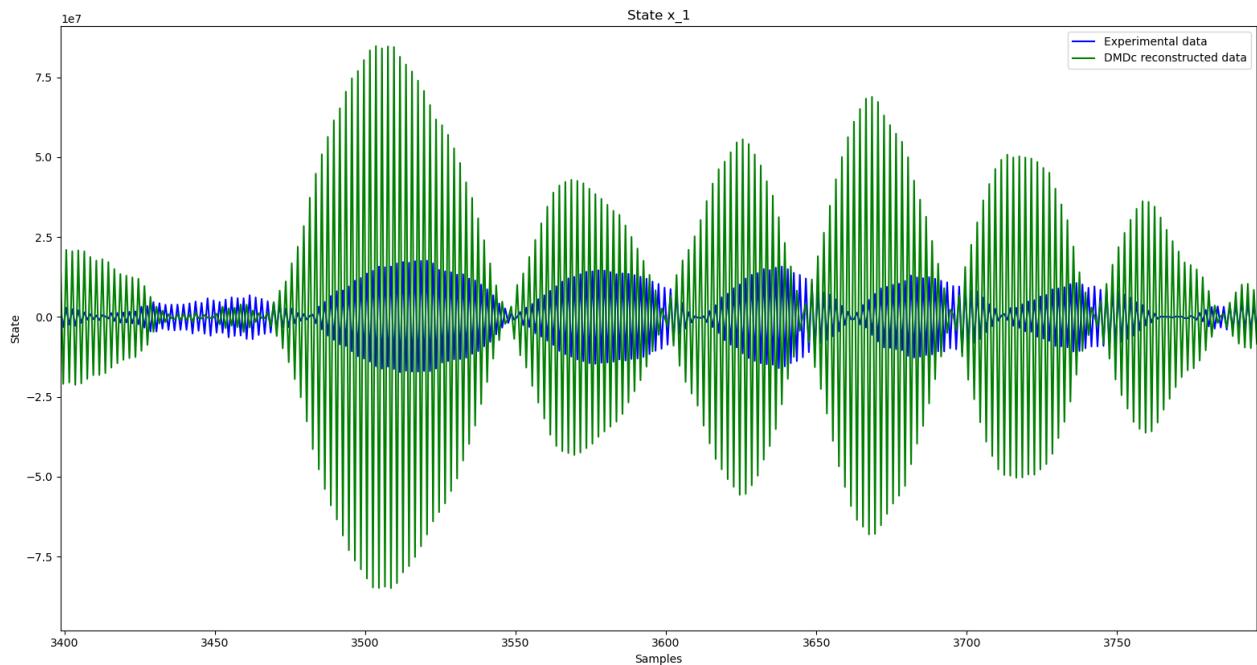


Figura 4.6

Confronto tra il primo stato del sistema originale con il primo stato del sistema ricostruito in figura:



Zoom in figura:



Ulteriore zoom in figura:

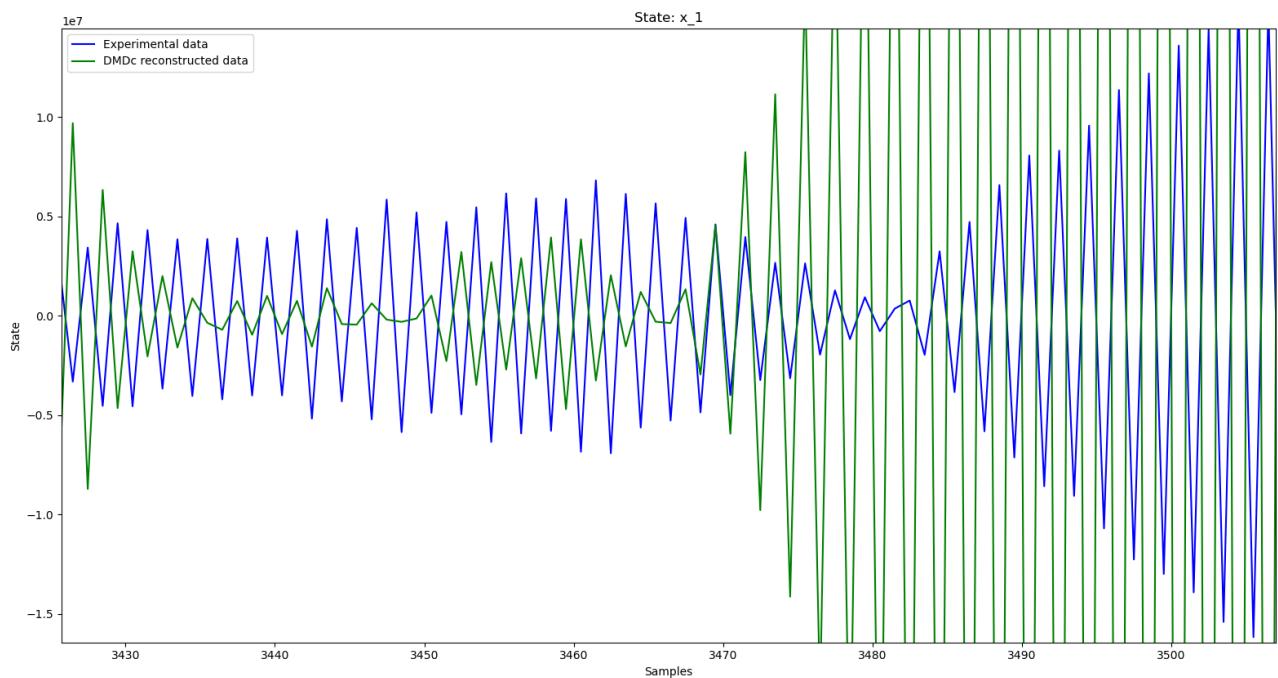


Figura 4.7

Vediamo il grafico dell'errore in figura:

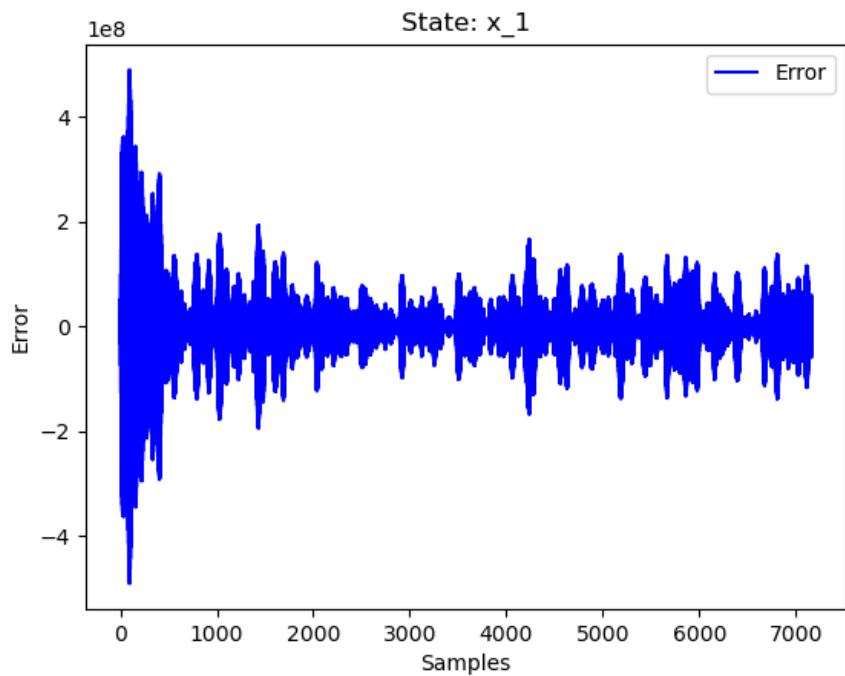


Figura 4.8

Vediamo i risultati dei KPI, calcolati considerando gli interi sistemi e non solo una colonna:

MSE	1.32e+15
MAPE	2.23
MAE	1.12e+07
RMSE	3.64e+07
R^2	0.33

#### 4.2.2 Sistema sintetico con ingresso n.5 non ritardato dai dati SRU

Sono stati selezionati:

- D (serie temporale di stati) dal dataset generato dal sistema ad autovalori complessi descritto nel capitolo 3.
- U (serie temporale di ingressi) dal dataset SRU, dove viene considerato solo il quinto ingresso non ritardato (colonna 161).

Vediamo la rappresentazione della serie temporale degli stati del dataset sperimentale in figura:

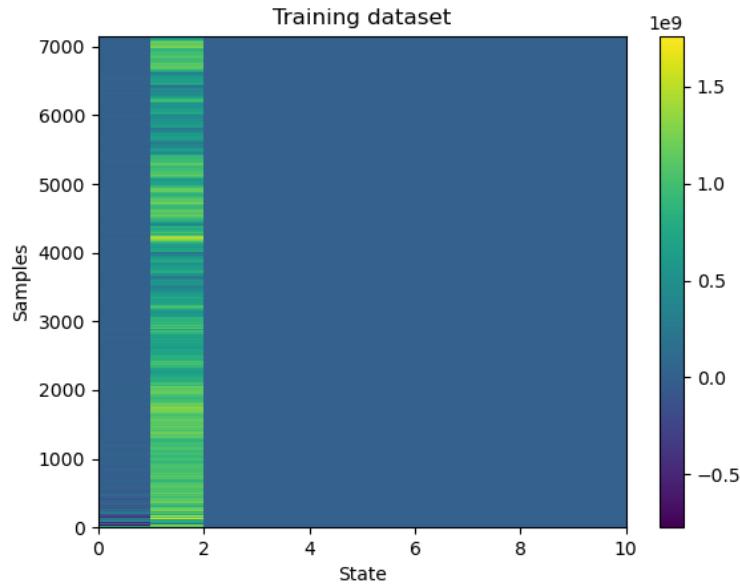


Figura 4.9

E vediamo anche la rappresentazione dell'evoluzione del primo stato nel dataset sperimentale in figura:

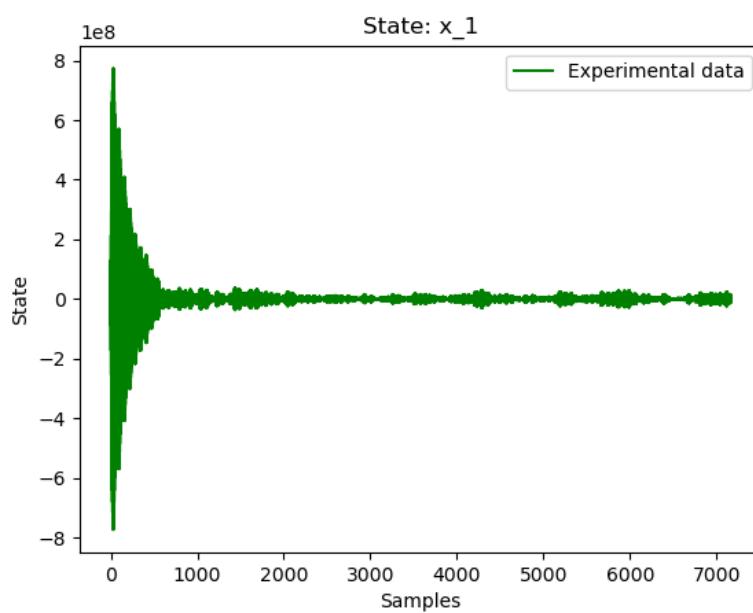


Figura 4.10

Presentiamo un confronto tra i dati sperimentali con i dati ricostruiti dal DMDc in figura:

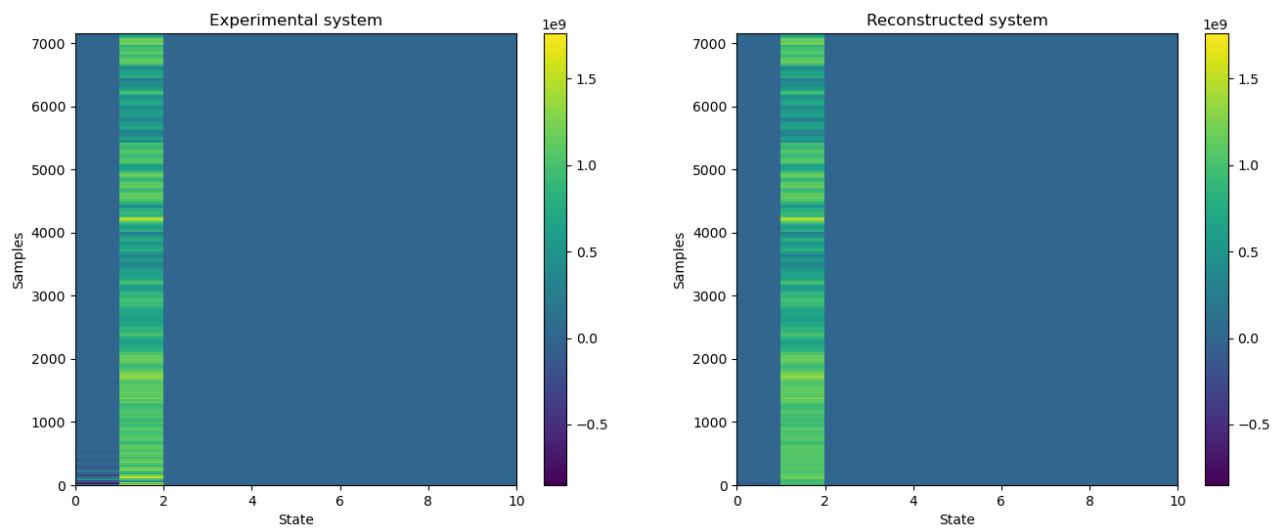


Figura 4.11

Confronto tra il primo stato del sistema originale con il primo stato del sistema ricostruito in figura:

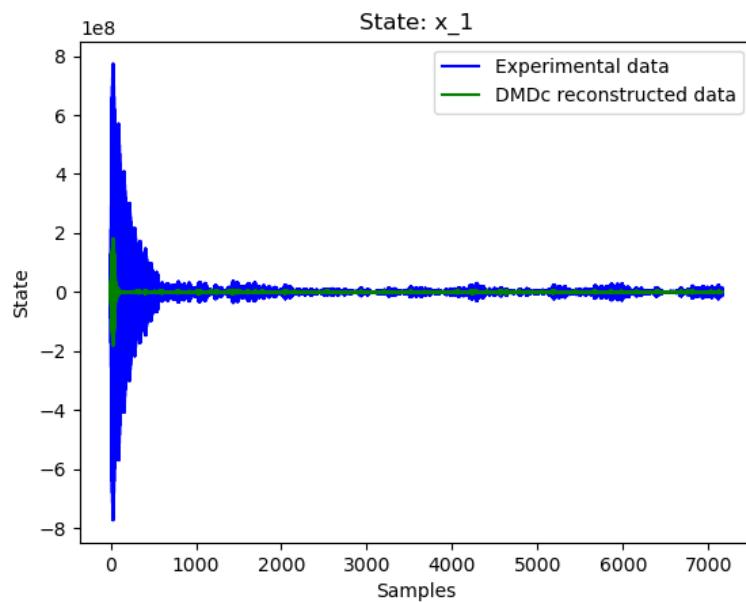


Figura 4.12

Vediamo il grafico dell'errore in figura:

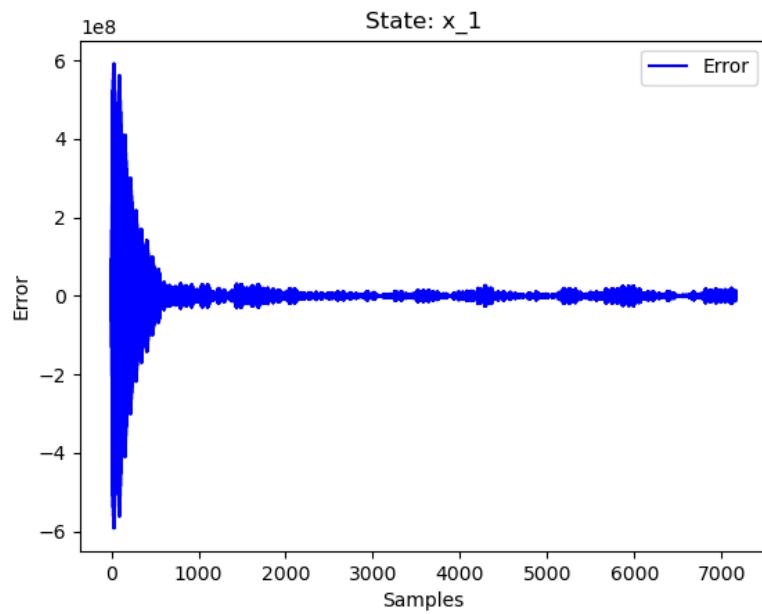


Figura 4.13

Vediamo i risultati dei KPI, calcolati considerando gli interi sistemi e non solo una colonna:

MSE	9.93e+14
MAPE	1.06e+02
MAE	5.18e+06
RMSE	3.15e+07
R^2	-1.04e+01

## 4.2.3 V2G

### 4.2.3.1 Stato non ritardato ed ingressi non ritardati

Sono stati selezionati D (serie temporale dello stato) ed U (serie temporale di ingressi) dal dataset V2G.

Rappresentazione dell'evoluzione dello stato nel dataset sperimentale in figura:

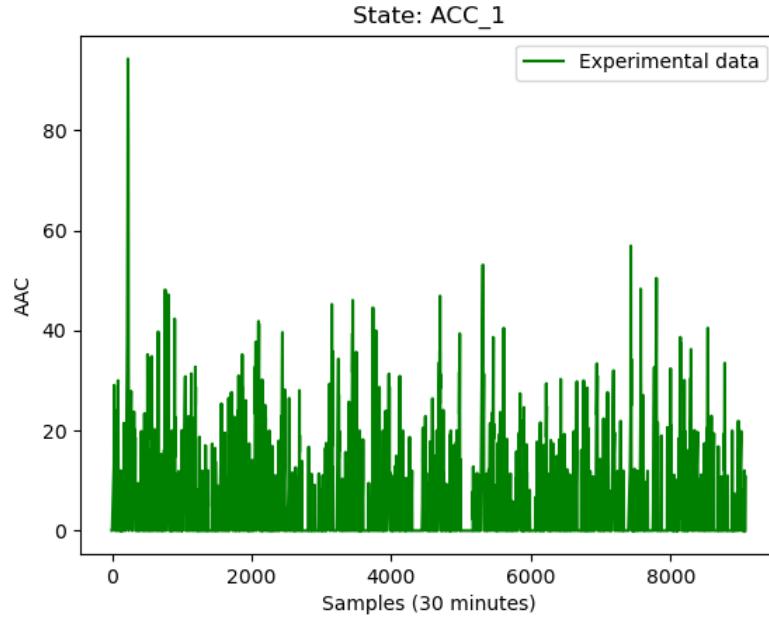
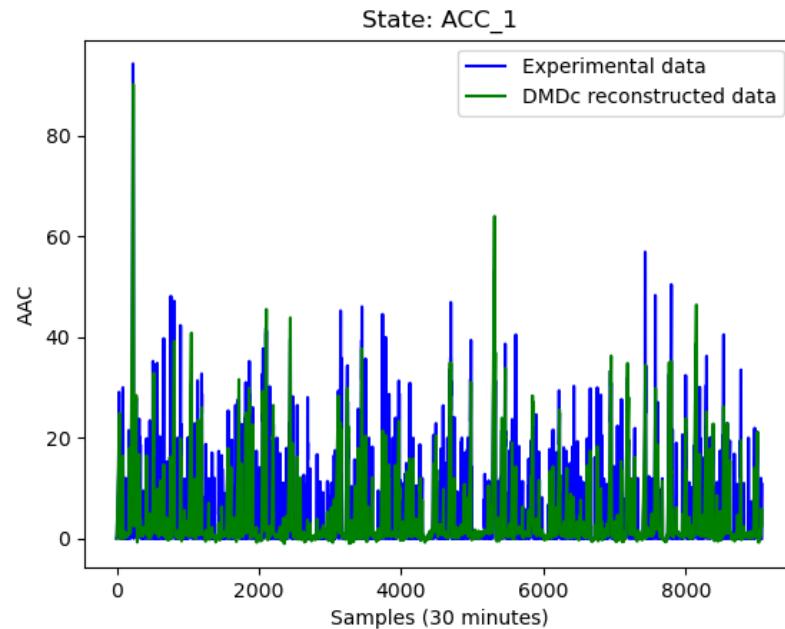
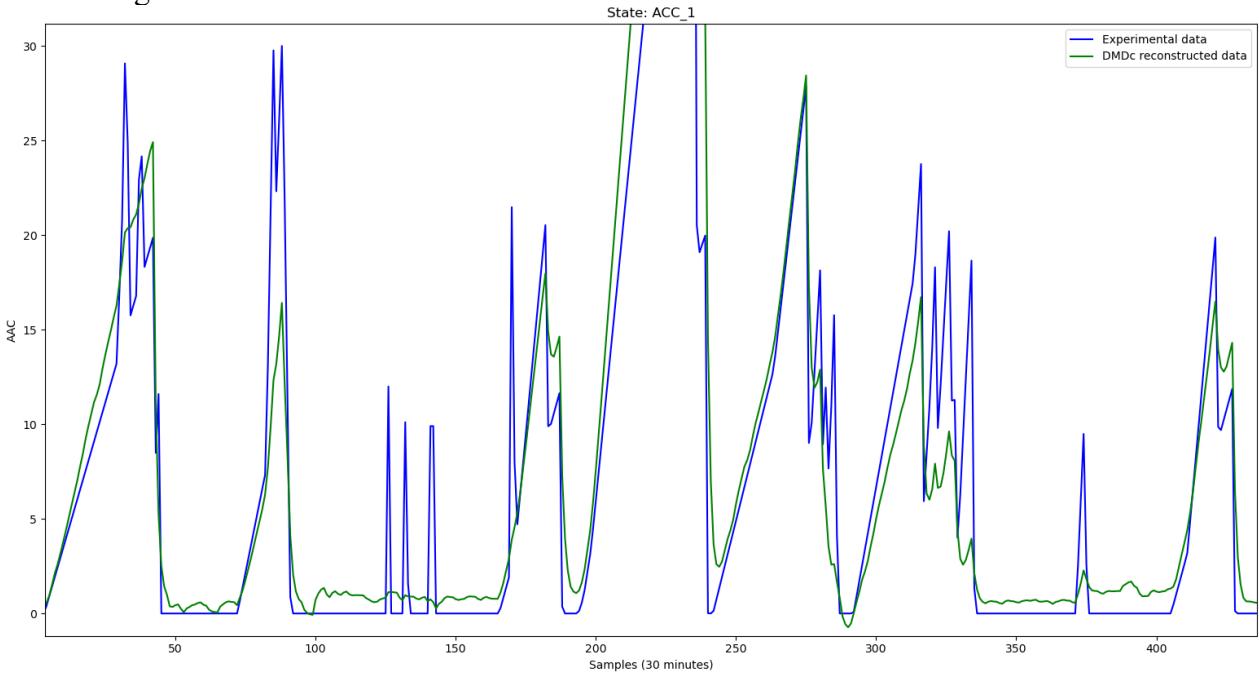


Figura 4.14

Confronto tra il primo stato del sistema originale con il primo stato del sistema ricostruito in figura:



Zoom in figura:



Ulteriore zoom in figura:

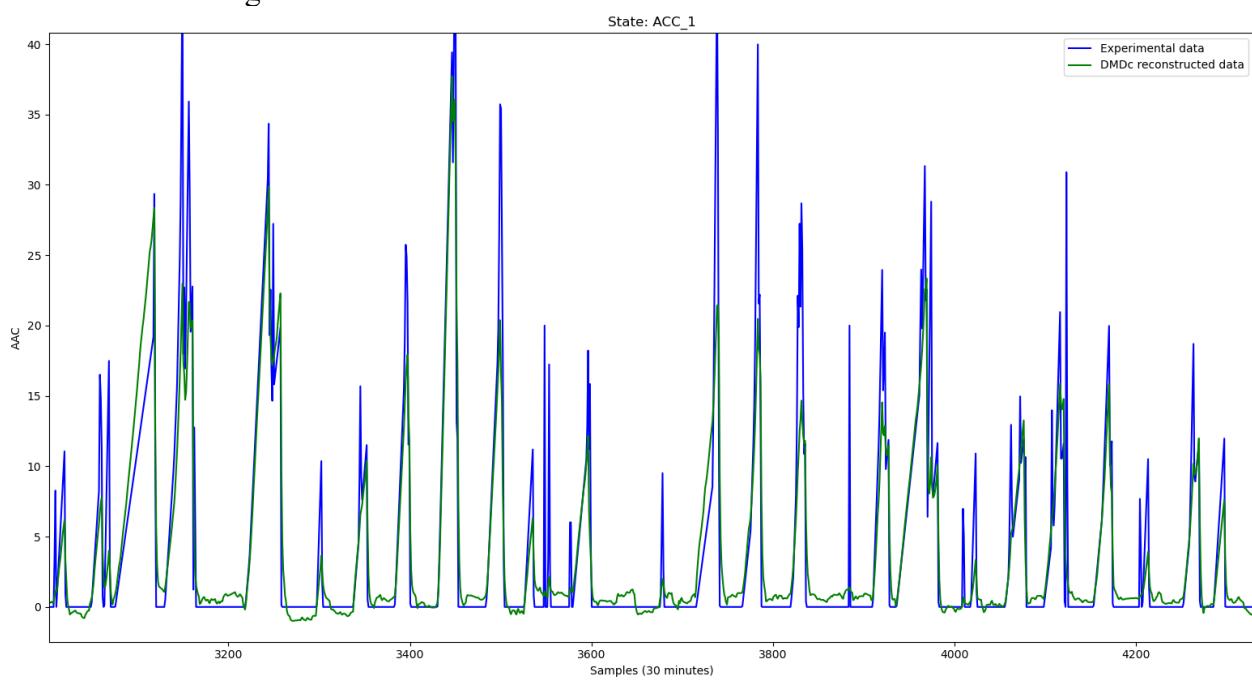


Figura 4.15

Vediamo il grafico dell'errore in figura:

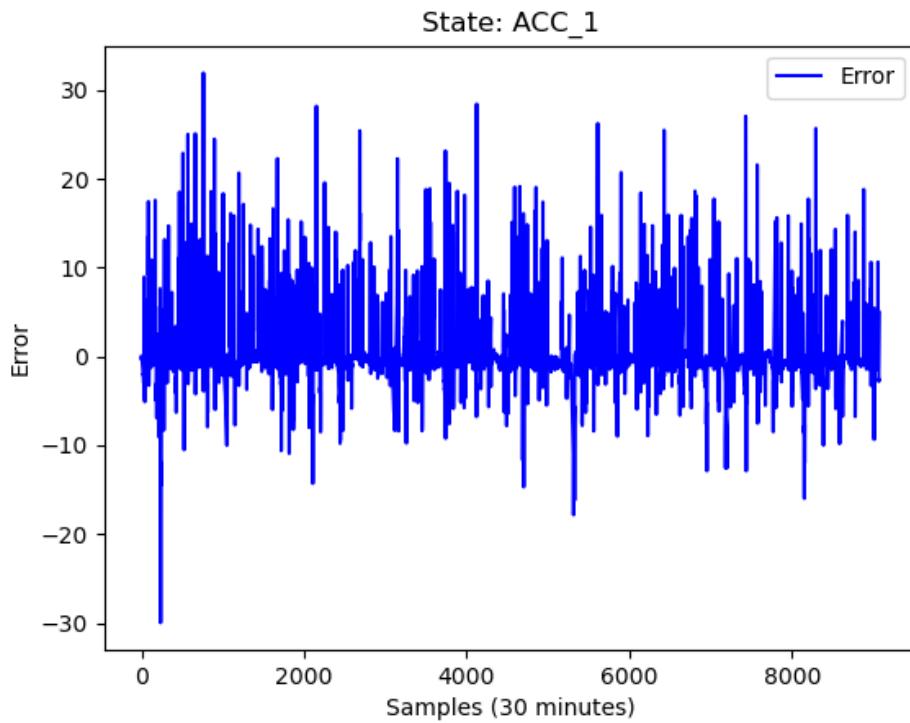


Figura 4.16

Vediamo i risultati dei KPI, calcolati considerando gli interi sistemi e non solo una colonna:

MSE	1.15e+01
MAPE	0.94
MAE	1.77
RMSE	3.39
R^2	0.80

#### 4.2.3.2 Stato ritardato ed ingressi non ritardati

Sono stati selezionati D (serie temporale dello stato ritardato 48 volte) ed U (serie temporale di ingressi) dal dataset V2G.

E vediamo anche la rappresentazione dell'evoluzione del primo stato nel dataset sperimentale in figura:

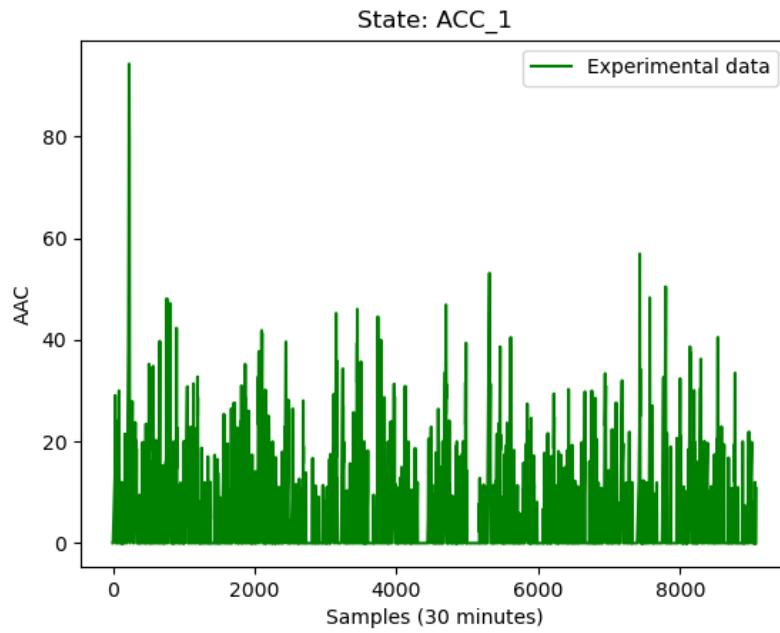


Figura 4.17

Confronto tra il primo stato del sistema originale con il primo stato del sistema ricostruito in figura:

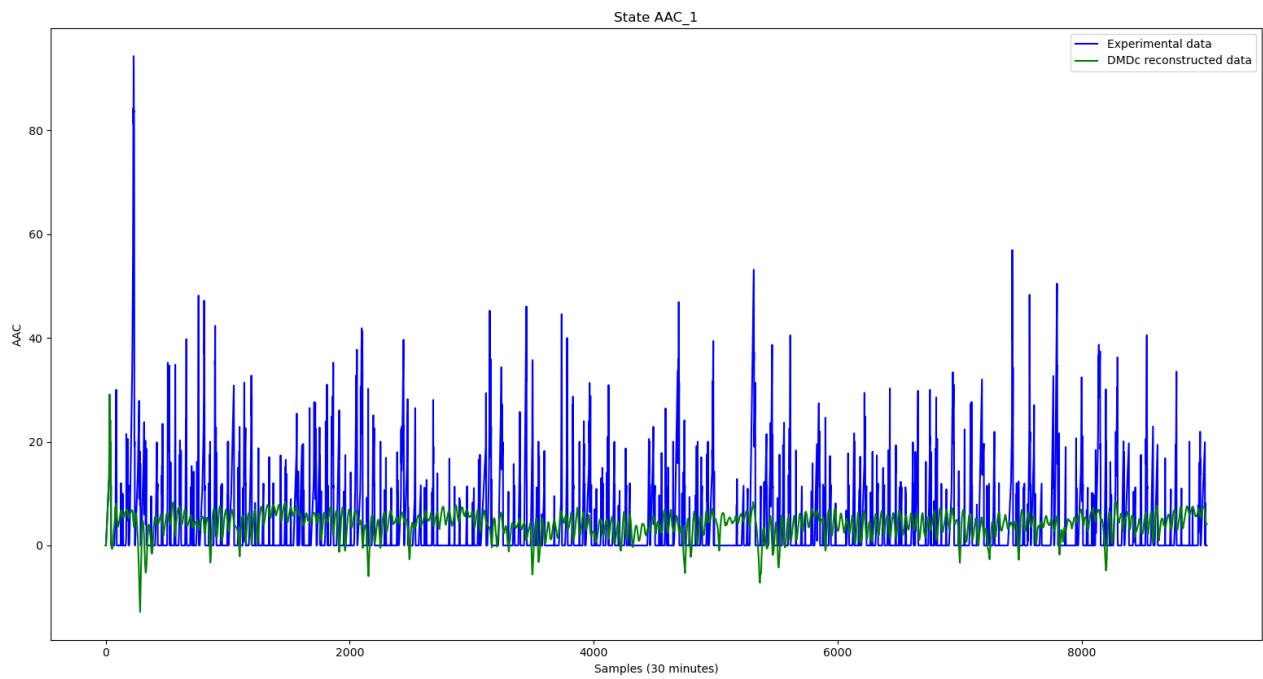


Figura 4.18

Vediamo il grafico dell'errore in figura:

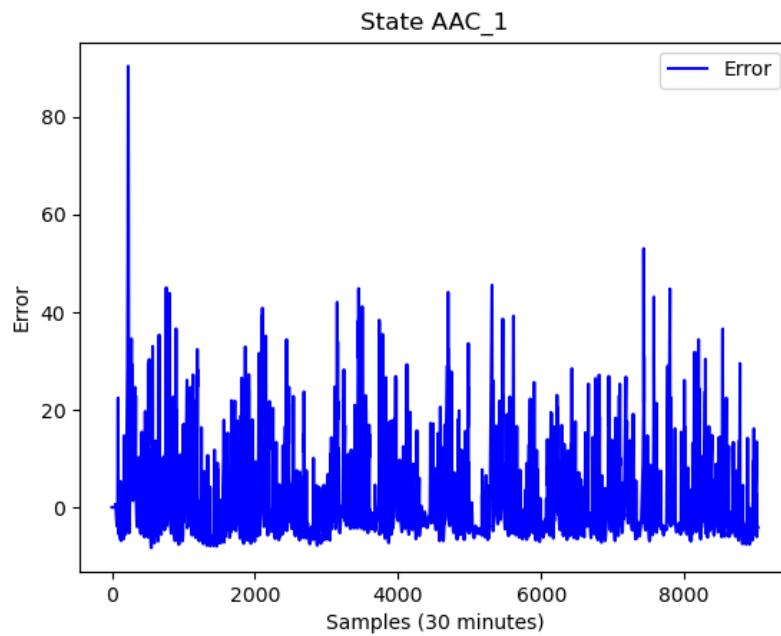


Figura 4.19

Vediamo i risultati dei KPI, calcolati considerando gli interi sistemi e non solo una colonna:

MSE	7.09e+01
MAPE	2.34
MAE	5.77
RMSE	8.42
R^2	-12.1

#### 4.2.3.3 Stato ritardato ed ingressi ritardati

Con questa soluzione sono stati eseguiti vari esperimenti dove si andavano a modificare gli ingressi, cercando la soluzione che permettesse di ottenere il risultato migliore. Di seguito vediamo le combinazioni provate:

- *meteo + aggregated*
- *aggregated*
- *meteo (no rhum<sub>t</sub>) + aggregated*
- *meteo (no rhum<sub>t</sub>) + aggregated (no holidays)*
- *aggregated (no holidays)*

Per ogni esperimento vengono riportate le matrici A e B, i KPI, ed un confronto tra tutte le ricostruzioni. Verrà eseguito anche il test con la valutazione dei KPI.

- *meteo + aggregated*:

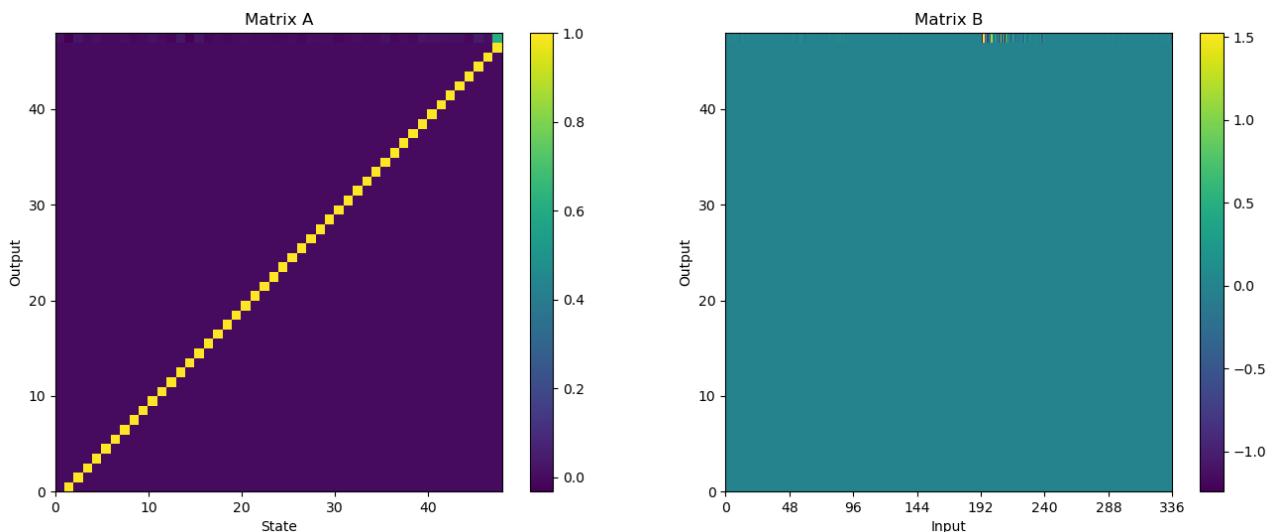


Figura 4.20

Train		Test	
MSE	9.10	MSE	1.18e+01
MAPE	1.88	MAPE	1.01
MAE	1.74	MAE	2.02
RMSE	3.02	RMSE	3.44
R^2	0.85	R^2	0.82

- aggregated:

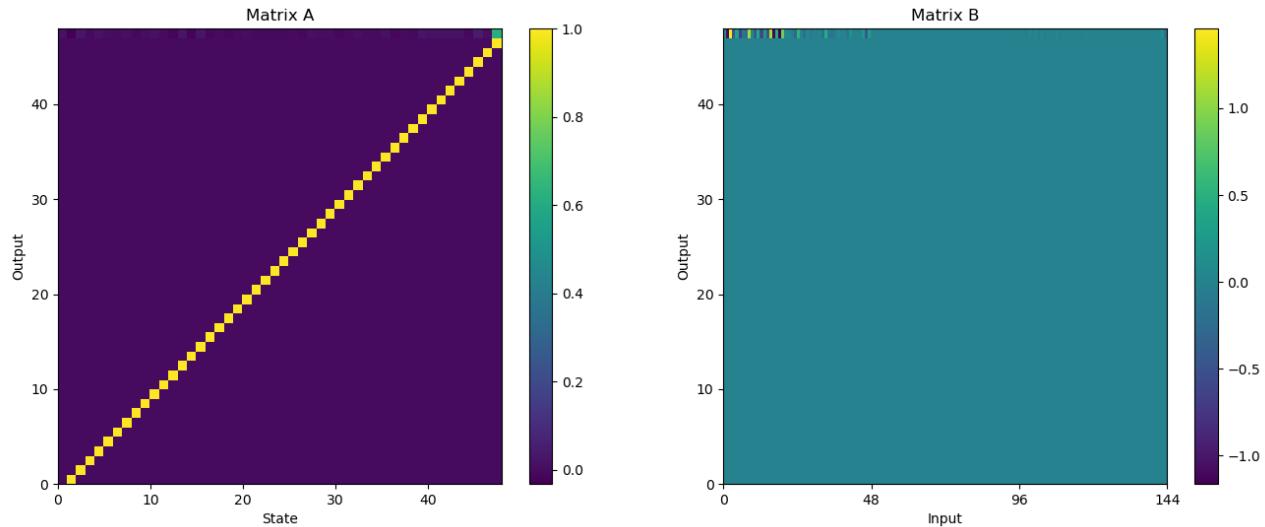


Figura 4.21

Train		Test	
MSE	9.63	MSE	1.19e+01
MAPE	0.97	MAPE	1.43
MAE	1.70	MAE	1.92
RMSE	3.10	RMSE	3.45
R^2	0.84	R^2	0.82

- meteo (no rhum<sub>t</sub>) + aggregated:

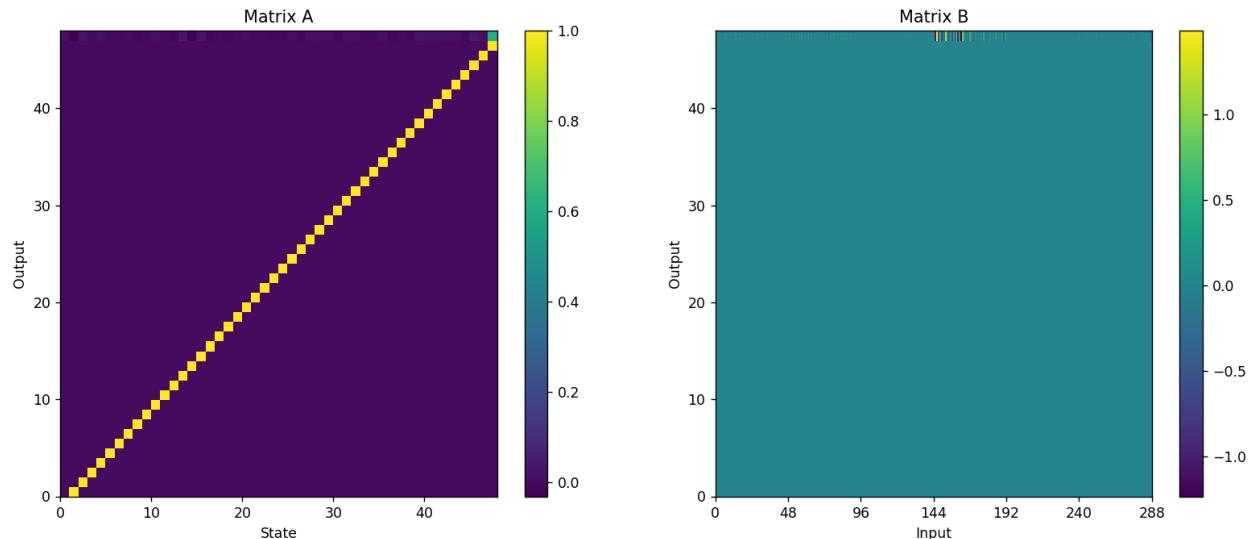


Figura 4.22

Train		Test	
MSE	9.20	MSE	1.19e+01
MAPE	1.09	MAPE	1.11
MAE	1.73	MAE	2.00
RMSE	3.03	RMSE	3.45
R^2	0.85	R^2	0.82

- *meteo (no rhum<sub>t</sub>) + aggregated (no holidays)*:

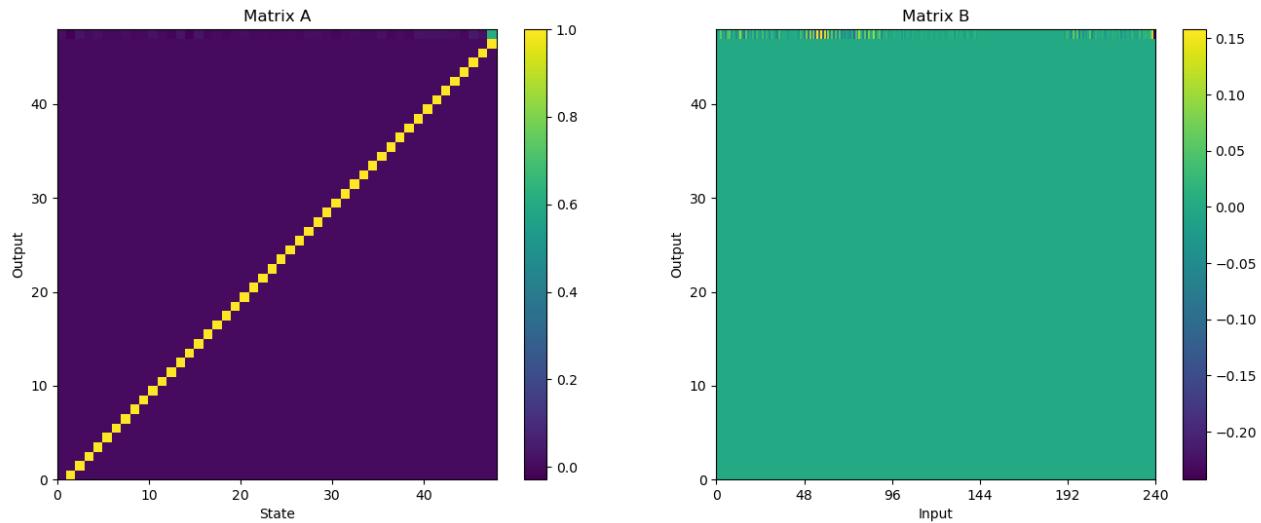


Figura 4.23

Train		Test	
MSE	9.27	MSE	1.18e+01
MAPE	1.16	MAPE	1.02
MAE	1.72	MAE	1.97
RMSE	3.04	RMSE	3.44
R <sup>2</sup>	0.85	R <sup>2</sup>	0.82

- *aggregated (no holidays)*:

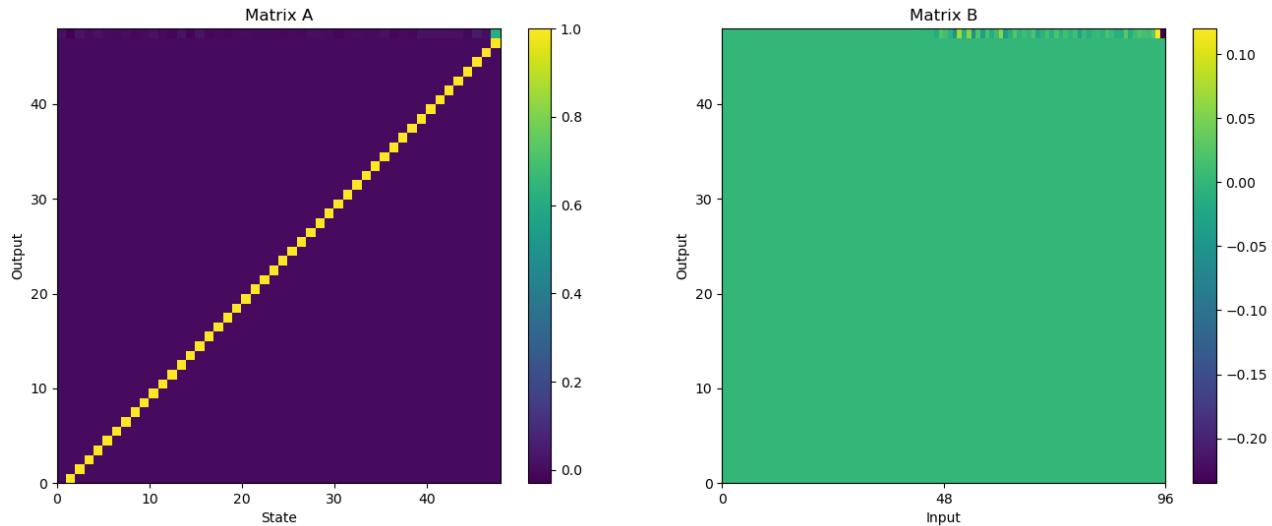


Figura 4.24

Train		Test	
MSE	9.70	MSE	1.18e+01
MAPE	0.99	MAPE	1.12
MAE	1.69	MAE	1.89
RMSE	3.11	RMSE	3.44
R <sup>2</sup>	0.84	R <sup>2</sup>	0.82

Vengono ripetuti i grafici delle A e B con stati ed ingressi normalizzati tramite min max *scaler*:

- *meteo + aggregated*:

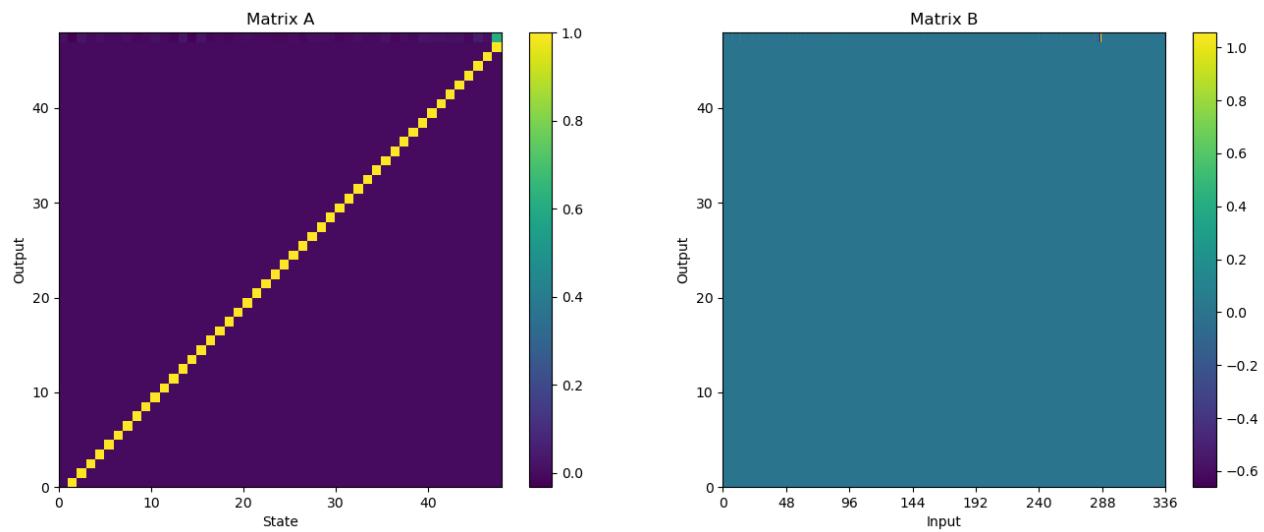


Figura 4.25

- *aggregated*:

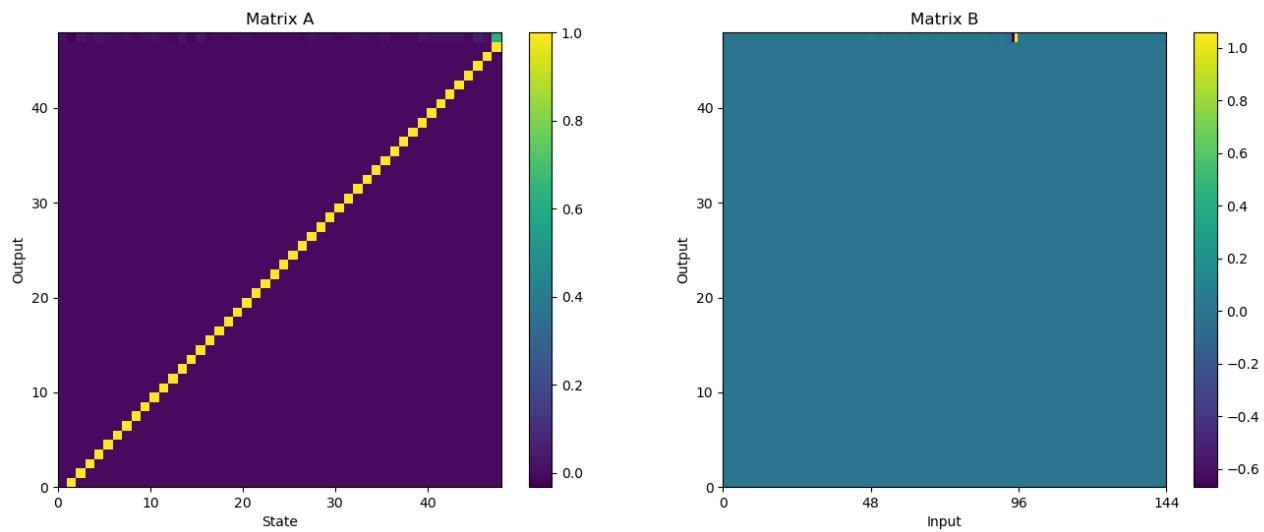


Figura 4.26

- *meteo (no rhum<sub>t</sub>) + aggregated:*

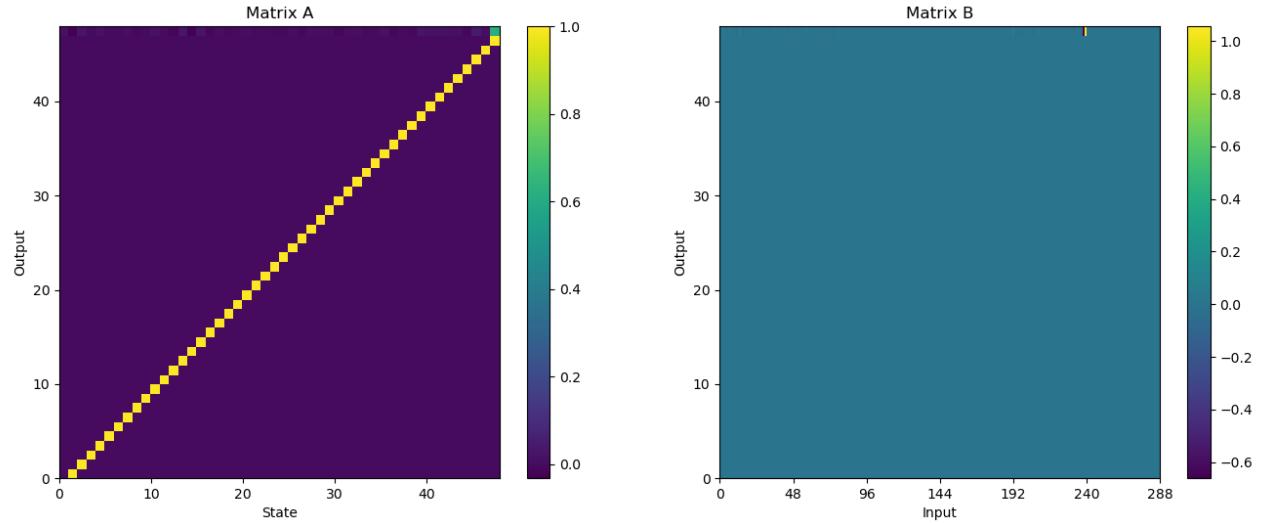


Figura 4.27

- *meteo (no rhum<sub>t</sub>) + aggregated (no holidays):*

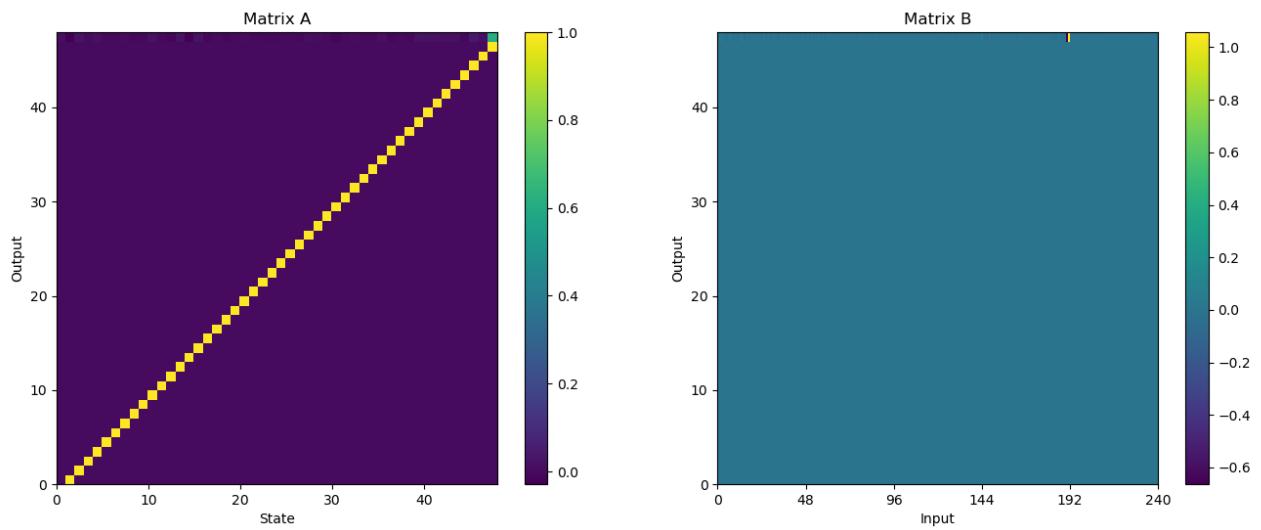


Figura 4.28

- *aggregated (no holidays)*:

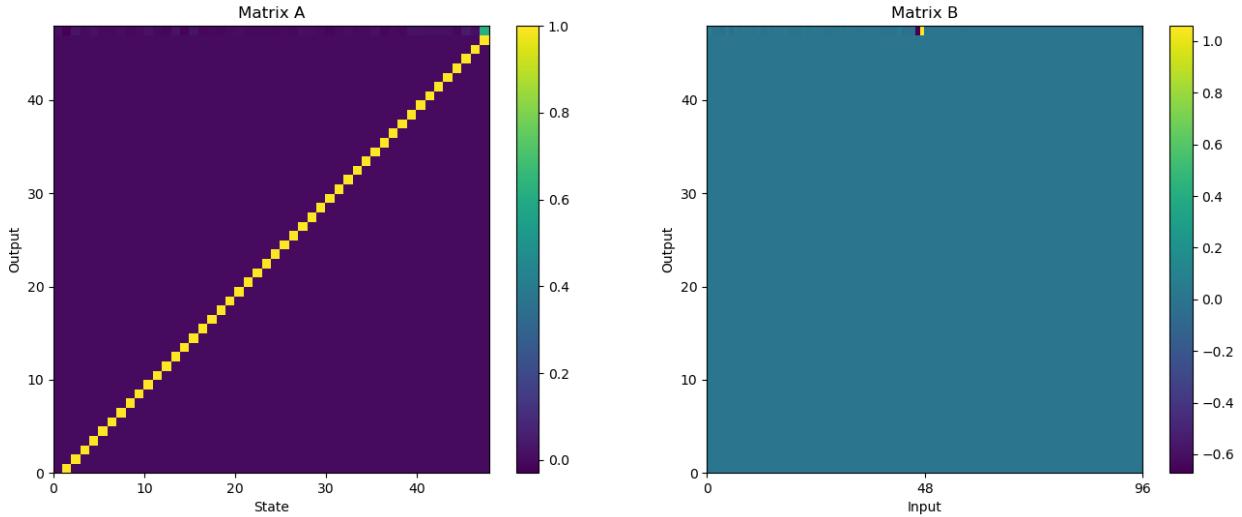
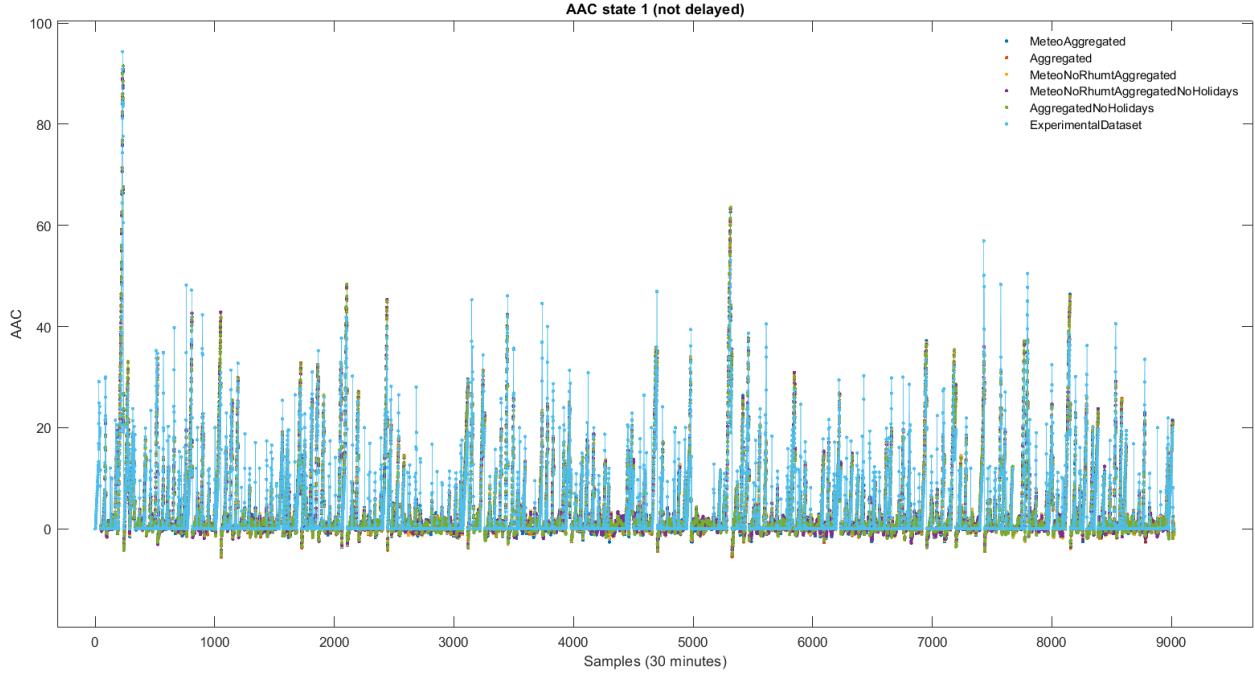


Figura 4.29

Viene presentato un plot con tutte le evoluzioni dello stato non ritardato (corrispondente alla prima colonna delle matrici ricostruite) per tutte le prove descritte in precedenza.

Grafico training:



Zoom:

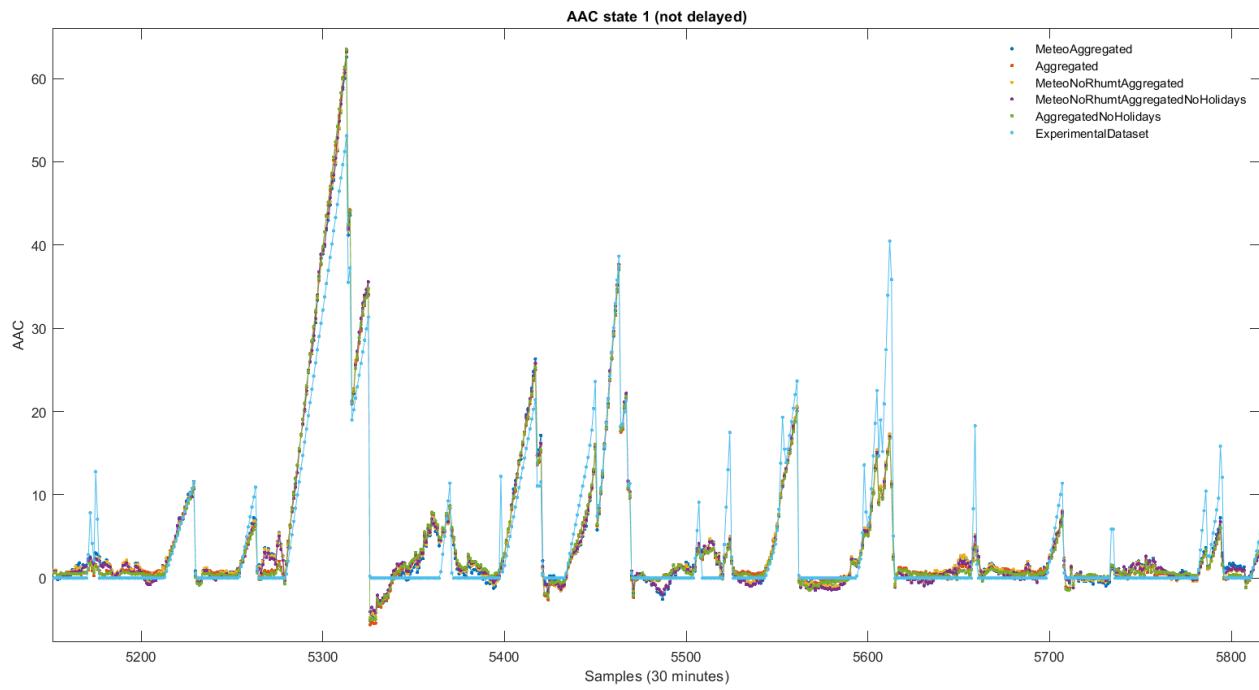
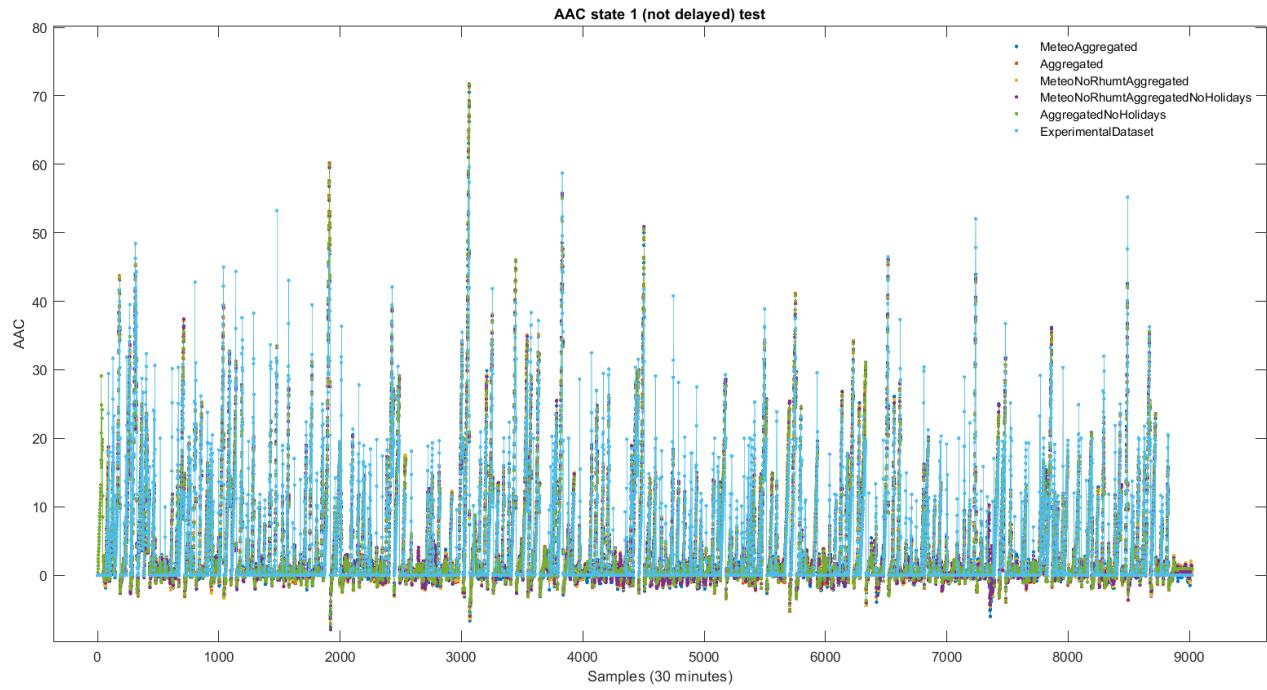
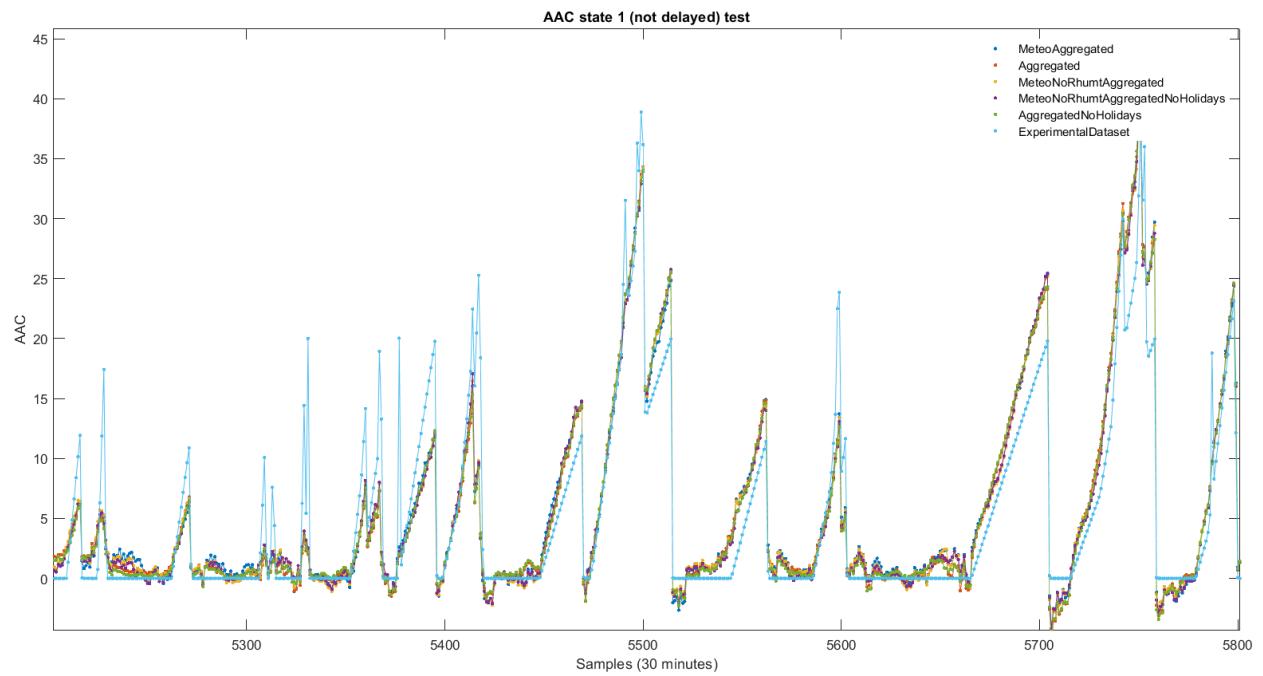


Grafico test:



Zoom:



## 4.3 mrDMDc

Presentazione dei risultati dei casi di studio applicati al mrDMDc.

### 4.3.1 Ricostruzione a passo 1

Viene presentato un singolo esempio di ricostruzione con le dinamiche calcolate con passo  $dt = 1$ , come è stato descritto nel capitolo 2 riguardante la teoria dell'algoritmo mrDMDc.

#### 4.3.1.1 SRU

4.3.1.1.1 *svd rank = -1, max cycles = 10, slow feature scale = 15, reconstruction = 2:*

```
#svd_rank for DMDc object
'''param svd_rank: the rank for the truncation; If 0, the method computes the optimal rank and uses it for truncation;
| if positive interger, the method uses the argument for the truncation; if float between 0 and 1, the rank is the number
| of the biggest singular values that are needed to reach the 'energy' specified by `svd_rank`; if -1, the method does
| not compute truncation.''''
svd_rank_set = -1

#max_cycles for mrDMDc
max_cycles_set = 10
'''

there is a possibility that if the reconstruction is not good (an example if the reconstructed system is unstable),
changing the parameter will work.
No way has been figured out to decide whether to increase or decrease it in the event of a bad reconstruction
'''

#index of filtration, needs to set the size of rho
slow_feature_scale = 15

'''selection of the reconstruction type
1 Reconstruction with A and B dt = step
2 Reconstruction with A and B dt = 1 (trasformation of A and B with Harold library)
3 Reconstruction with forced response (work in progress)
'''
reconstruction = 2
```

Codice 4.1

Rappresentazione dell'evoluzione del primo stato nel dataset sperimentale in figura:

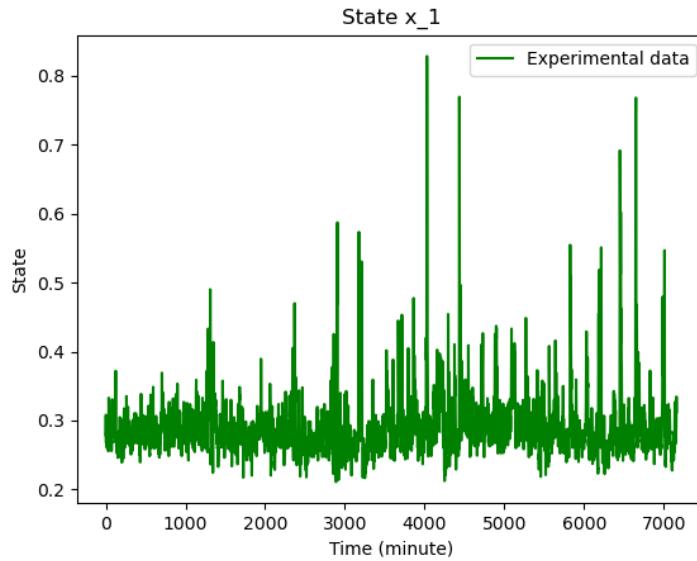
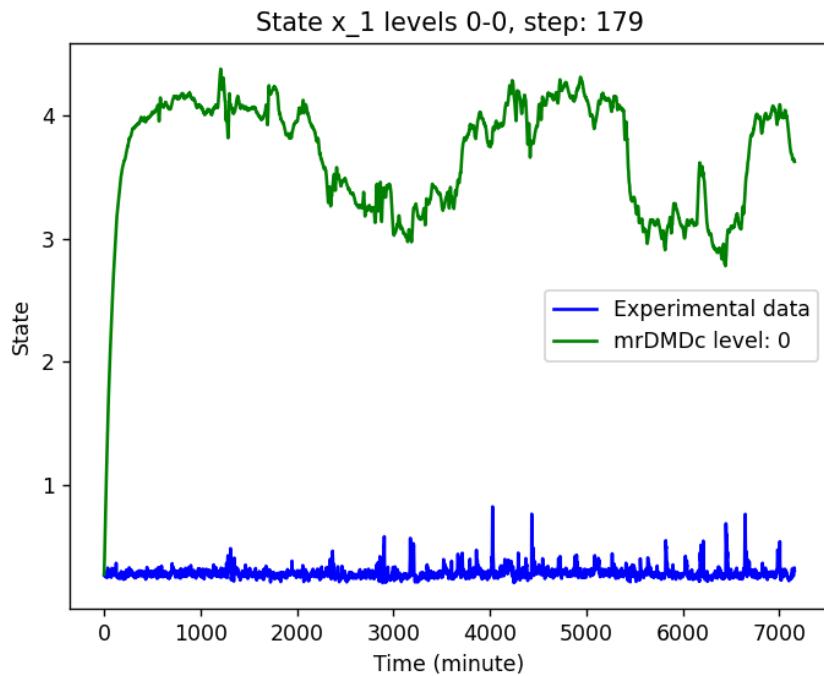
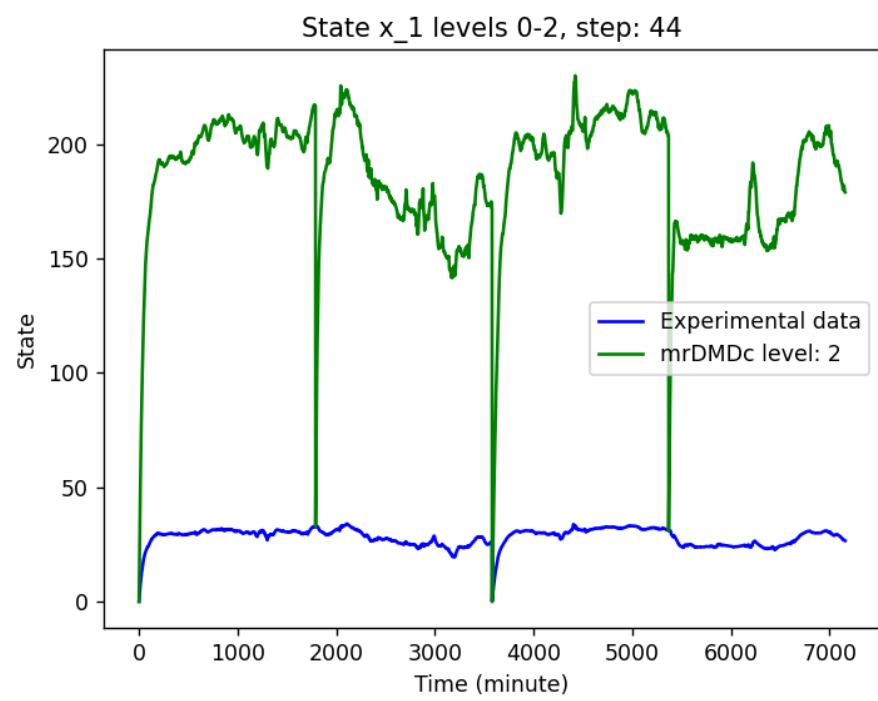
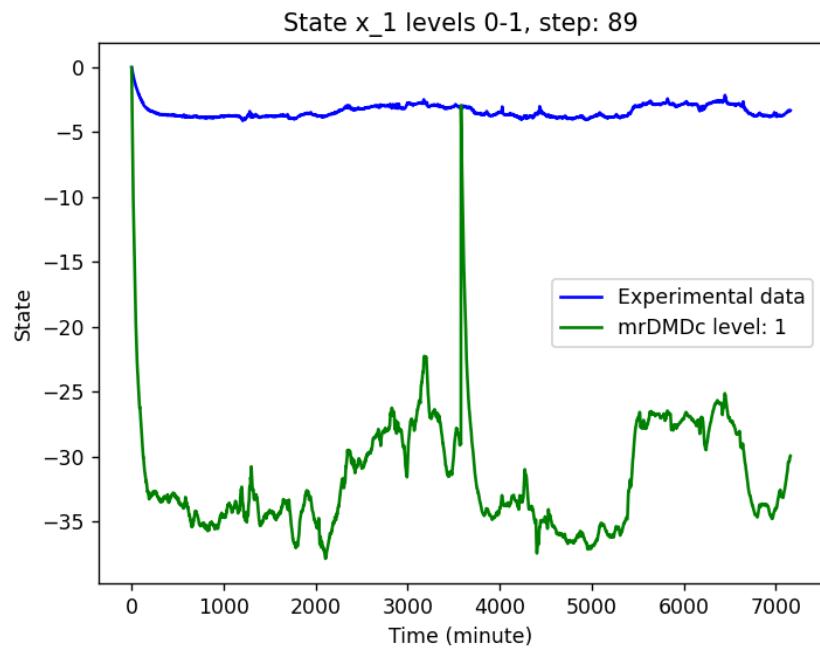


Figura 4.30

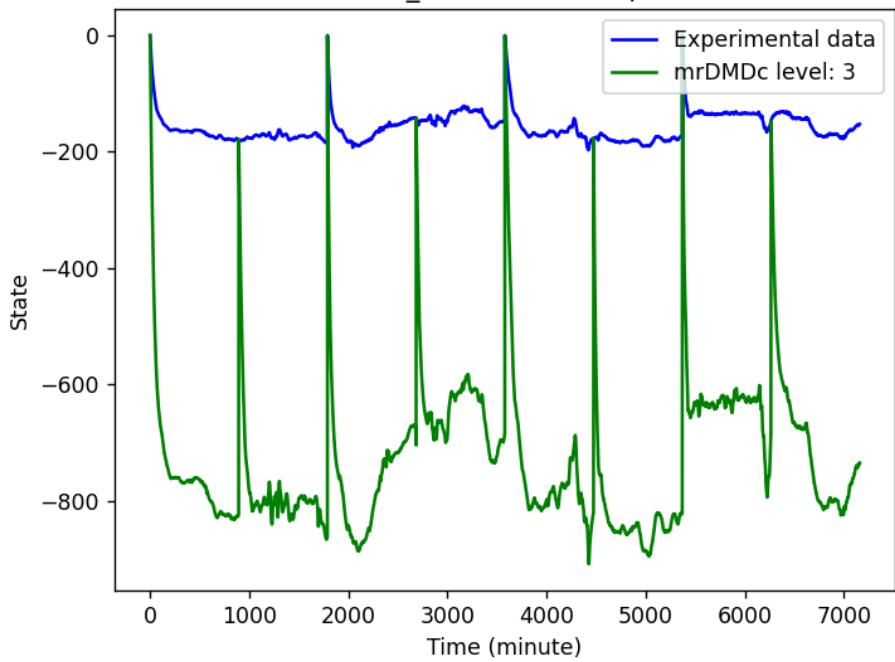
Di seguito la ricostruzione di ogni livello eseguita dal mrDMDc.

I grafici visualizzati sono il confronto tra i dati sperimentali e i dati ricostruiti dall'mrDMDc. Come spiegato in precedenza ogni ricostruzione del mrDMDc viene sottratta ai dati di training passati come parametro all'algoritmo; nelle figure vengono riproposti i dati di cui l'mrDMDc esegue la ricostruzione (che chiaramente come si può dedurre ricostruisce in maniera scorretta):

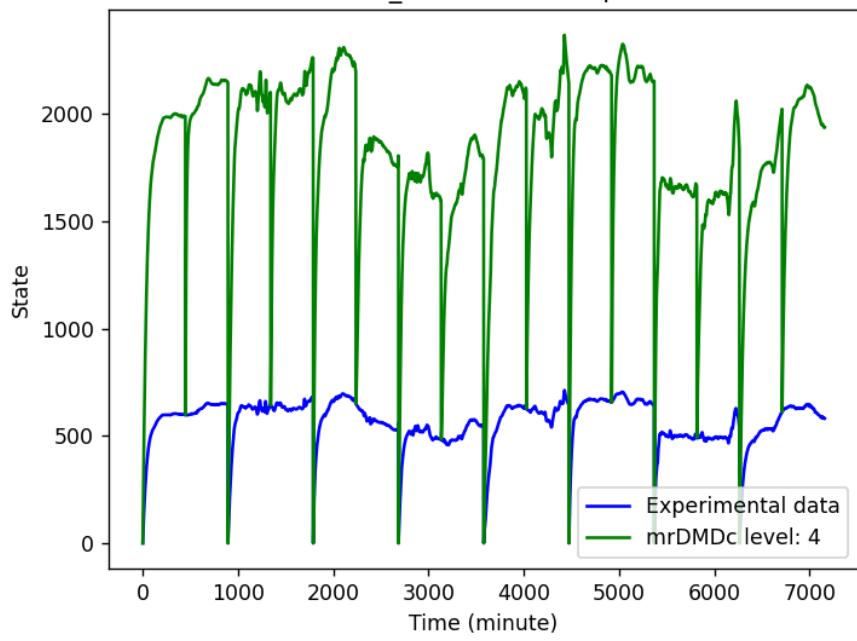


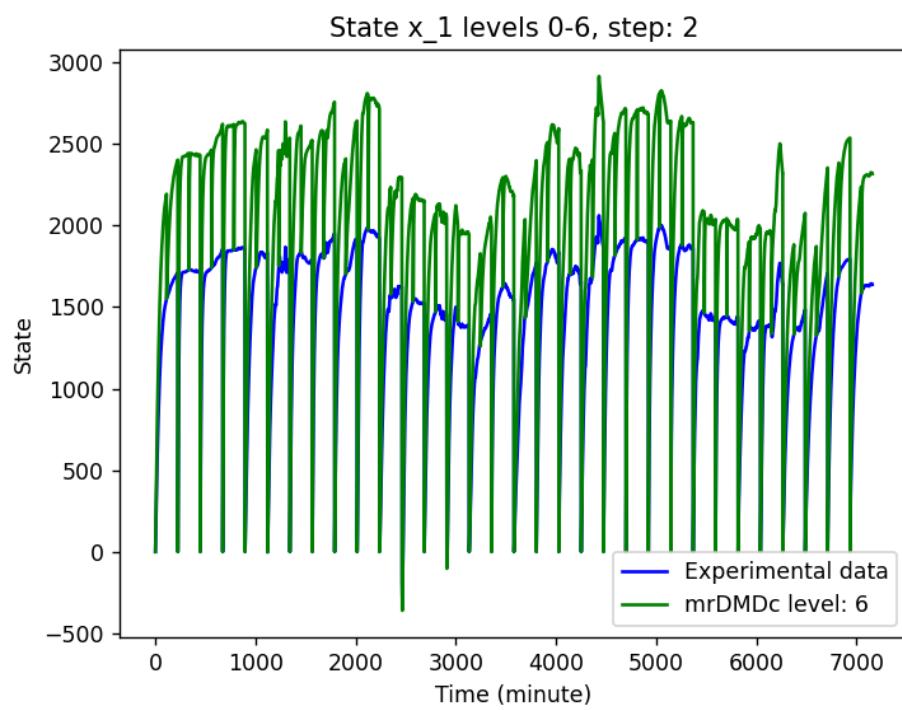
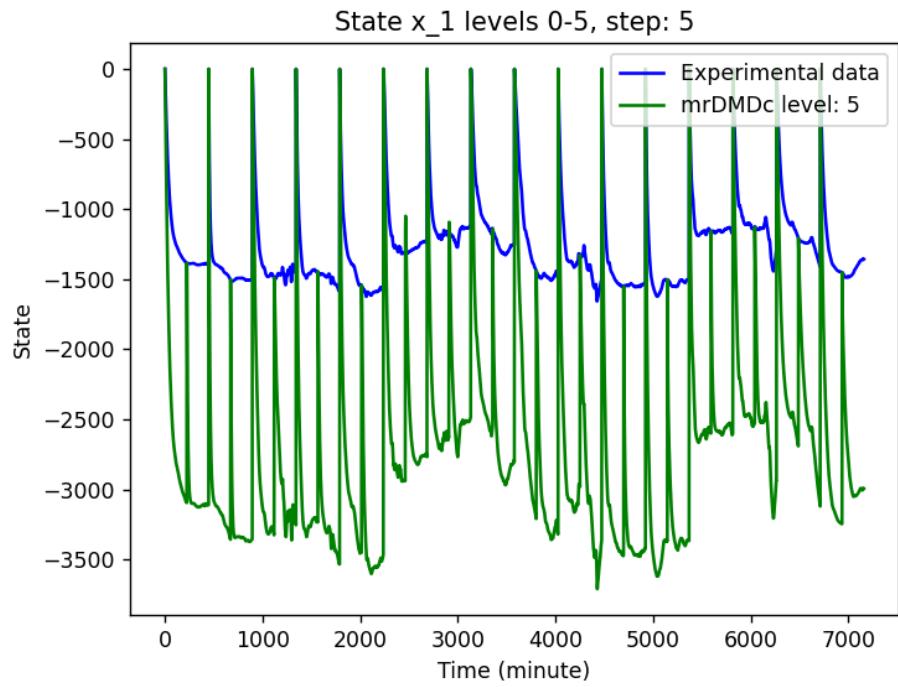


State  $x_1$  levels 0-3, step: 22



State  $x_1$  levels 0-4, step: 11





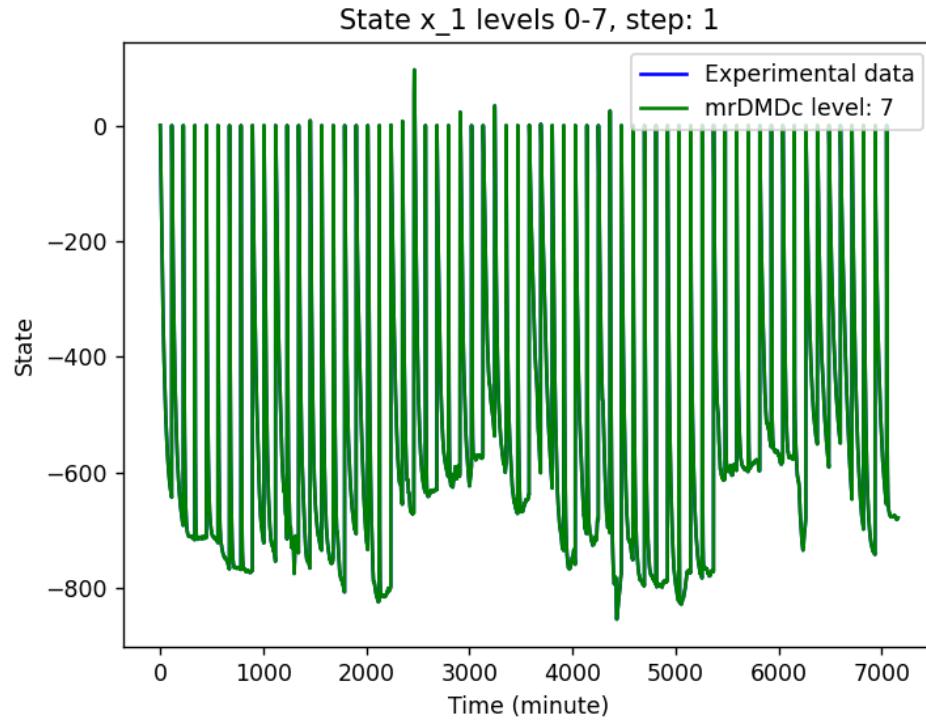


Figura 4.31

Per la ricostruzione finale è possibile vedere i contributi di ogni singolo livello analizzando questo grafico:

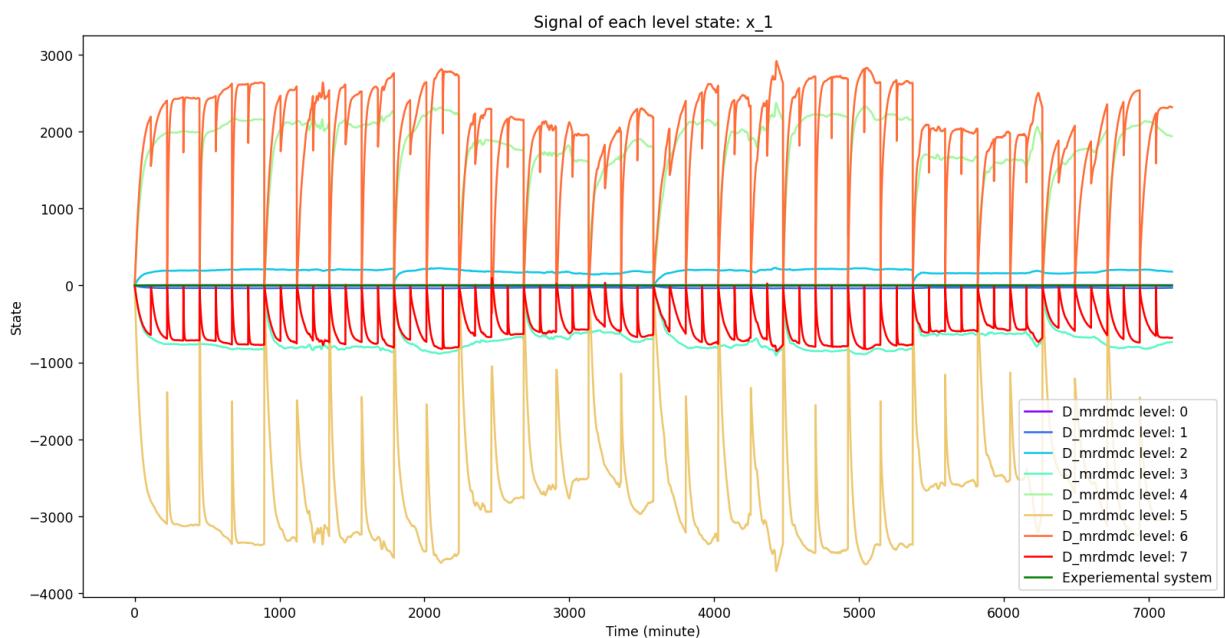
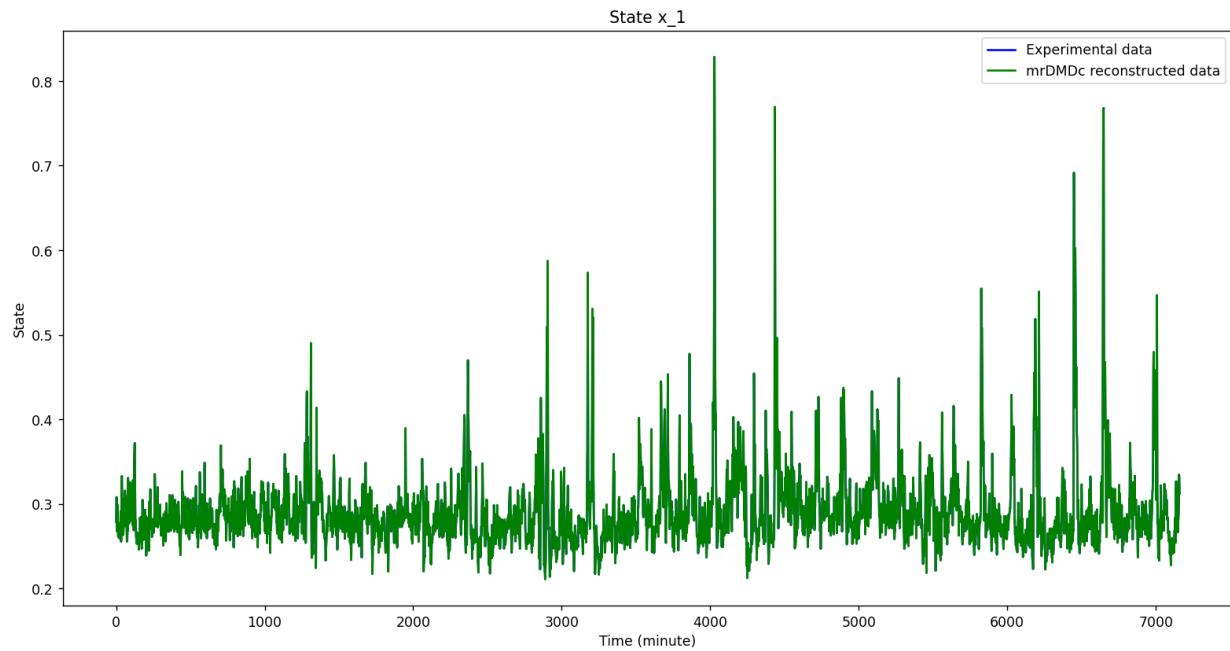


Figura 4.32

E di seguito viene presentato il confronto tra la ricostruzione e i dati sperimentali:



Zoom della ricostruzione:

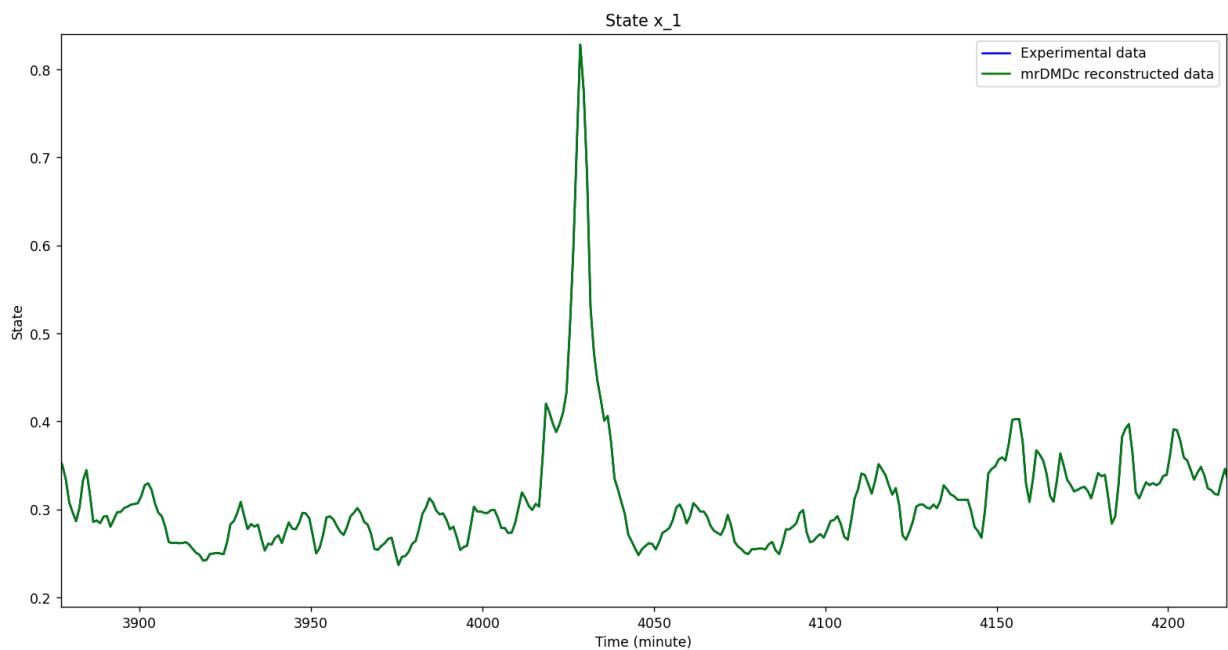


Figura 4.33

Errore tra dati sperimentali e ricostruzione:

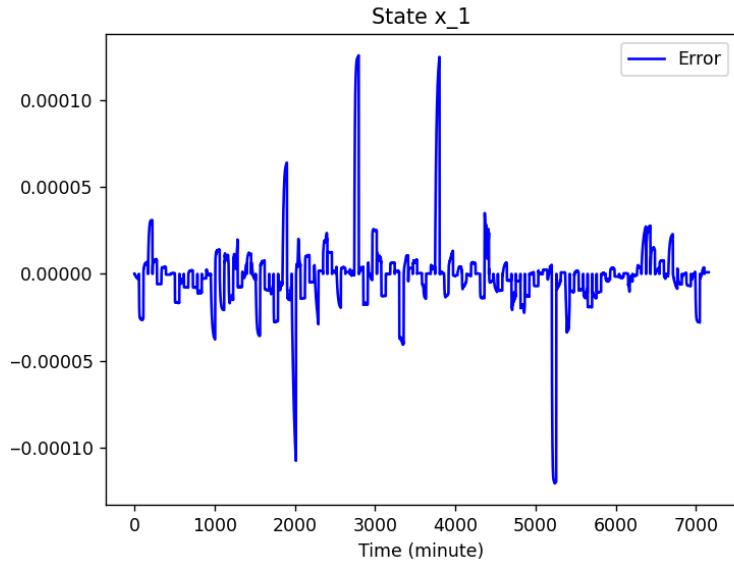


Figura 4.34

Risultati dei KPI calcolati considerando l'intero dataset e non solo una colonna:

MSE	4.61e-10
MAPE	3.77e-05
MAE	1.07e-05
RMSE	2.15e-05
R <sup>2</sup>	0.99

Filtrazione per le slow feature in ogni livello:

Filtration level 0	12%
Filtration level 1	8%
Filtration level 2	5%
Filtration level 3	2%
Filtration level 4	2%
Filtration level 5	2%
Filtration level 6	18%
Filtration level 7	15%

La ricostruzione risulta ottima, ma chiaramente quei segnali per la ricostruzione non sono fisicamente possibili. Inoltre, in caso avessimo deciso di troncare, il risultato non sarebbe più stato così ottimale. Vediamo il risultato finale della ricostruzione ponendo *svd rank = 0* e mantenendo gli altri parametri uguali:

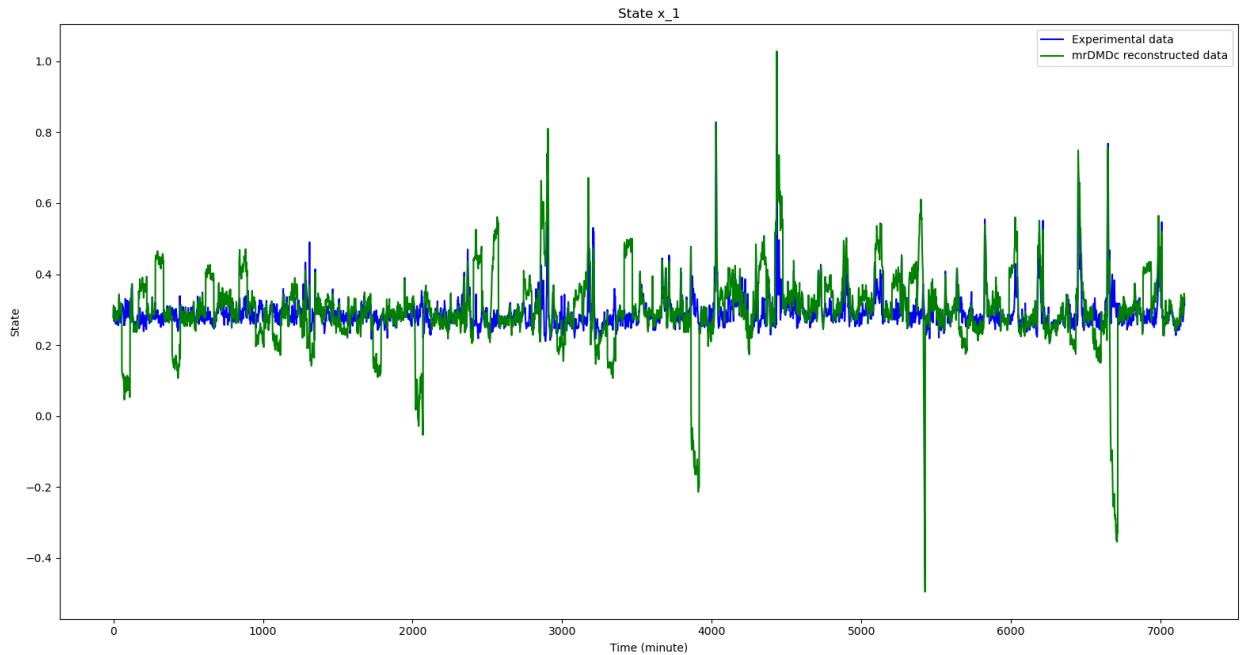


Figura 4.35

MSE	7.36e-03
MAPE	0.71
MAE	4.76e-02
RMSE	8.58e-02
R <sup>2</sup>	0.20

La filtrazione è nulla per tutti i livelli con l'attuale impostazione del parametro di scala degli slow.

Questi risultati sono chiaramente non ottimali, ed inoltre anch'essi soffrono della presenza di elevati segnali.

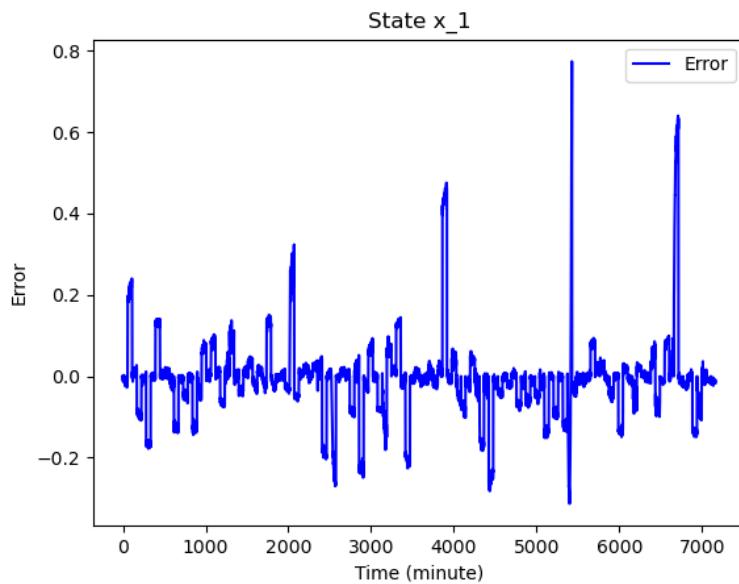


Figura 4.36

In conclusione, è possibile affermare che la ricostruzione eseguita tramite le dinamiche calcolate a passo  $dt = 1$  è scorretta, solo nel caso migliore in cui non vi è un'elevata filtrazione o un troncamento tramite SVD per  $Y$  o  $\Omega$  la ricostruzione risulta ottimale, ma ciò non toglie che la ricostruzione è sempre eseguita tramite segnali che hanno un'ampiezza molto elevata (e ciò rappresenta un assurdo fisico) che si compensano tra loro e permettono una giusta ricostruzione.

A questo punto passiamo all'analisi della ricostruzione con dinamiche calcolate con passo  $dt = step$ .

### 4.3.2 Ricostruzione a passo step

Verrà fatta un'analisi per i casi di studio presenti nel quale sarà eseguita la ricostruzione con le dinamiche calcolate a passo  $dt = step$ , che, come si potrà notare, avranno un andamento corretto.

#### 4.3.2.1 SRU

4.3.2.1.1 *svd rank = -1, max cycles = 10, slow feature scale = 15, reconstruction = 1:*

```
#svd_rank for DMDc object
'''param svd_rank: the rank for the truncation; If 0, the method computes the optimal rank and uses it for truncation;
if positive integer, the method uses the argument for the truncation; if float between 0 and 1, the rank is the number
of the biggest singular values that are needed to reach the 'energy' specified by `svd_rank`; if -1, the method does
not compute truncation.'''
svd_rank_set = -1

#max_cycles for mrDMDc
max_cycles_set = 10
'''

there is a possibility that if the reconstruction is not good (an example if the reconstructed system is unstable),
changing the parameter will work.
No way has been figured out to decide whether to increase or decrease it in the event of a bad reconstruction
'''

#index of filtration, needs to set the size of rho
slow_feature_scale = 15

'''selection of the reconstruction type
1 Reconstruction with A and B dt = step
2 Reconstruction with A and B dt = 1 (trasformation of A and B with Harold library)
3 Reconstruction with forced response (work in progress)
'''
reconstruction = 1
```

Codice 4.2

Rappresentazione dell'evoluzione del primo stato nel dataset sperimentale in figura:

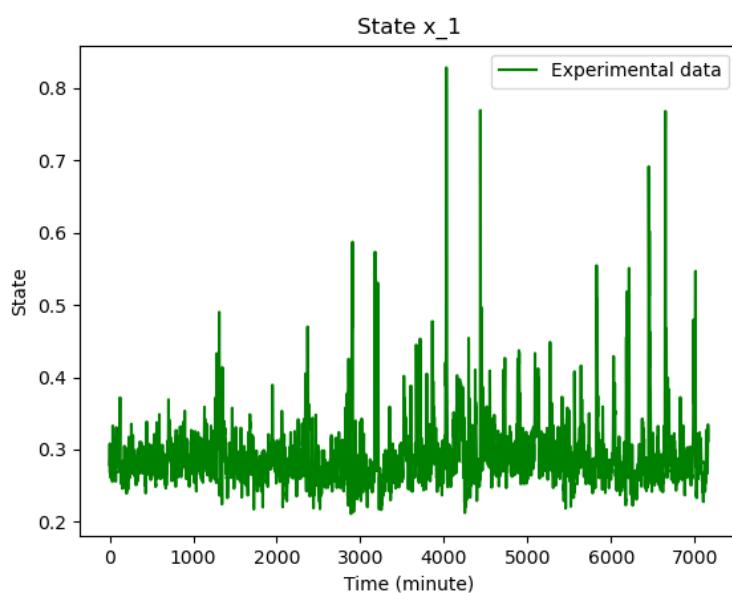
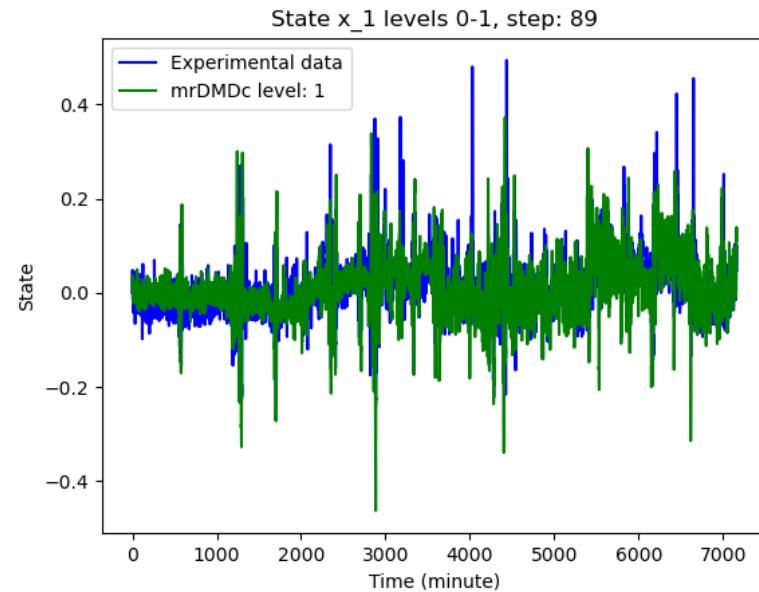
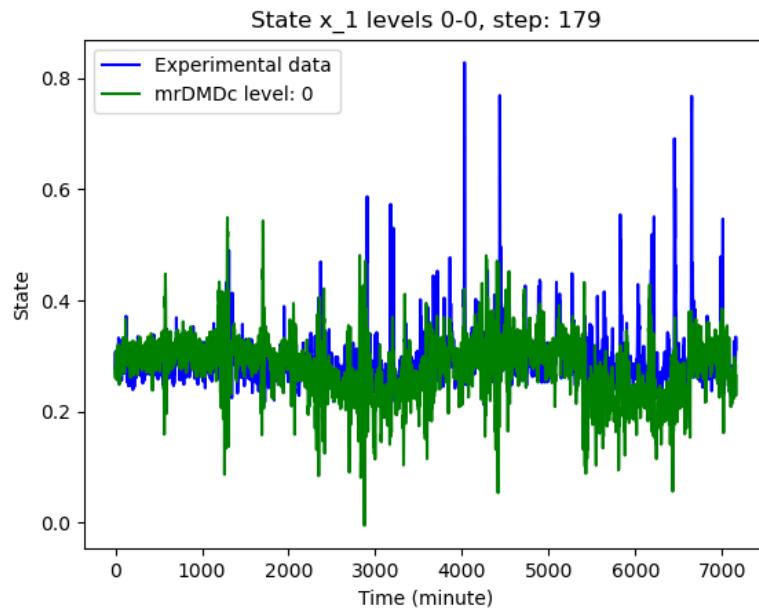
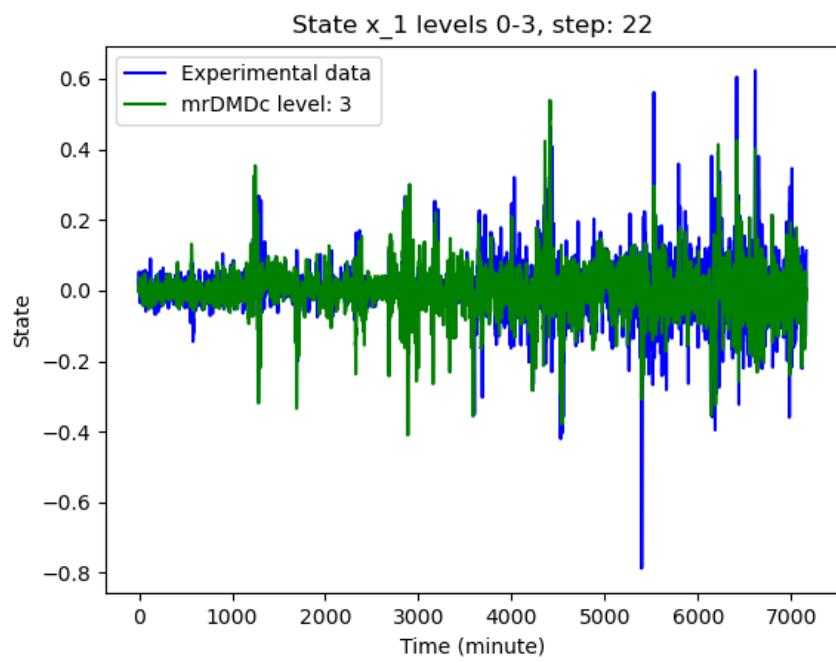
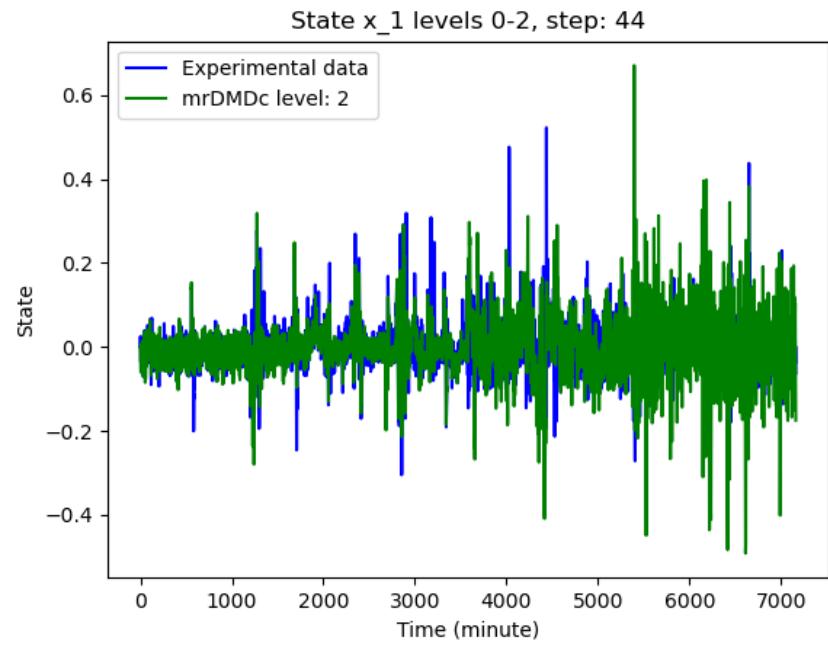


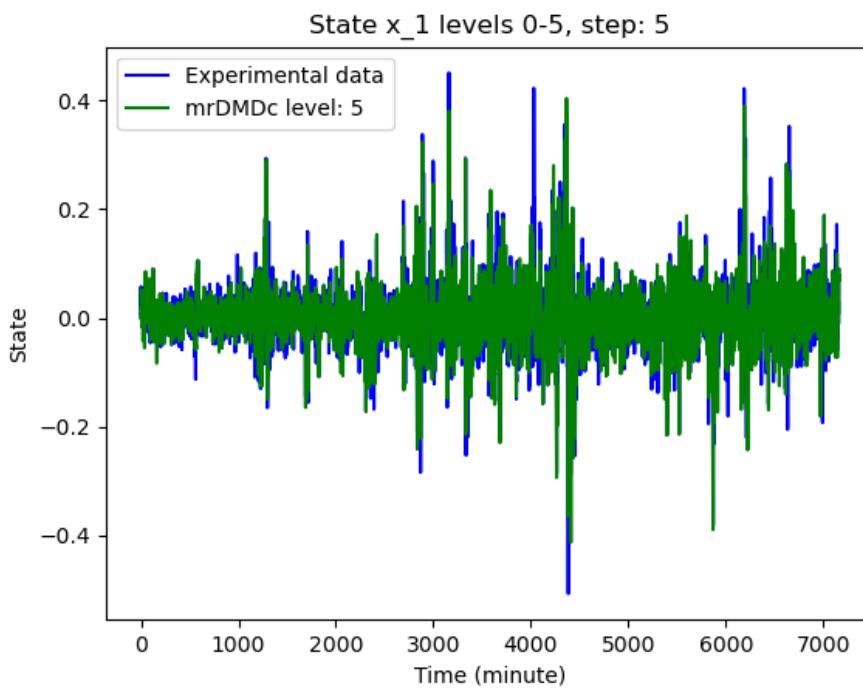
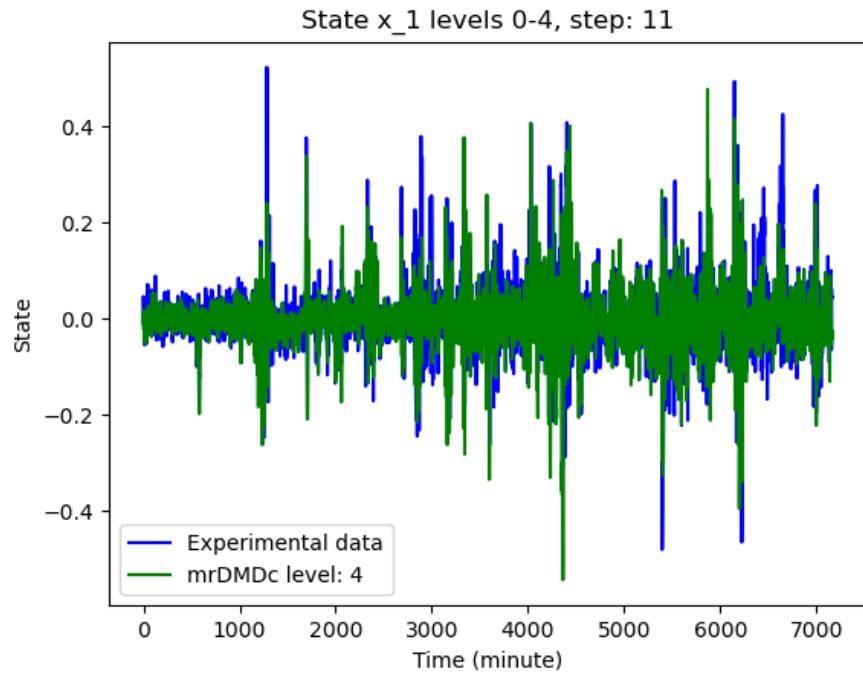
Figura 4.37

Di seguito la ricostruzione di ogni livello eseguita dal mrDMDc.

Grafici di confronto tra i dati sperimentali e i livelli mrDMDc. Ai dati sperimentali viene sottratto il livello ricostruito.







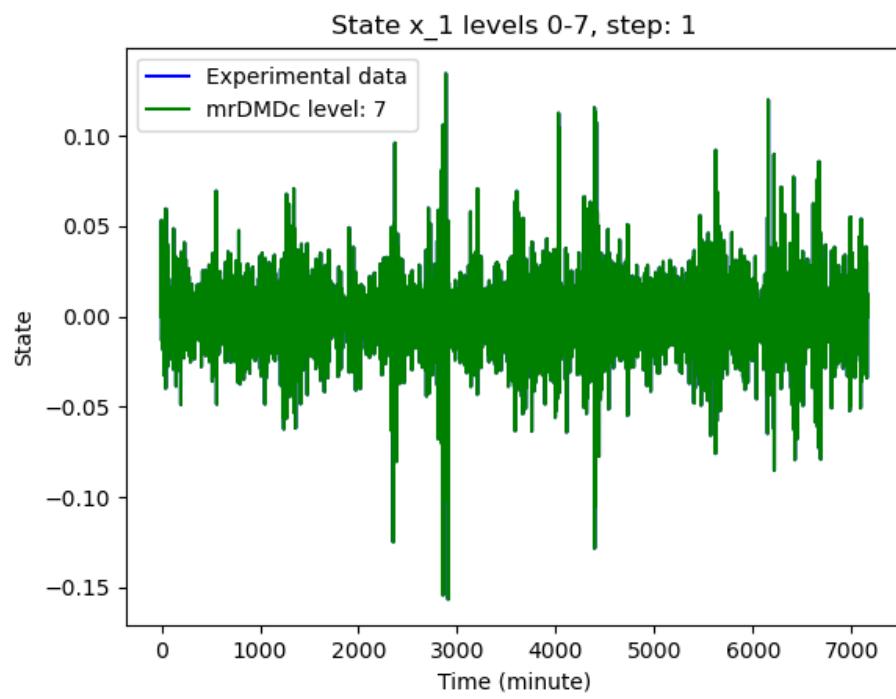
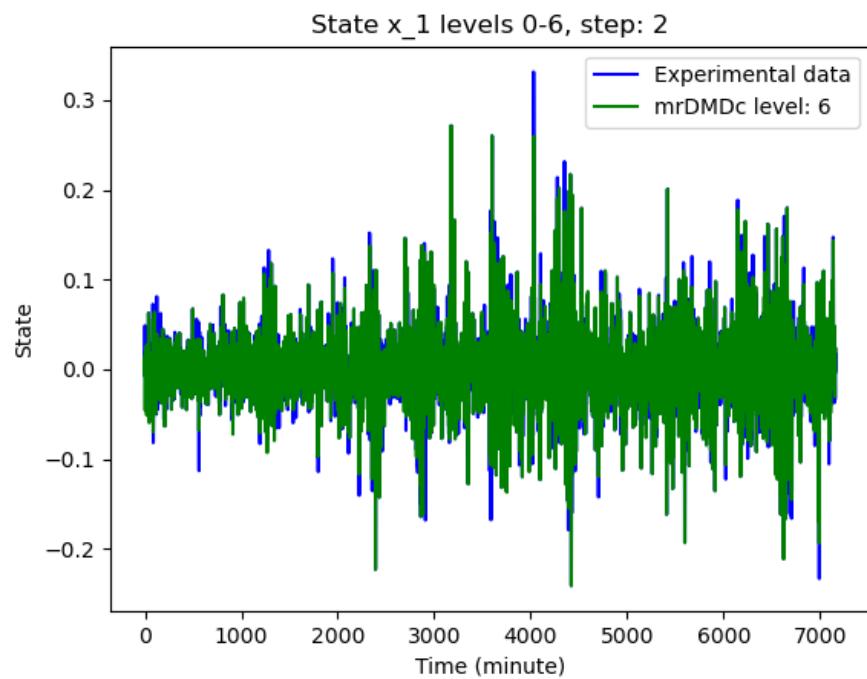


Figura 4.38

Contributi di ogni singolo livello:

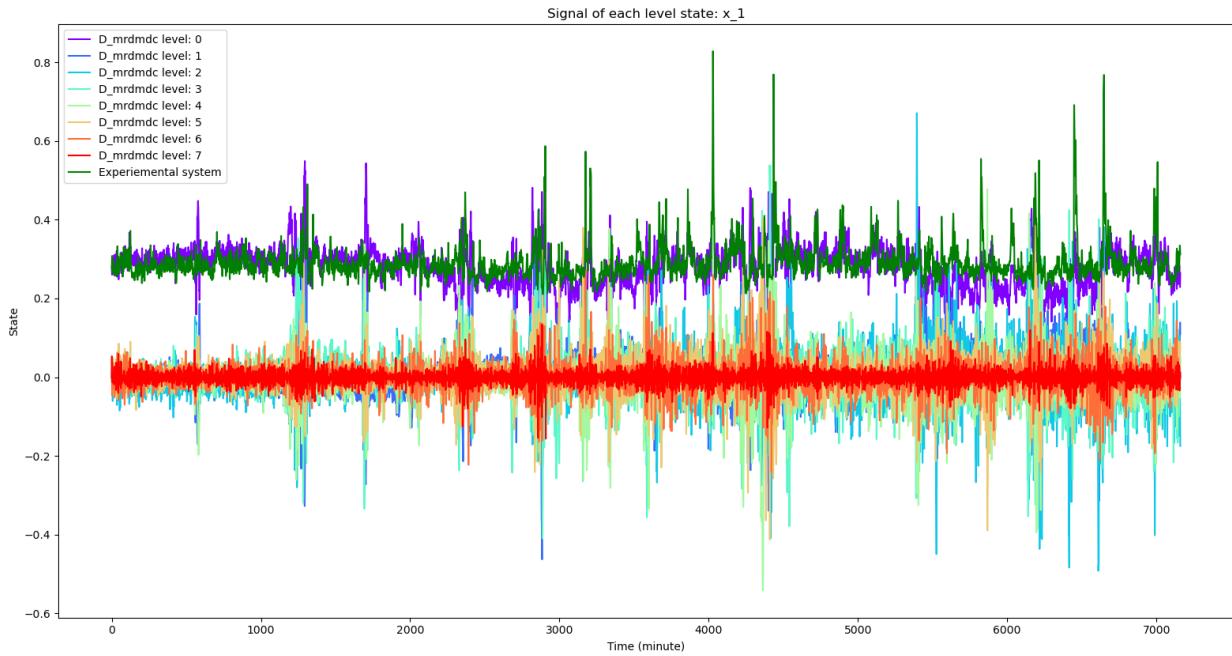
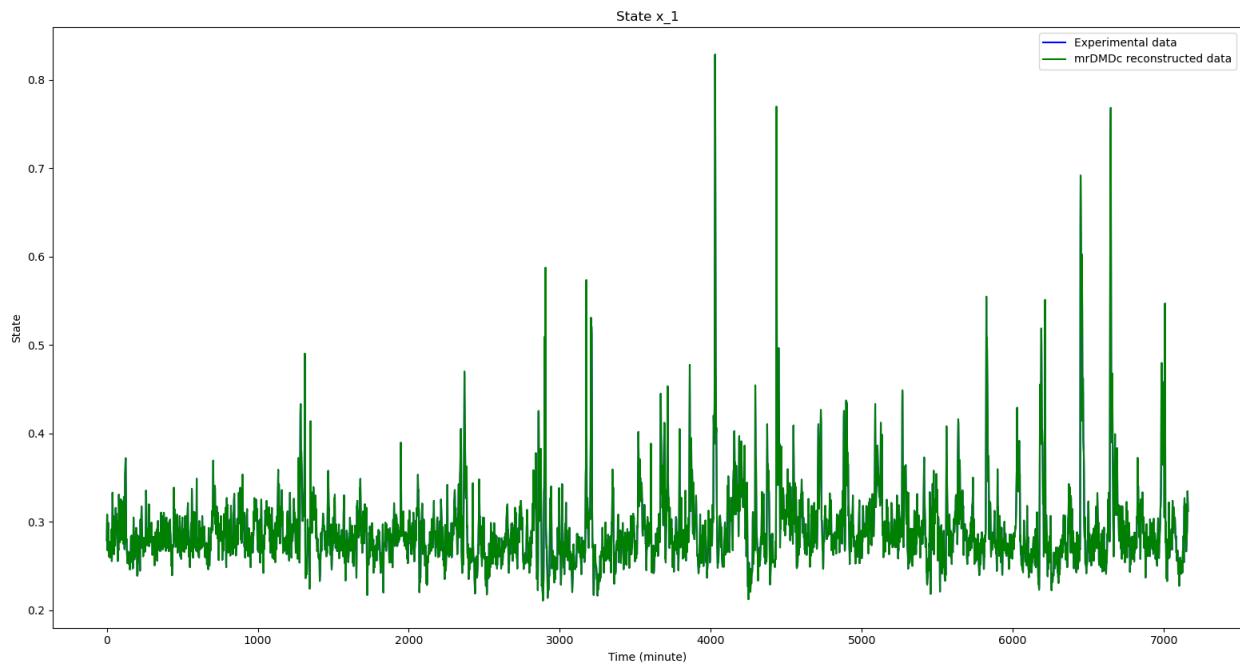


Figura 4.39

Si nota chiaramente come la ricostruzione con le dinamiche calcolate a  $dt = step$  non solo non crescano all'aumentare del numero dei livelli, ma rispetta anche quelle che sono le previsioni di estrazioni delle dinamiche, cioè il livello 0 sottrae il valor medio del segnale e tutti gli altri livelli catturano le dinamiche più veloci per la ricostruzione.

Grafico della ricostruzione finale:



Zoom della ricostruzione:

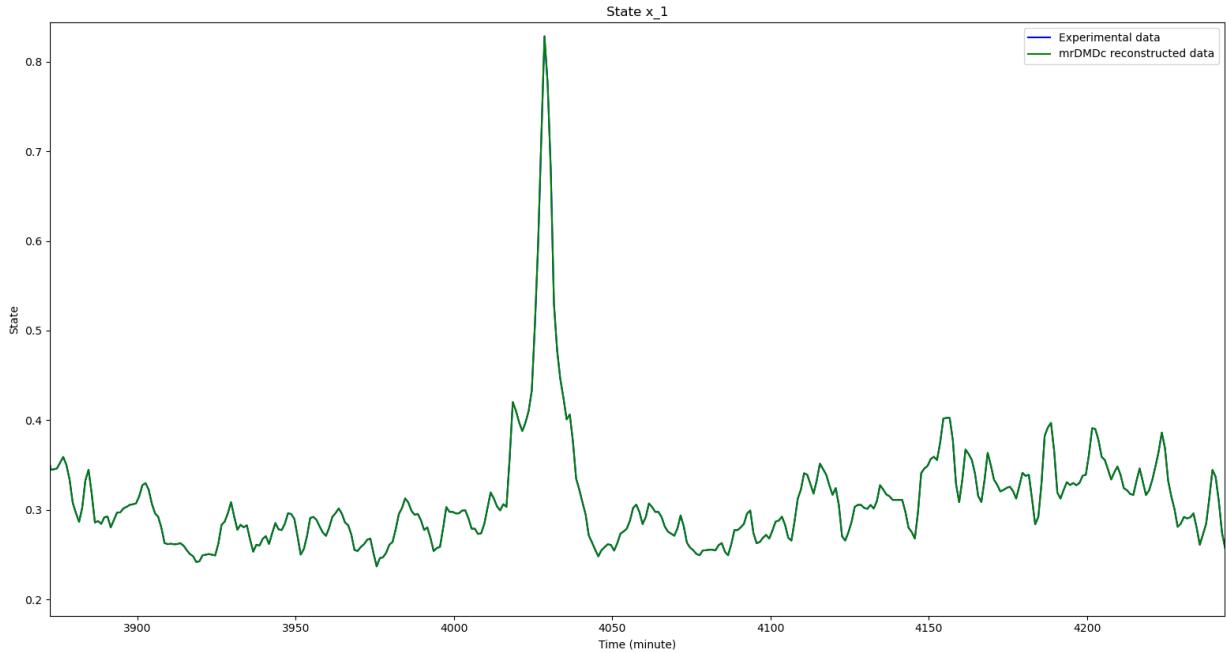


Figura 4.40

Errore tra dati sperimentali e ricostruzione:

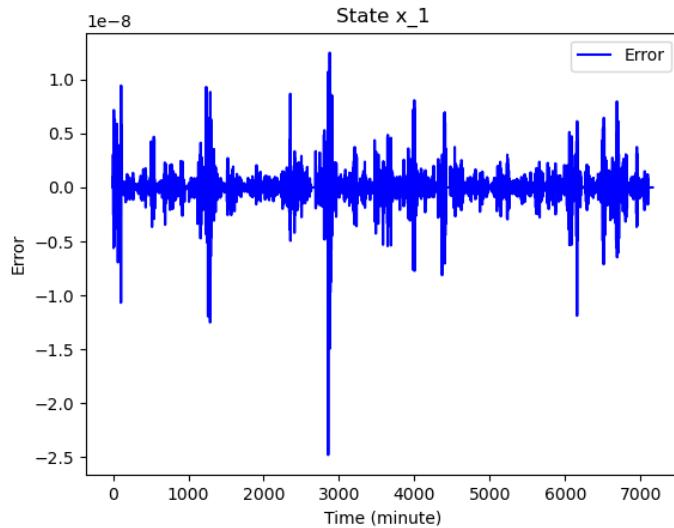


Figura 4.41

Risultati dei KPI calcolati considerando l'intero dataset e non solo una colonna:

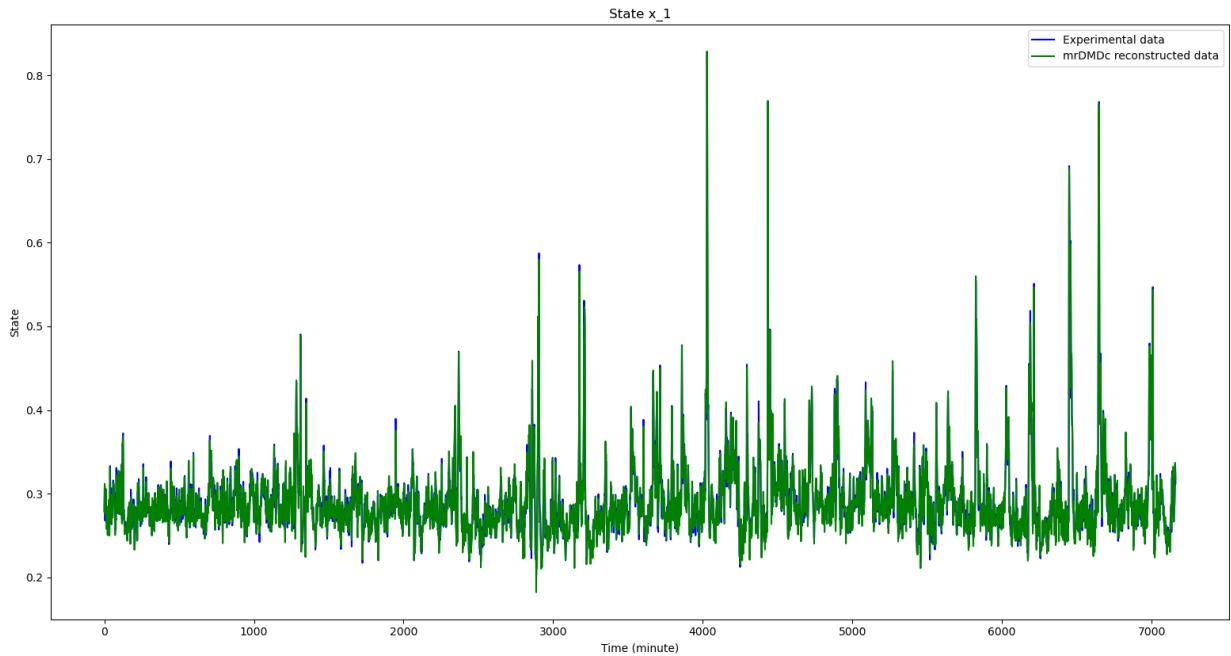
MSE	8.18e-19
MAPE	1.19e-09
MAE	3.45e-10
RMSE	9.04e-10
R <sup>2</sup>	0.9999

Filtrazione per le slow feature in ogni livello:

Filtration level 0	13%
Filtration level 1	10%
Filtration level 2	10%
Filtration level 3	28%
Filtration level 4	29%
Filtration level 5	8%
Filtration level 6	30%
Filtration level 7	38%

Come si può notare la ricostruzione risulta anche migliore della precedente con le dinamiche calcolate a passo  $dt = 1$  e non hanno il problema di quei segnali ad ampiezza elevata.

Vengono proposti i risultati anche con il parametro  $svd\ rank = 0$  mantenendo gli altri parametri uguali:



Zoom della ricostruzione:

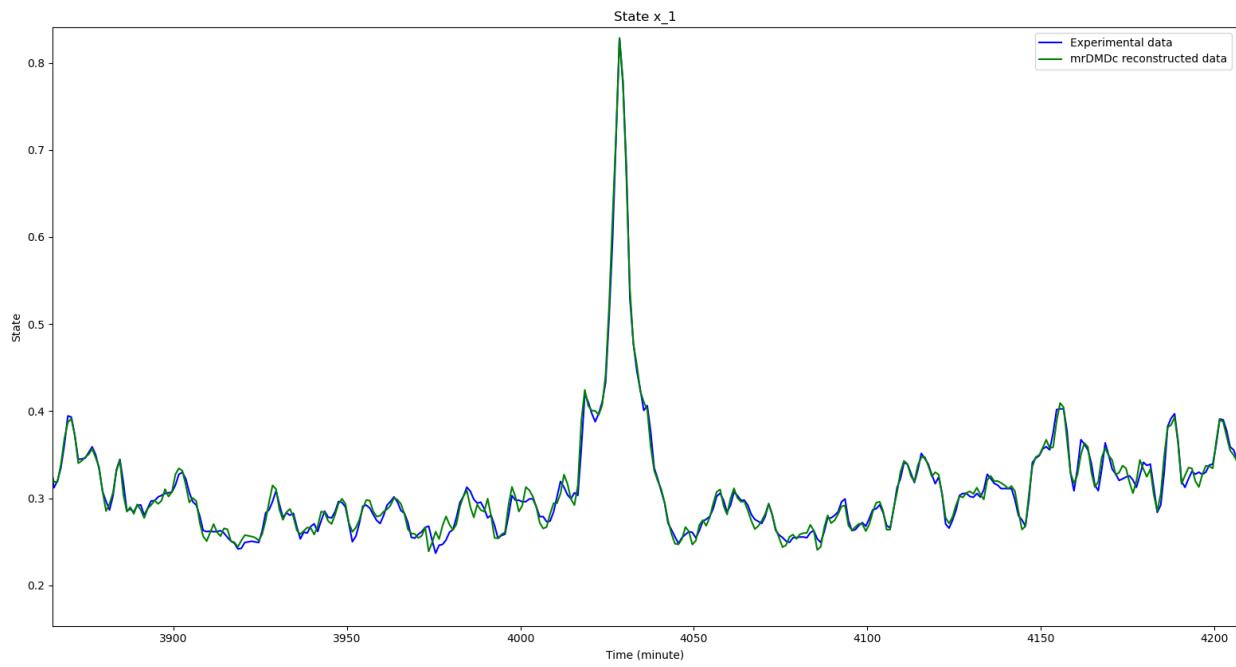


Figura 4.42

MSE	4.61e-05
MAPE	1.73e-02
MAE	5.02e-03
RMSE	6.79e-03
R^2	0.97

Non vi è stata alcune filtrazione in quanto, tramite il troncamento, molte dinamiche veloci sono state eliminate.

#### 4.3.2.1.2 *svd rank = 0, max cycles = 10, slow feature scale = 10, reconstruction = 1:*

Presentiamo un'ulteriore risultato con i seguenti parametri per vedere le prestazioni dell'algoritmo, modificando *svd rank* e *slow feature scale*, che settati rispettivamente a 0 (significa che viene eseguito il troncamento su  $\Omega$  ed Y) e 10 (riduzione del raggio  $\rho$ ) permettono una maggiore filtrazione:

```
#svd_rank for DMDc object
'''param svd_rank: the rank for the truncation; If 0, the method computes the optimal rank and uses it for truncation;
if positive interger, the method uses the argument for the truncation; if float between 0 and 1, the rank is the number
of the biggest singular values that are needed to reach the 'energy' specified by `svd_rank`; if -1, the method does
not compute truncation.'''  
svd_rank_set = 0

#max_cycles for mrDMDc
max_cycles_set = 10
...
there is a possibility that if the reconstruction is not good (an example if the reconstructed system is unstable),
changing the parameter will work.
No way has been figured out to decide whether to increase or decrease it in the event of a bad reconstruction
...

#index of filtration, needs to set the size of rho
slow_feature_scale = 10

'''selection of the reconstruction type
1 Reconstruction with A and B dt = step
2 Reconstruction with A and B dt = 1 (trasformation of A and B with Harold library)
3 Reconstruction with forced response (work in progress)
...
reconstruction = 1
```

Codice 4.3

Grafico dei livelli ricostruiti:

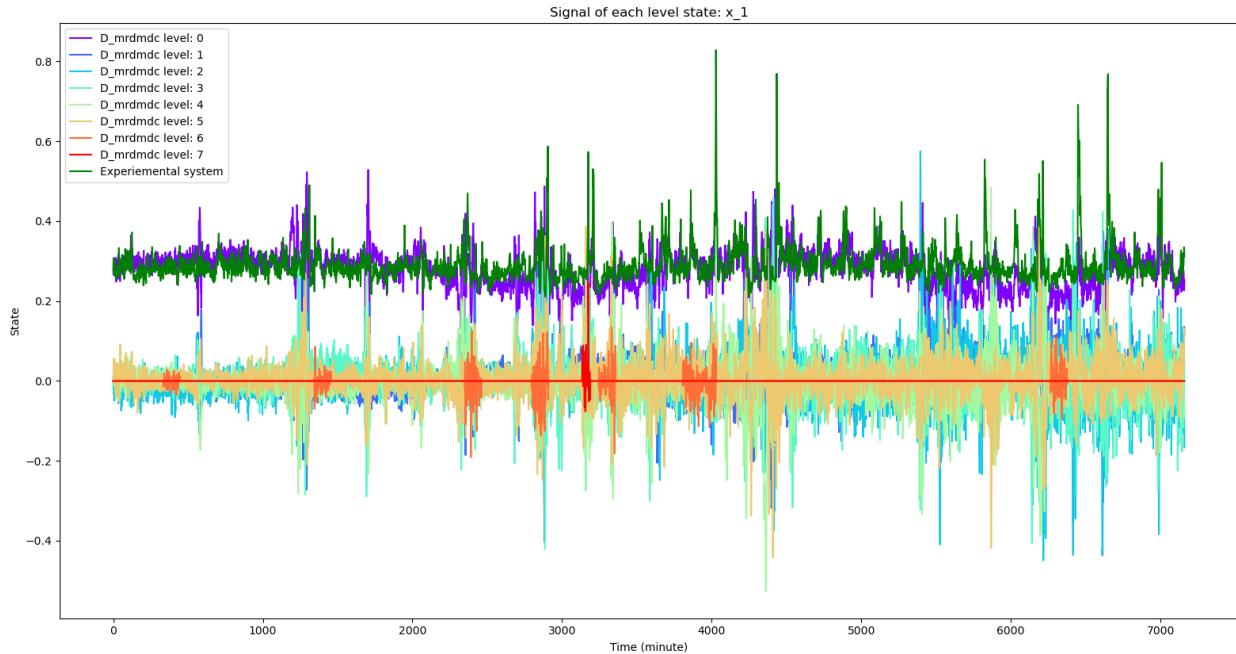
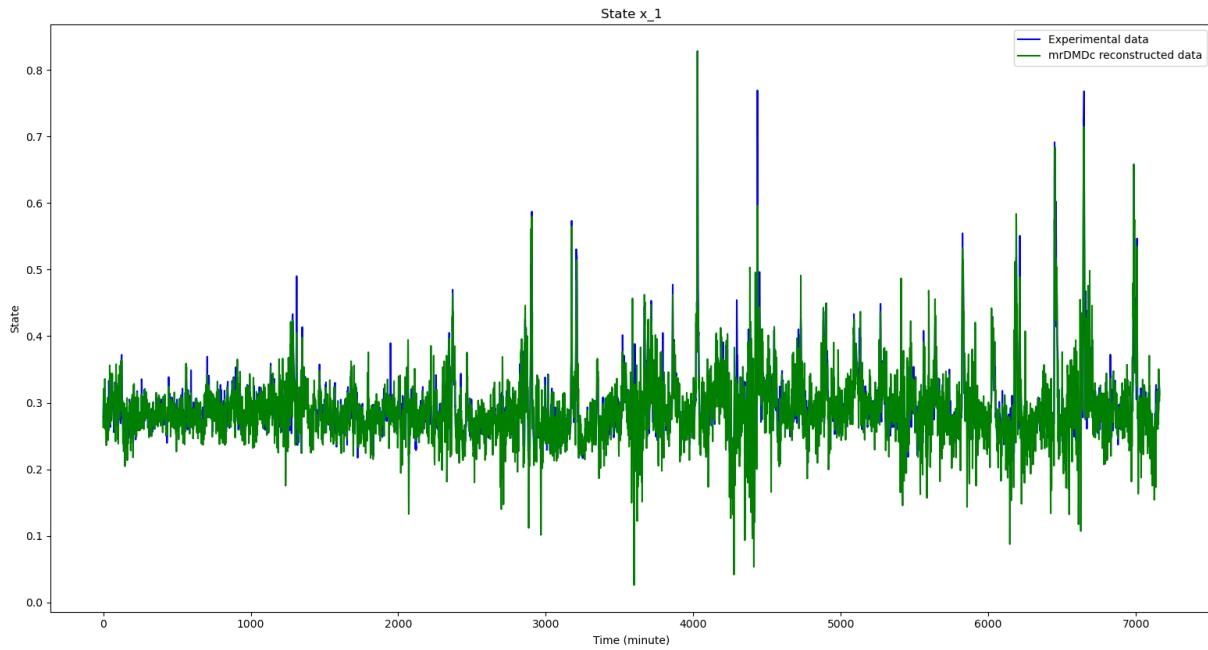


Figura 4.43

Come si può notare dal grafico dei livelli ricostruiti, il livello 6 e 7 sono quasi del tutto nulli, questo è dovuto al fatto che avendo ridotto il raggio per la filtrazione, ed essendo i livelli finali formati da dinamiche veloci, tutte le dinamiche che sarebbero dovute appartenere a questo livello sono state filtrate.

Grafico della ricostruzione:



Zoom della ricostruzione:

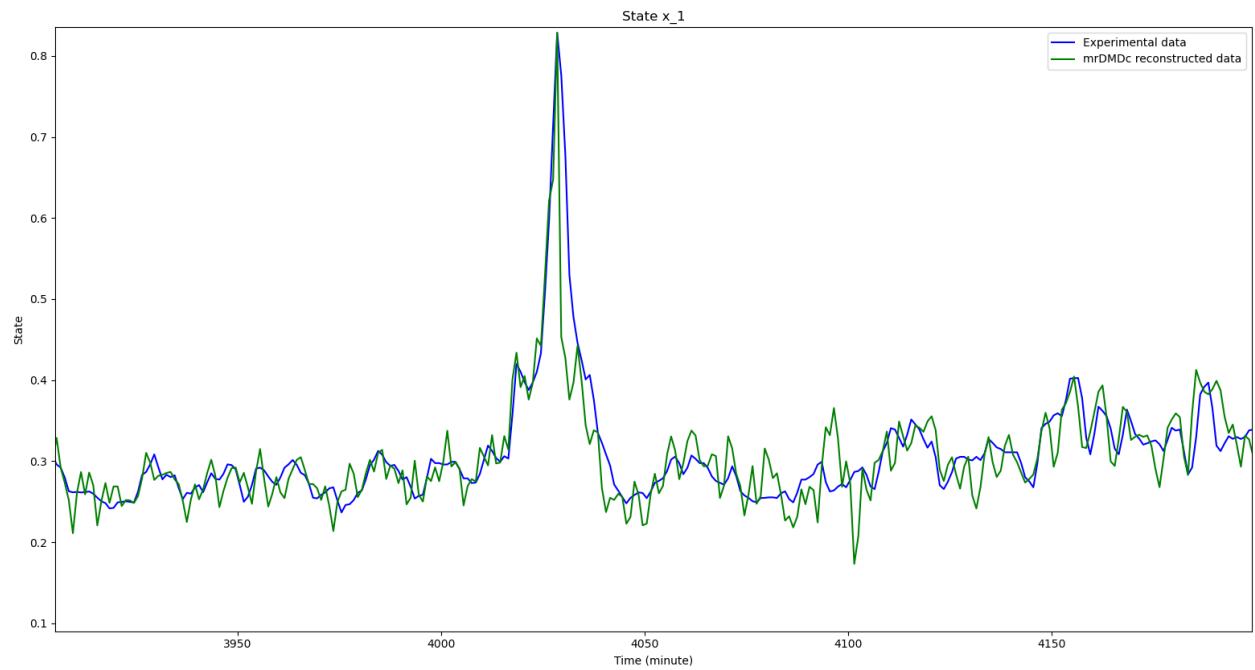


Figura 4.44

Errore tra dati sperimentali e dati ricostruiti:

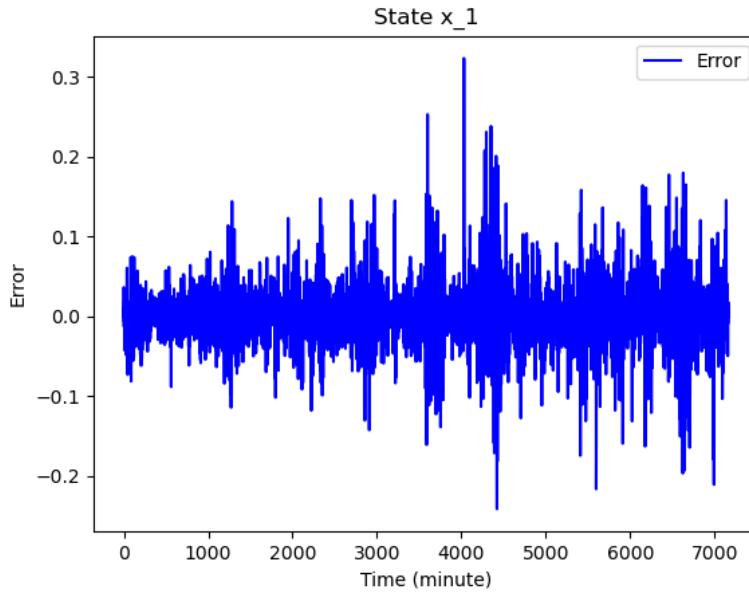


Figura 4.45

Risultati dei KPI calcolati considerando l'intero dataset e non solo una colonna:

MSE	4.59e-04
MAPE	0.13
MAE	1.43e-02
RMSE	2.14e-02
R^2	0.79

Filtrazione per le slow feature in ogni livello:

Filtration level 0	10%
Filtration level 1	4%
Filtration level 2	5%
Filtration level 3	9%
Filtration level 4	10%
Filtration level 5	32%
Filtration level 6	97%
Filtration level 7	99%

In conclusione, possiamo affermare che con questo tipo di ricostruzione l'analisi multi-resolution sembra corretta, e che dunque verrà considerato questo come metodo di ricostruzione per i successivi casi di studio che verranno affrontati.

#### 4.3.2.2 Sistema Sintetico con 5 Ingressi con ritardi dal Sistema SRU

Sono stati selezionati:

- D (serie temporale di stati) dal dataset generato dal sistema sintetico ad autovalori complessi descritto nel capitolo 3.
- U (serie temporale di ingressi) dal dataset SRU.

##### 4.3.2.2.1 $\text{svd rank} = -1, \text{max cycles} = 20, \text{slow feature scale} = 3, \text{reconstruction} = 1$ :

```
#svd_rank for DMDc object
'''param svd_rank: the rank for the truncation; If 0, the method computes the optimal rank and uses it for truncation;
if positive integer, the method uses the argument for the truncation; if float between 0 and 1, the rank is the number
of the biggest singular values that are needed to reach the 'energy' specified by `svd_rank`; if -1, the method does
not compute truncation.'''  
svd_rank_set = -1

#max_cycles for mrDMDc
max_cycles_set = 20
'''  
there is a possibility that if the reconstruction is not good (an example if the reconstructed system is unstable),
changing the parameter will work.  
No way has been figured out to decide whether to increase or decrease it in the event of a bad reconstruction  
'''  
  
#index of filtration, needs to set the size of rho
slow_feature_scale = 3

'''selection of the reconstruction type
1 Reconstruction with A and B dt = step
2 Reconstruction with A and B dt = 1 (trasformation of A and B with Harold library)
3 Reconstruction with forced response (work in progress)
'''  
reconstruction = 1
```

Codice 4.4

Rappresentazione del dataset:

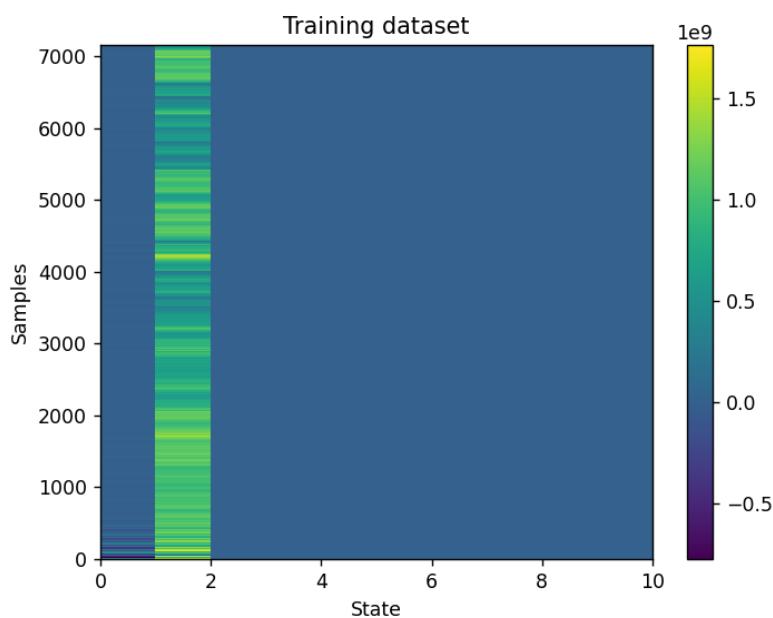


Figura 4.46

Rappresentazione dell'evoluzione del secondo stato nel dataset sperimentale:

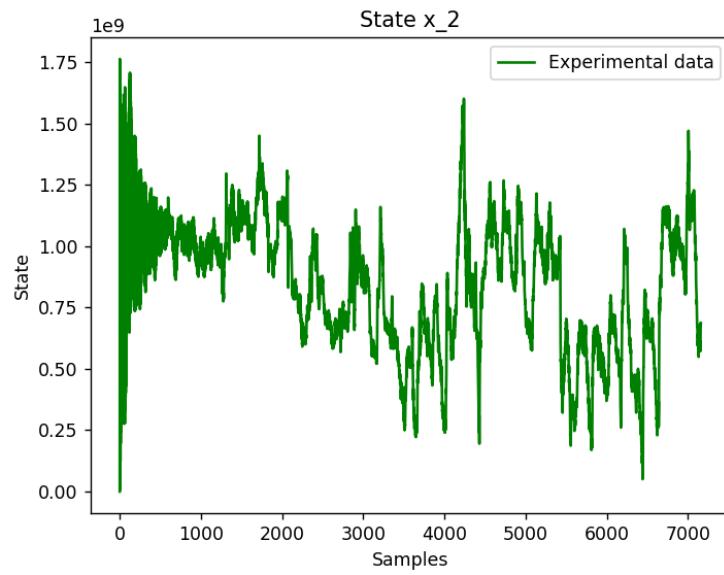
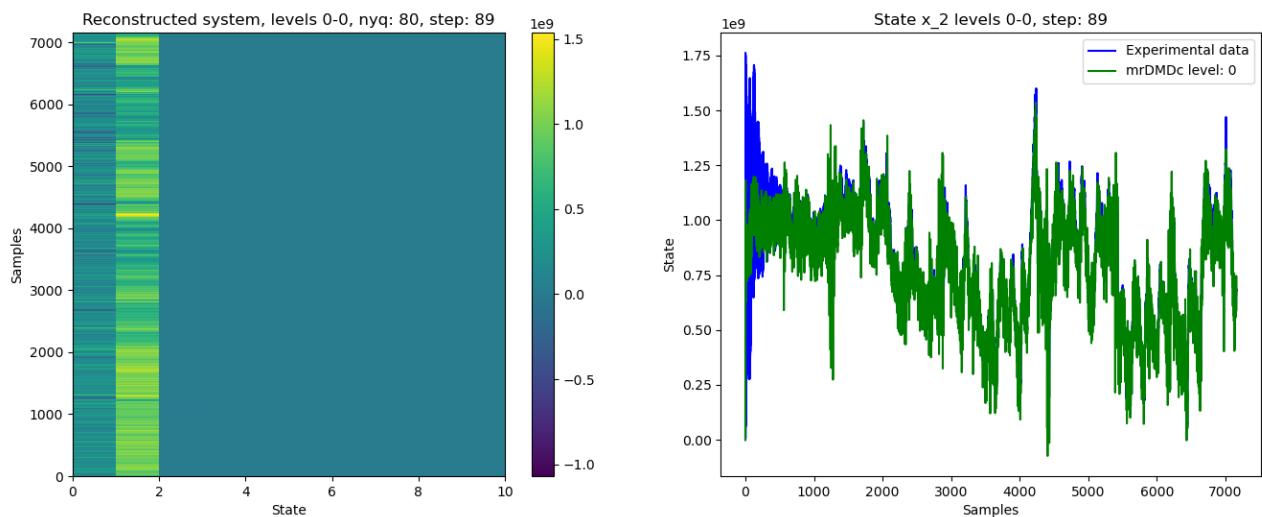
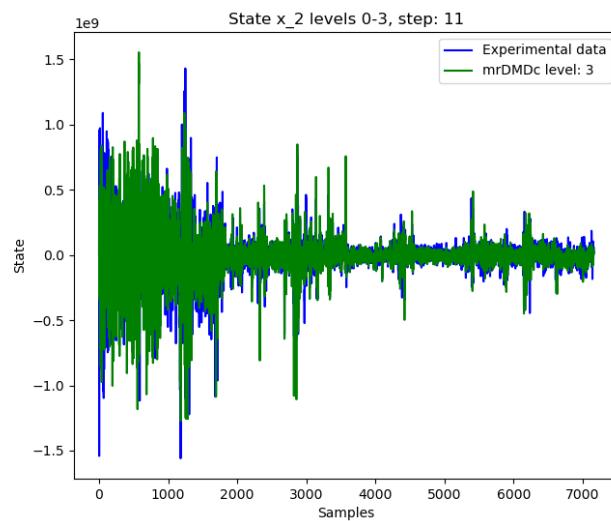
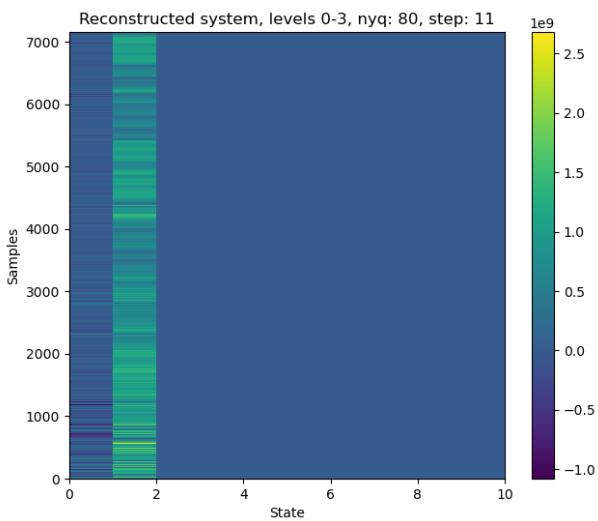
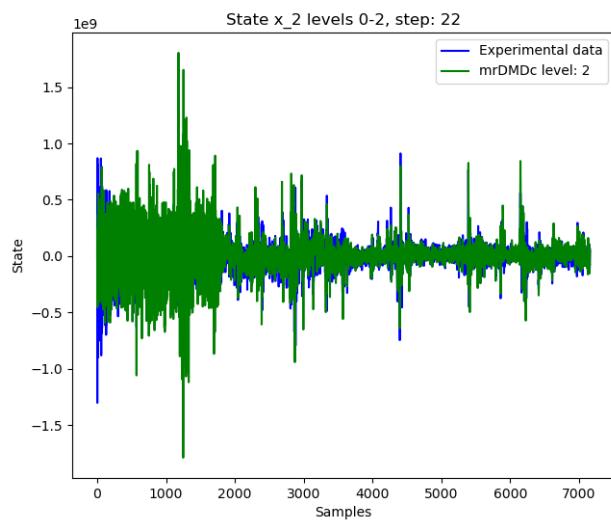
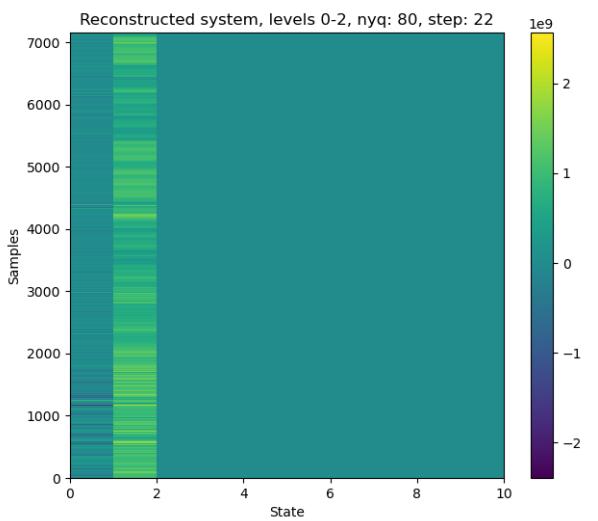
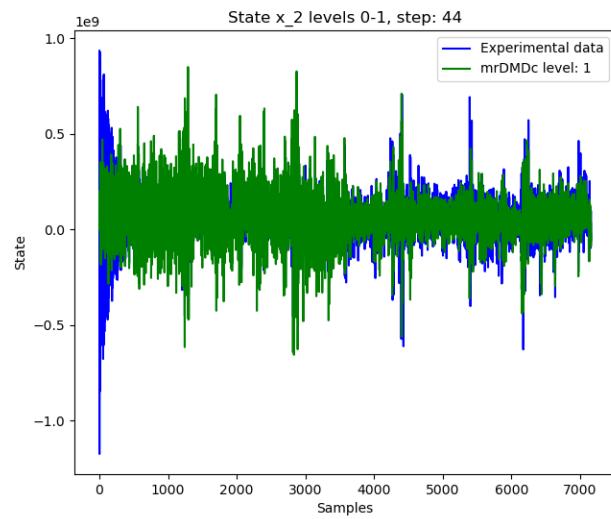
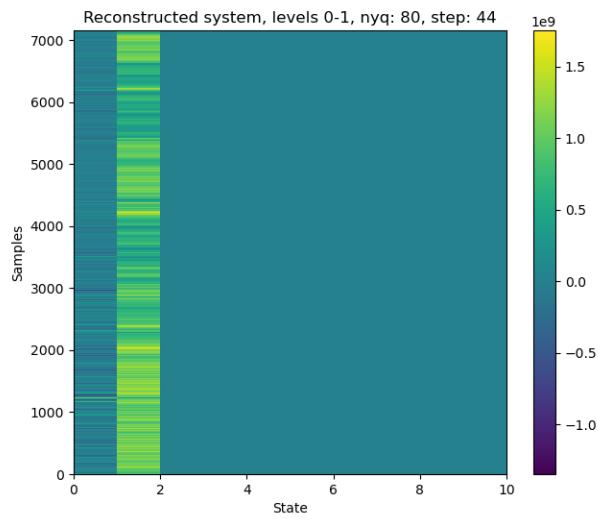


Figura 4.47

Ricostruzione di ogni livello:





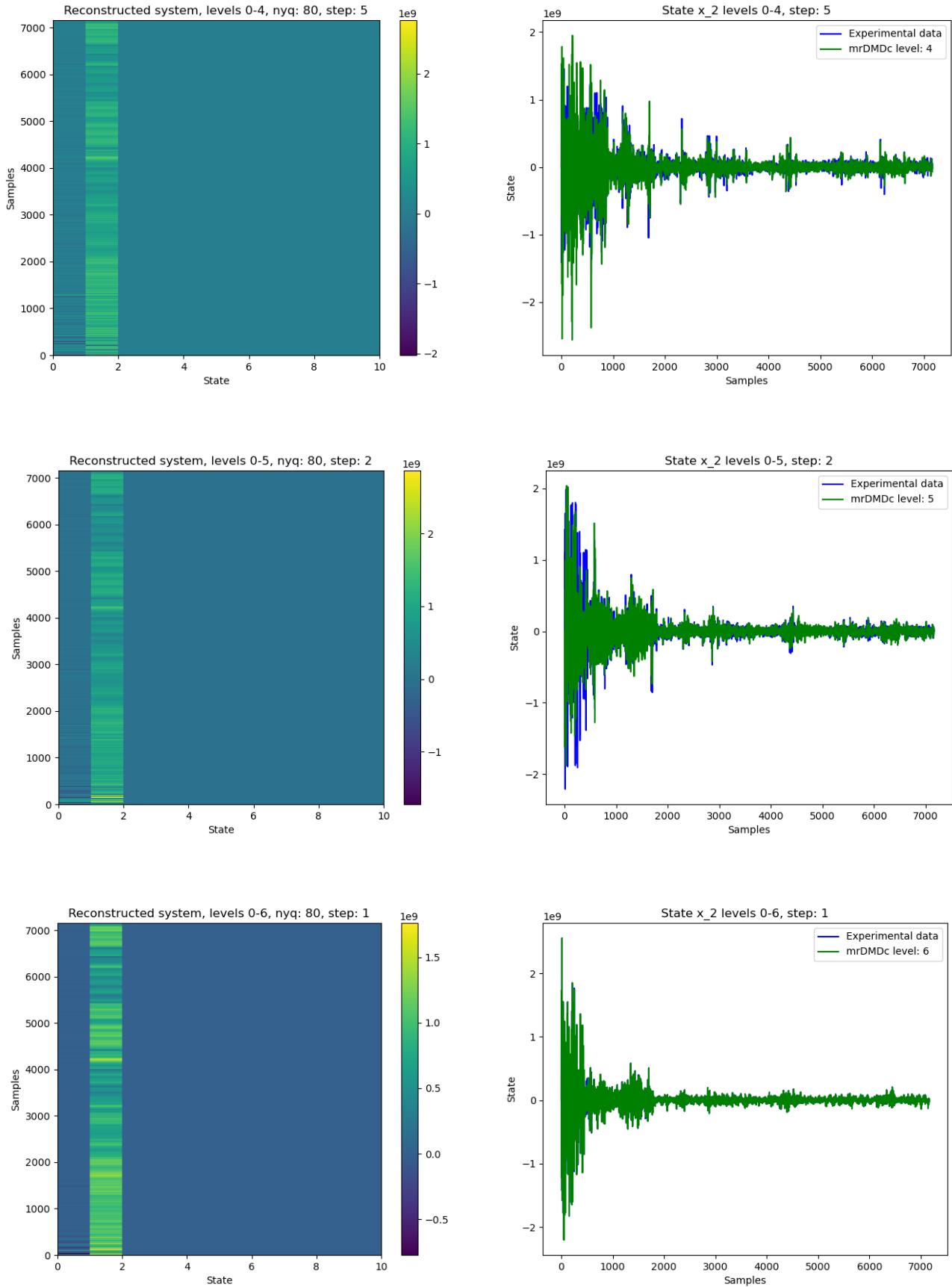


Figura 4.48

Contributi di ogni singolo livello: del secondo stato:

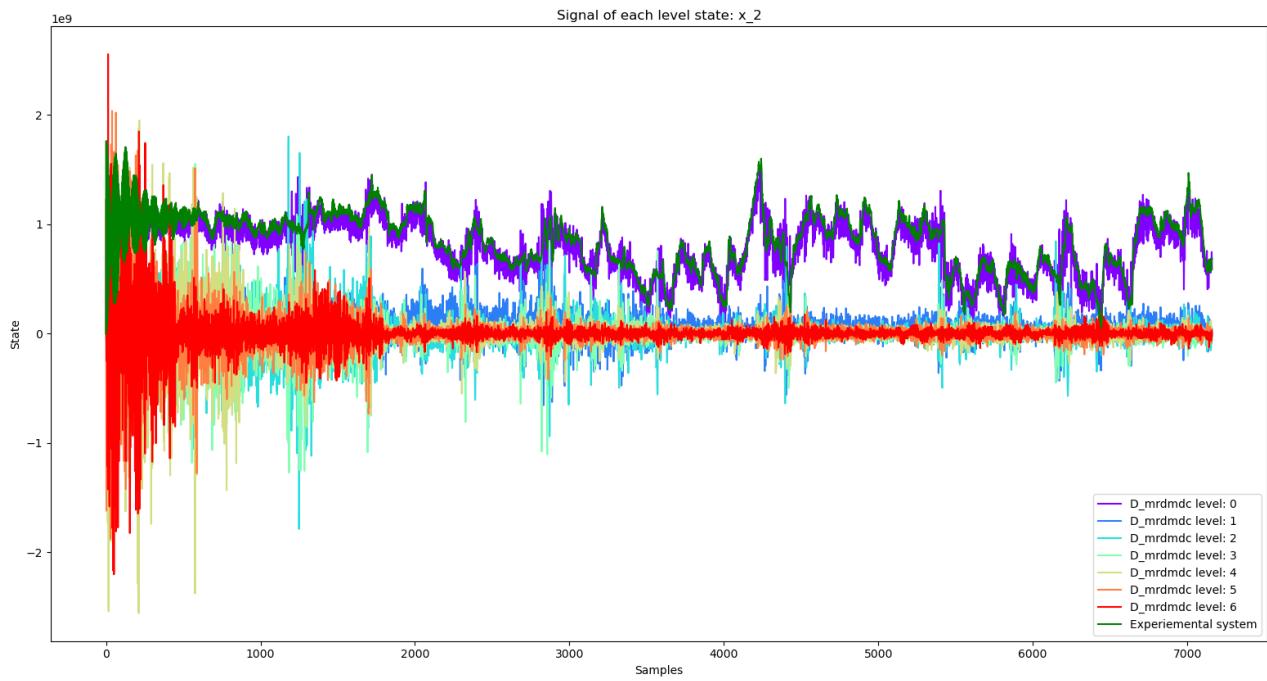


Figura 4.49

Grafico della ricostruzione finale:

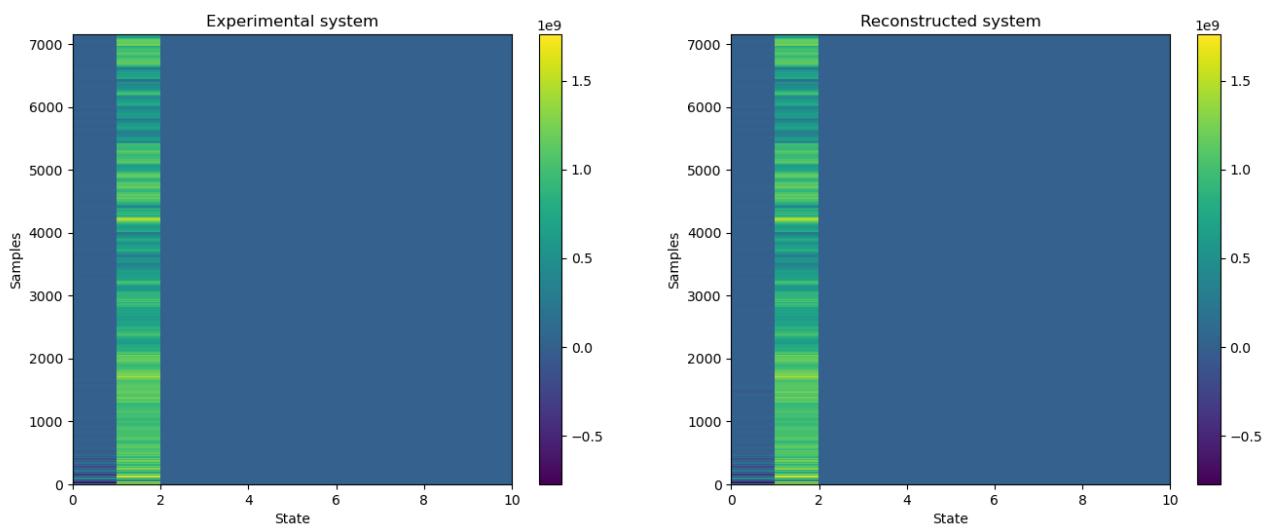
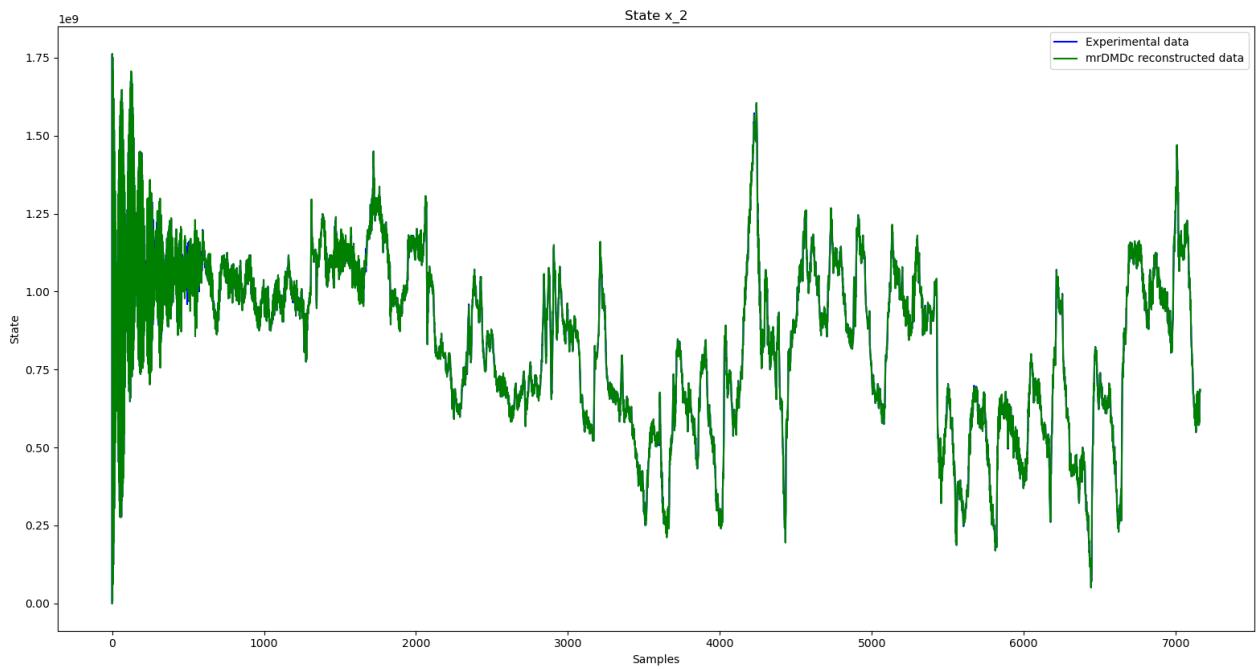
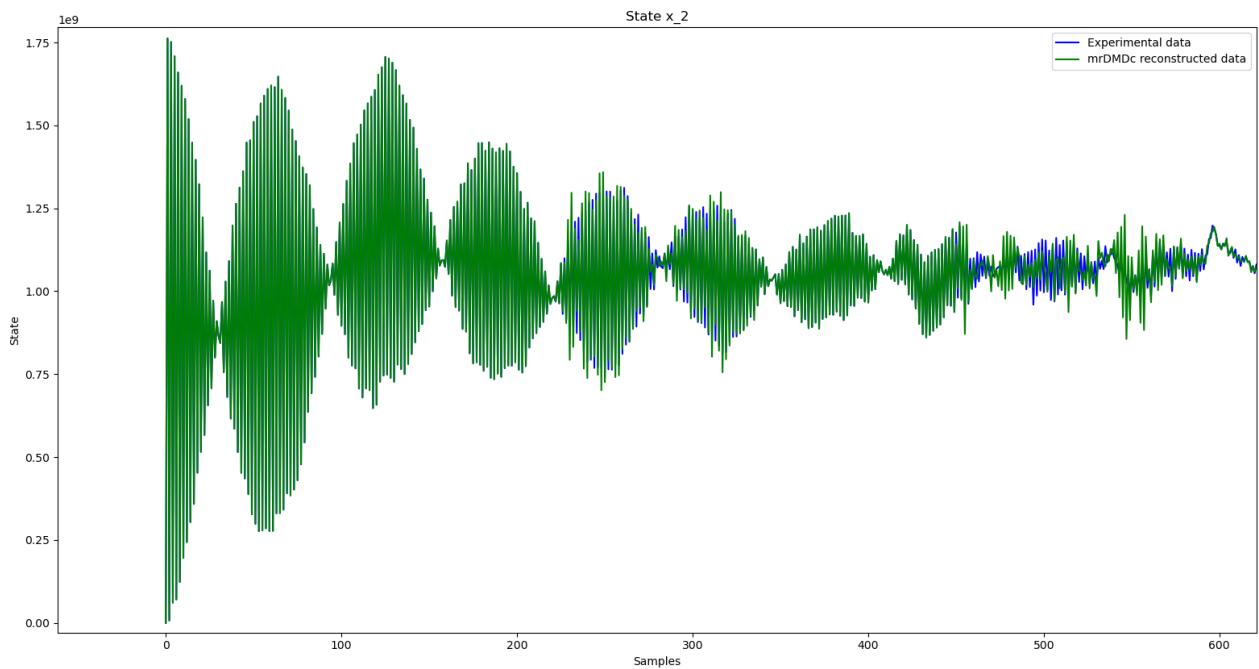


Figura 4.50

Ricostruzione dell'evoluzione del secondo stato:



Zoom della ricostruzione:



Zoom della ricostruzione:

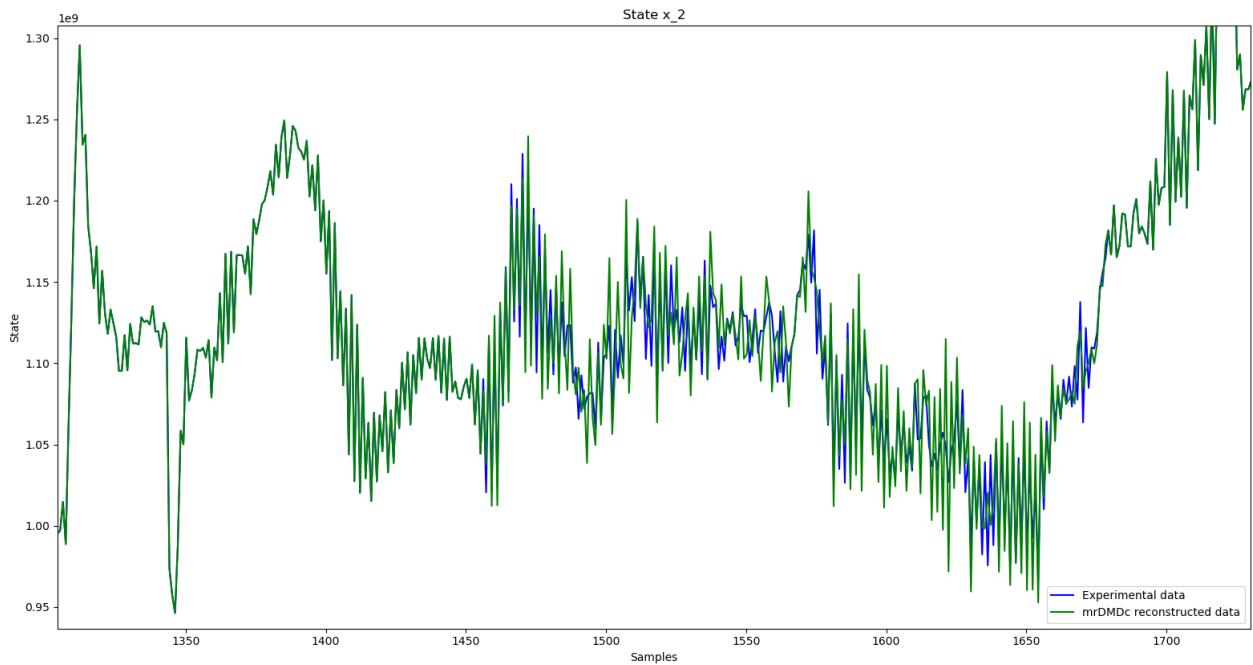


Figura 4.51

Errore tra dati sperimentali e ricostruzione:

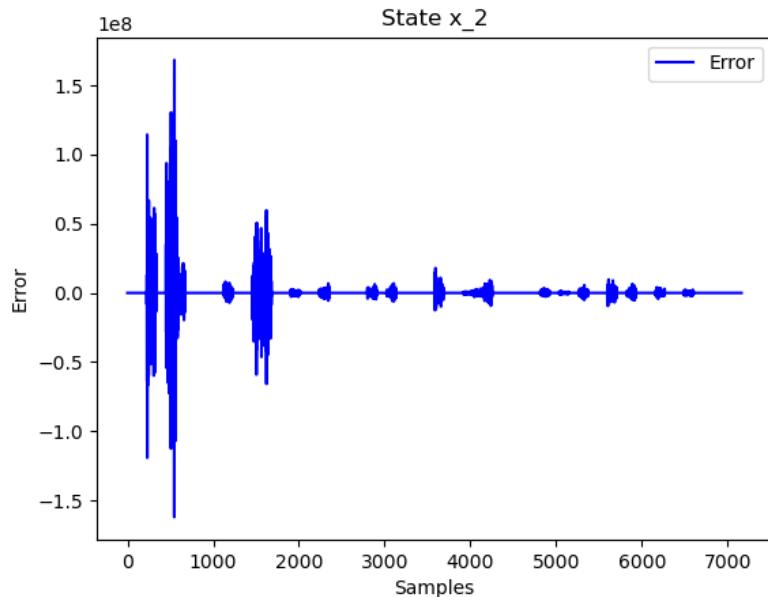


Figura 4.52

Risultati dei KPI calcolati considerando l'intero dataset e non solo una colonna:

MSE	1.32e+13
MAPE	0.34
MAE	3.79e+05
RMSE	3.63e+06
R^2	0.98

Filtrazione per le slow feature in ogni livello:

Filtration level 0	10%
Filtration level 1	0%
Filtration level 2	2%
Filtration level 3	1%
Filtration level 4	1%
Filtration level 5	15%
Filtration level 6	5%

#### 4.3.2.2.2 *svd rank = 0, max cycles = 8, slow feature scale = 3, reconstruction = 1:*

Anche in questo caso come per il dataset SRU si è aumentata la percentuale di filtrazione dei dati utilizzati per la modellazione del sistema, per comprendere quali sono le prestazioni dell'algoritmo.

```
#svd_rank for DMDc object
'''param svd_rank: the rank for the truncation; If 0, the method computes the optimal rank and uses it for truncation;
| if positive interger, the method uses the argument for the truncation; if float between 0 and 1, the rank is the number
| of the biggest singular values that are needed to reach the 'energy' specified by `svd_rank`; if -1, the method does
| not compute truncation.'''  

svd_rank_set = 0  
  

#max_cycles for mrDMDc
max_cycles_set = 8
'''  

there is a possibility that if the reconstruction is not good (an example if the reconstructed system is unstable),
changing the parameter will work.  

No way has been figured out to decide whether to increase or decrease it in the event of a bad reconstruction
'''  
  

#index of filtration, needs to set the size of rho
slow_feature_scale = 3  
  

'''selection of the reconstruction type
1 Reconstruction with A and B dt = step
2 Reconstruction with A and B dt = 1 (trasformation of A and B with Harold library)
3 Reconstruction with forced response (work in progress)
'''  

reconstruction = 1
```

Codice 4.5

Ricostruzione dei singoli livelli:

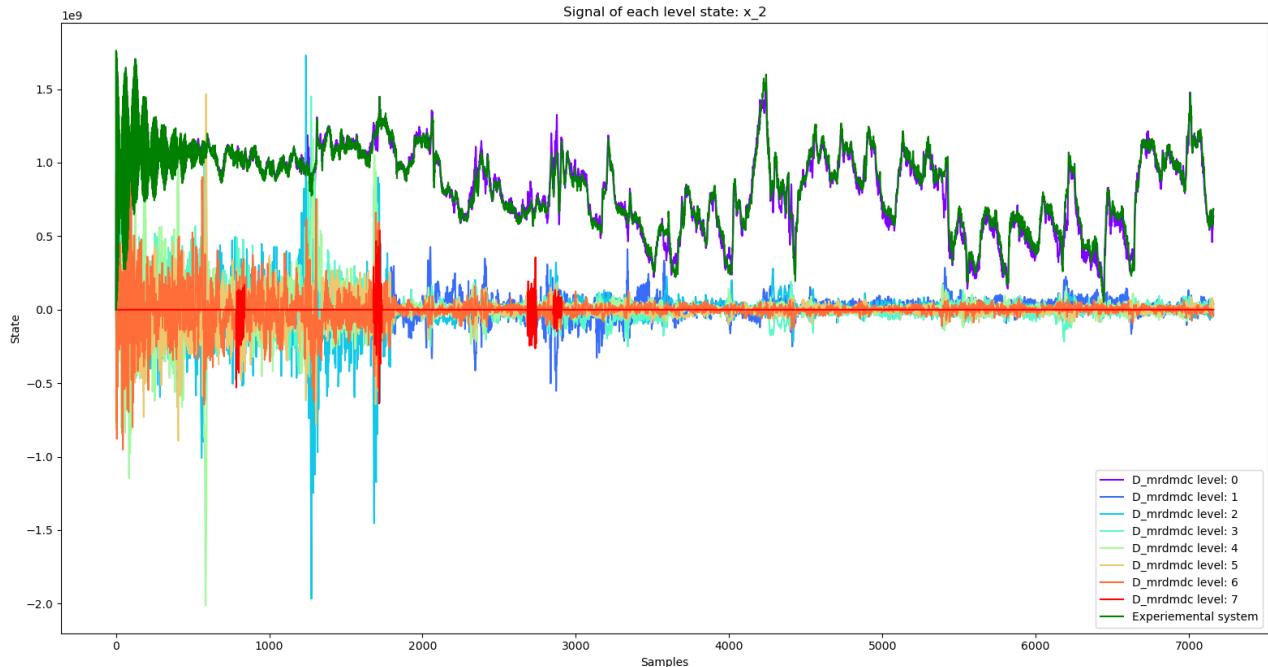


Figura 4.53

Si nota come il livello sette sia quasi del tutto filtrato; quindi, le dinamiche più veloci vengono filtrate a causa della variazione dei parametri

Ricostruzione intero sistema:

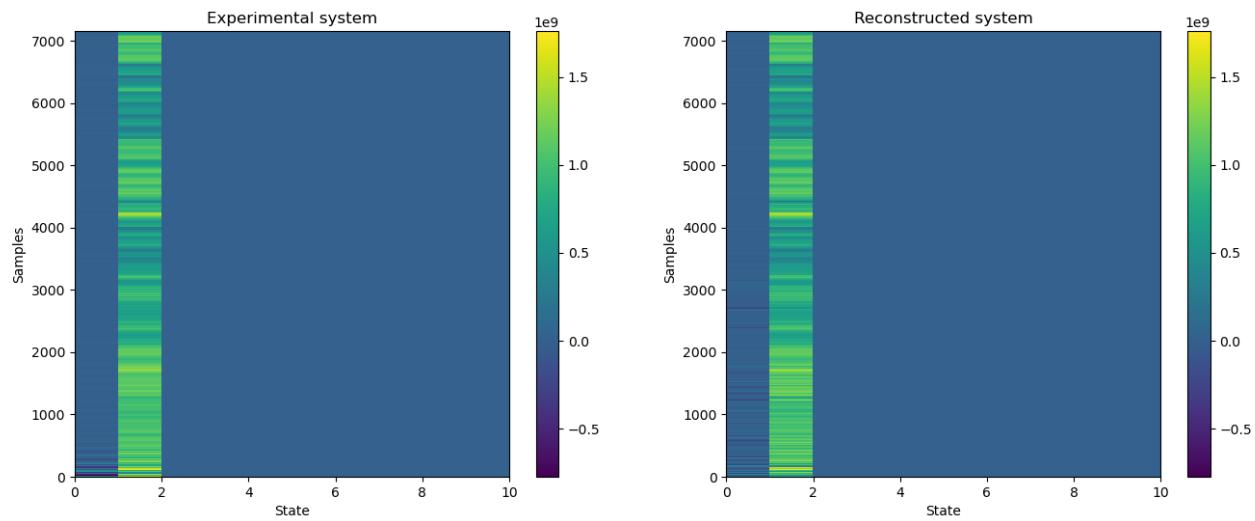
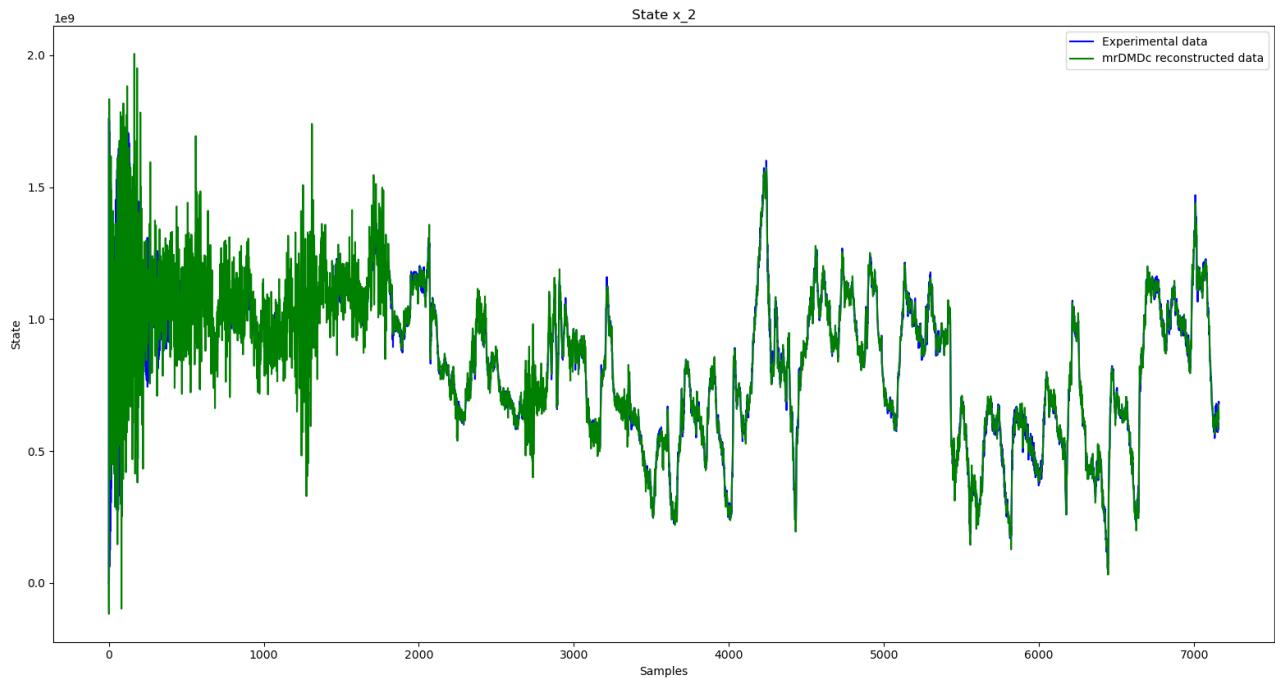
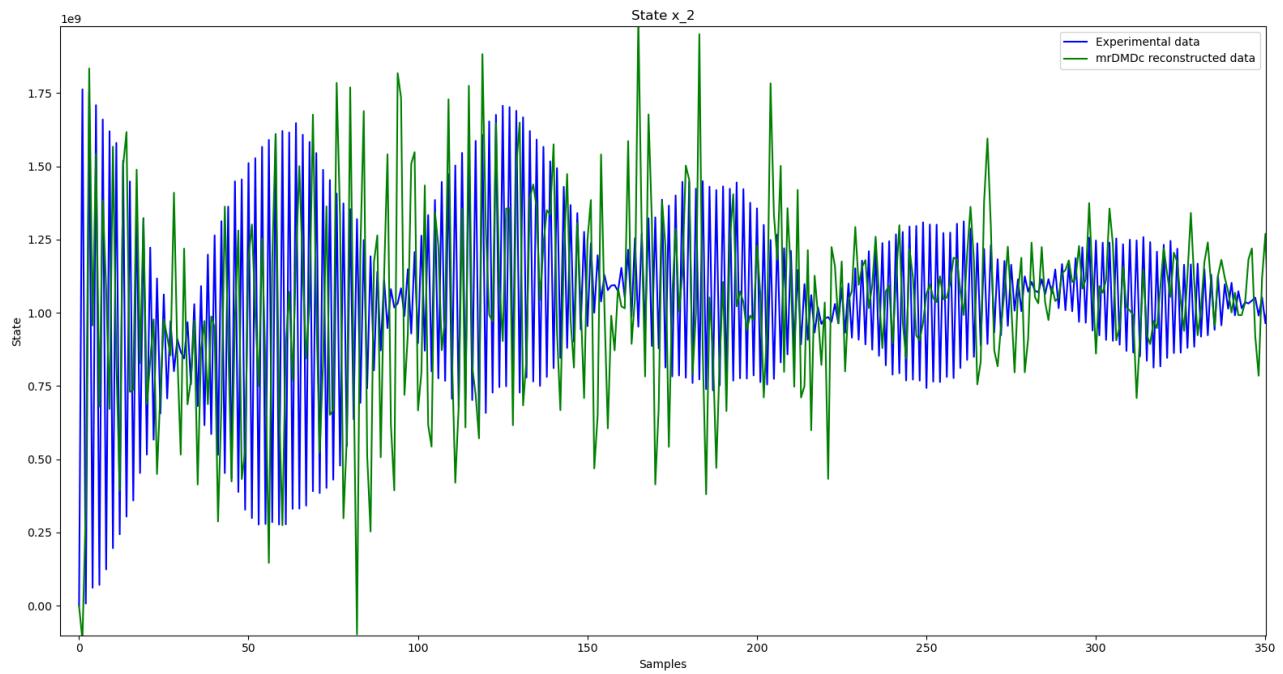


Figura 4.54

Ricostruzione secondo stato:



$1^\circ$  Zoom:



$2^\circ$  Zoom:

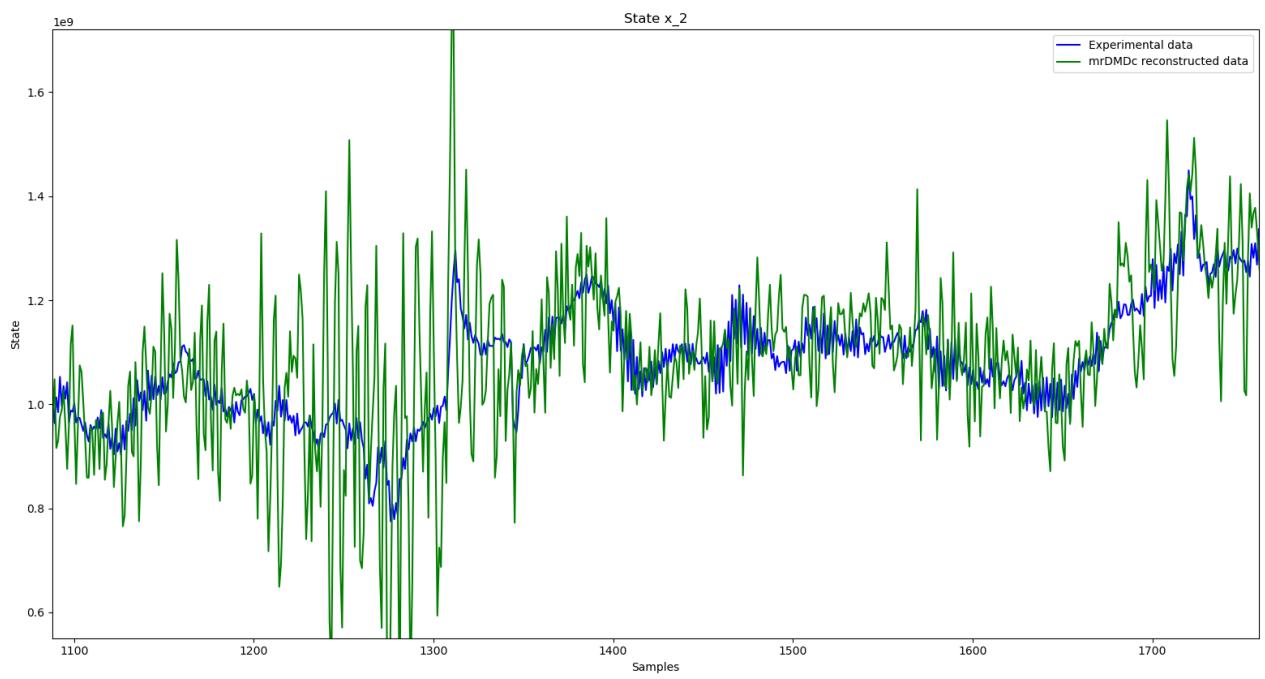


Figura 4.55

Errore:

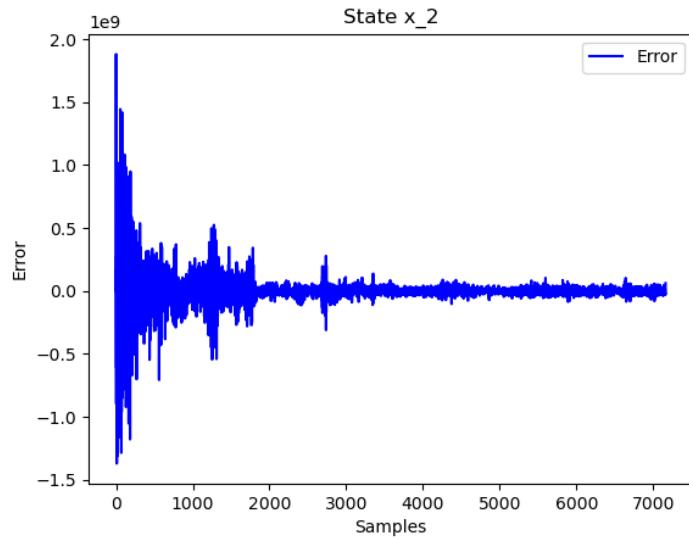


Figura 4.56

Risultati dei KPI calcolati considerando l'intero dataset e non solo una colonna:

MSE	2.40e+15
MAPE	2.77
MAE	9.36e+06
RMSE	4.90e+07
R^2	5.83e-02

Filtrazione per le slow feature in ogni livello:

Filtration level 0	0%
Filtration level 1	0%
Filtration level 2	0%
Filtration level 3	0%
Filtration level 4	1%
Filtration level 5	1%
Filtration level 6	3%
Filtration level 7	98%

#### 4.3.2.3 Sistema sintetico con ingresso n.5 non ritardato dai dati SRU

Sono stati selezionati:

- D (serie temporale di stati) dal dataset generato dal sistema sintetico ad autovalori complessi descritto nel capitolo 3.
- U (serie temporale di ingressi) dal dataset SRU, di cui viene considerato solo il quinto ingresso non ritardato.

##### 4.3.2.3.1 $\text{svd rank} = -1, \text{max cycles} = 18, \text{slow feature scale} = 5, \text{reconstruction} = 1$ :

```
#svd_rank for DMDc object
'''param svd_rank: the rank for the truncation; If 0, the method computes the optimal rank and uses it for truncation;
if positive interger, the method uses the argument for the truncation; if float between 0 and 1, the rank is the number
of the biggest singular values that are needed to reach the 'energy' specified by `svd_rank`; if -1, the method does
not compute truncation.'''
svd_rank_set = -1

#max_cycles for mrDMDc
max_cycles_set = 18
'''

there is a possibility that if the reconstruction is not good (an example if the reconstructed system is unstable),
changing the parameter will work.
No way has been figured out to decide whether to increase or decrease it in the event of a bad reconstruction
'''

#index of filtration, needs to set the size of rho
slow_feature_scale = 5

'''selection of the reconstruction type
1 Reconstruction with A and B dt = step
2 Reconstruction with A and B dt = 1 (trasformation of A and B with Harold library)
3 Reconstruction with forced response (work in progress)
'''
reconstruction = 1
```

Codice 4.6

Rappresentazione del dataset:

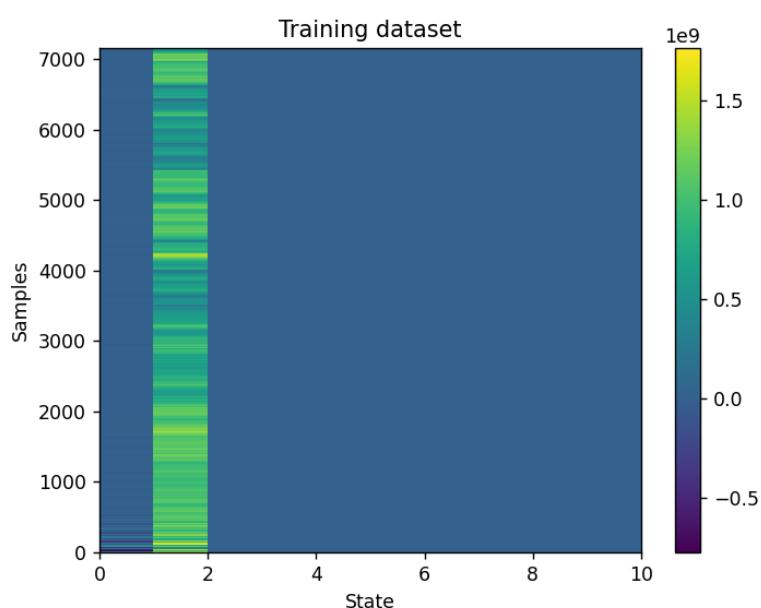


Figura 4.57

Rappresentazione dell'evoluzione del secondo stato nel dataset sperimentale:

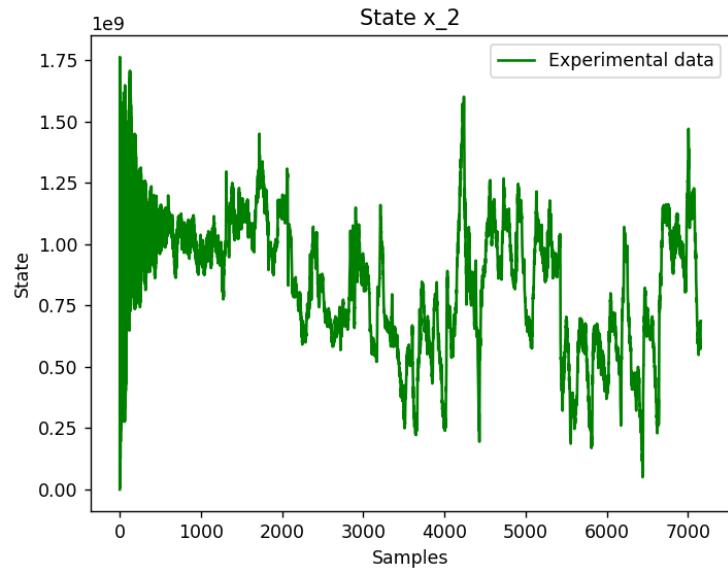
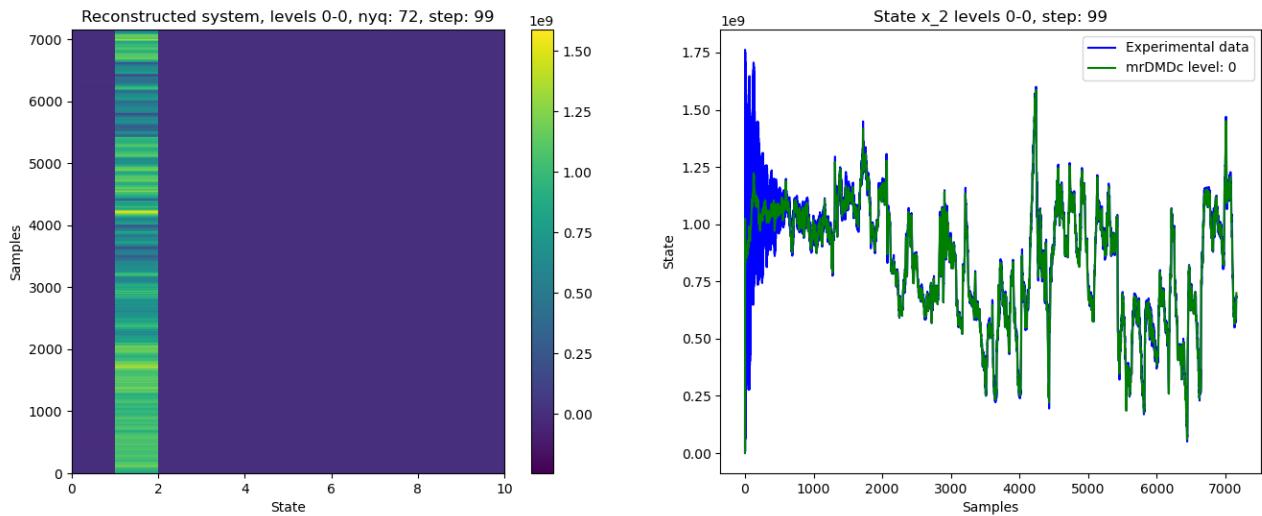
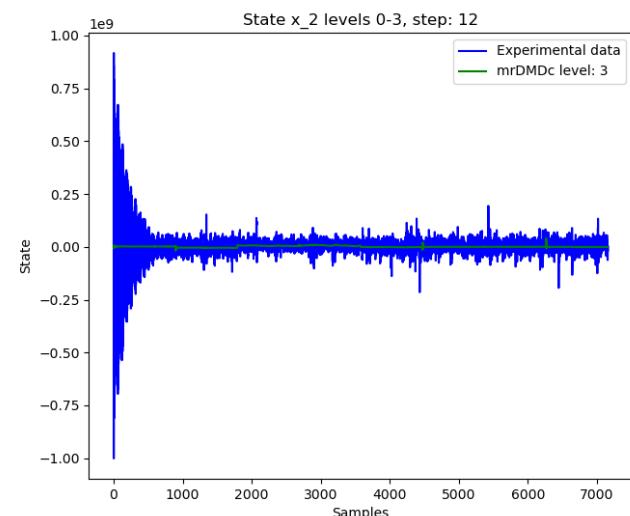
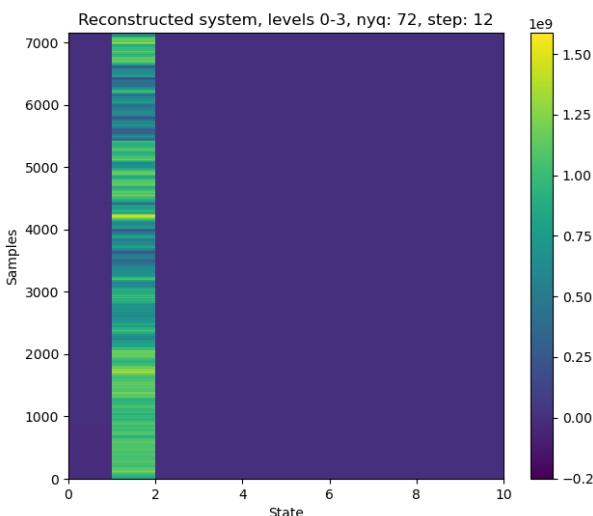
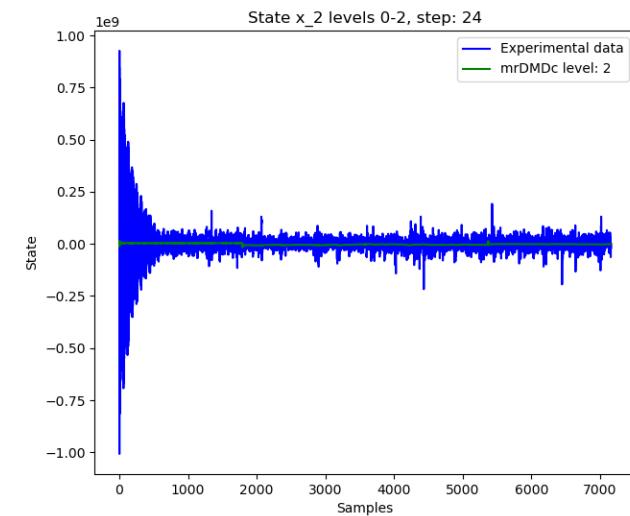
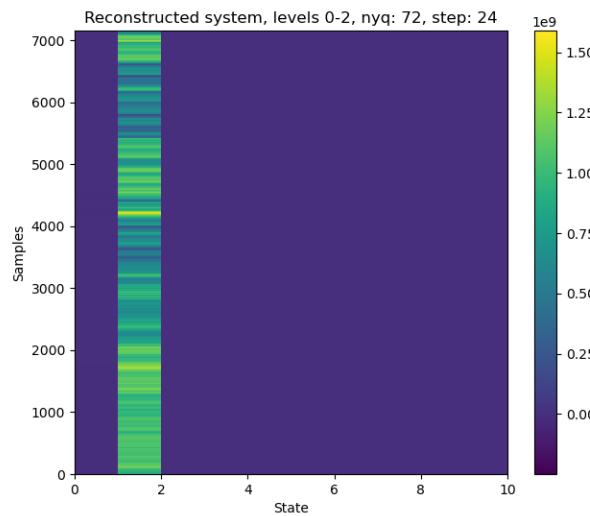
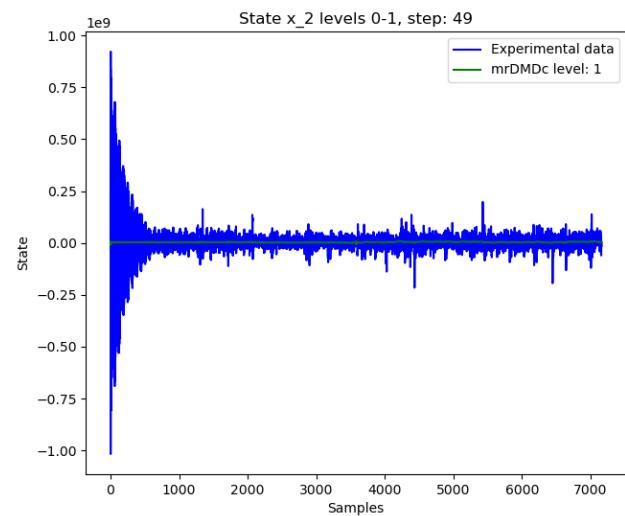
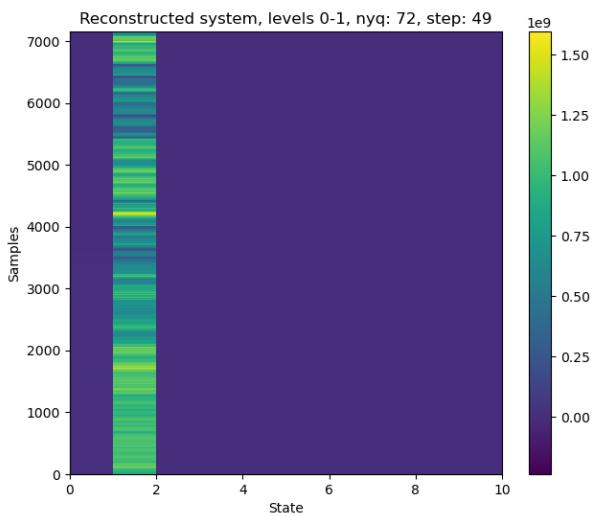


Figura 4.58

Ricostruzione di ogni livello:





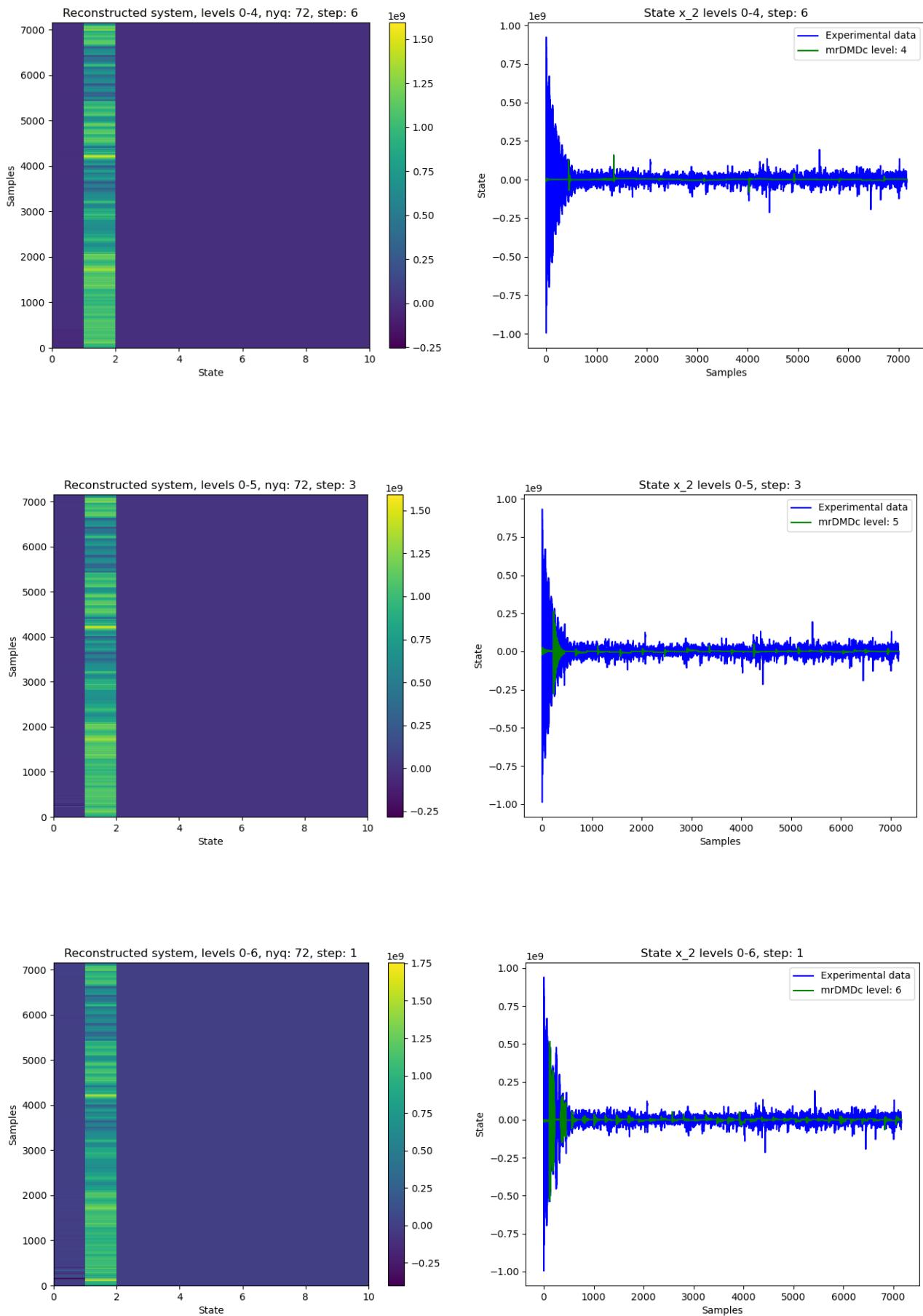
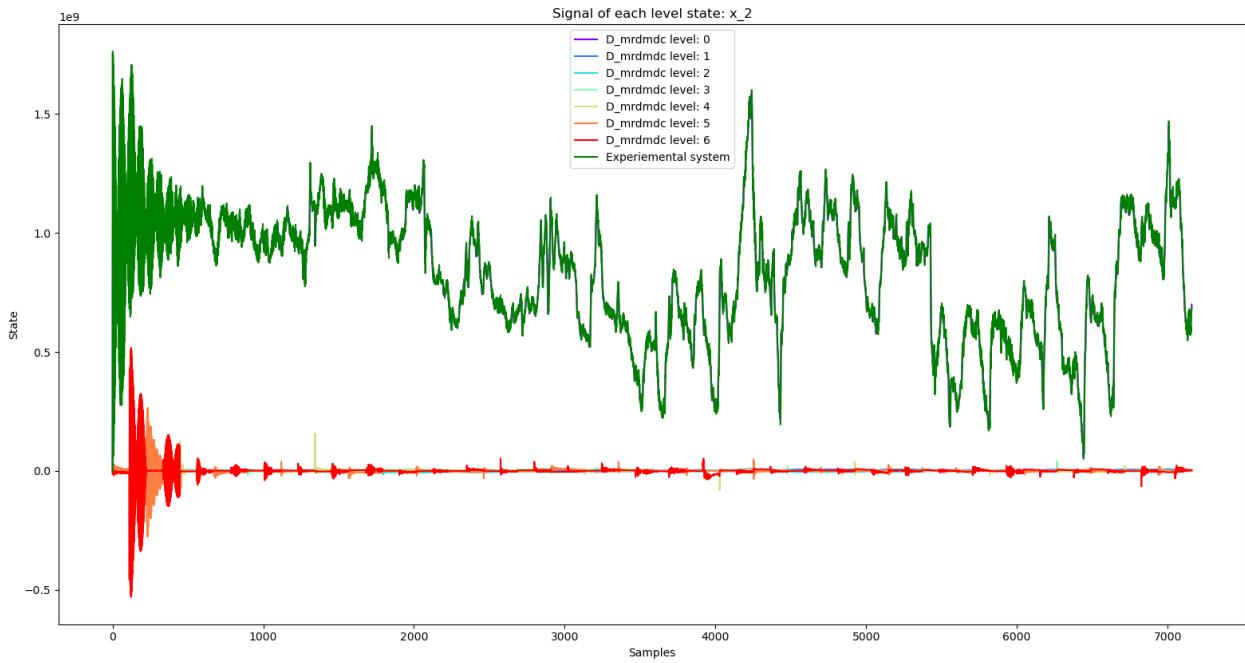


Figura 4.59

Contributi di ogni singolo livello:



Zoom:

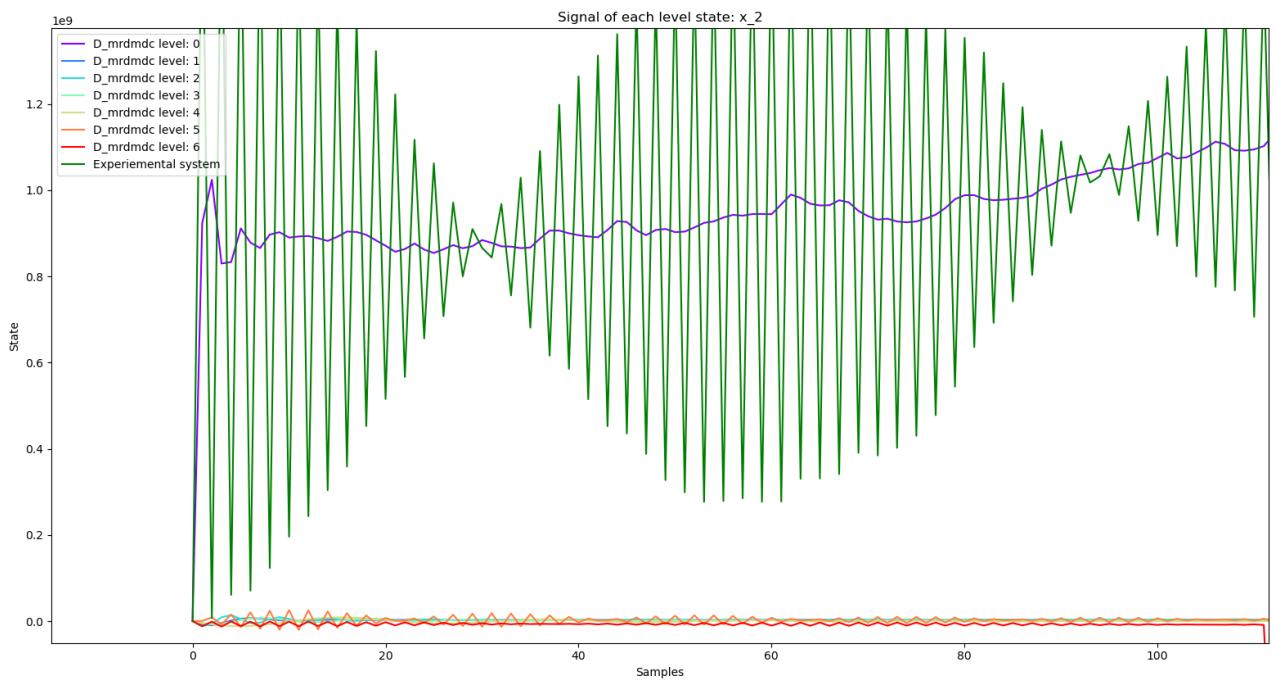


Figura 4.60

Ricostruzione:

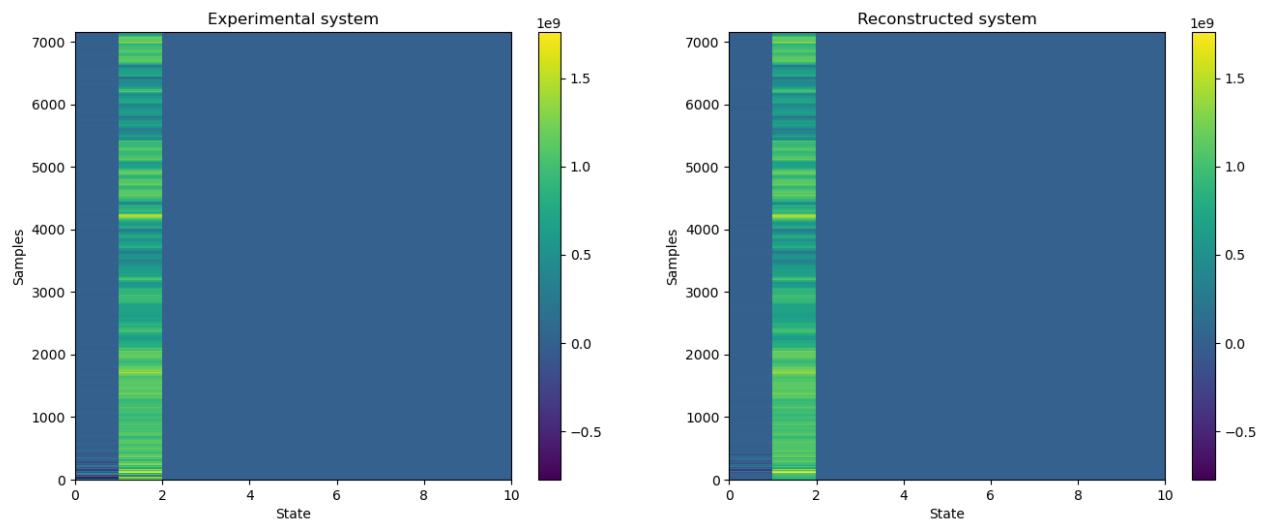
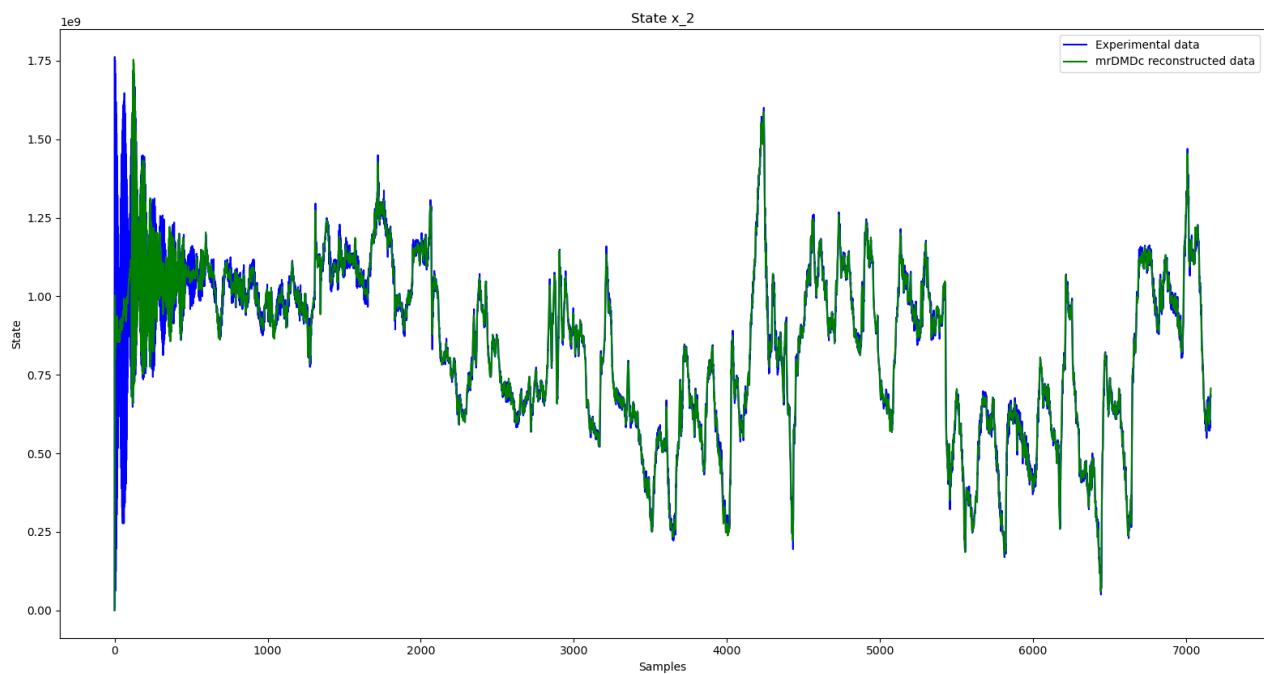
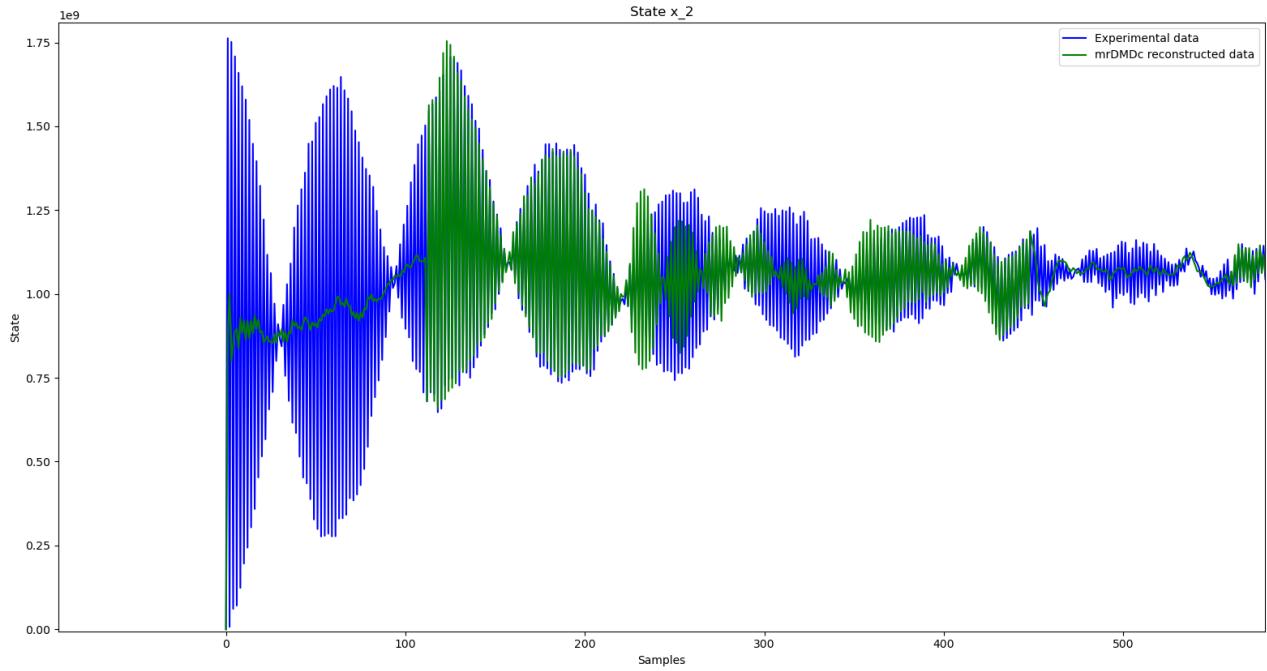


Figura 4.61

Ricostruzione del secondo stato:



1° Zoom:



2° Zoom:

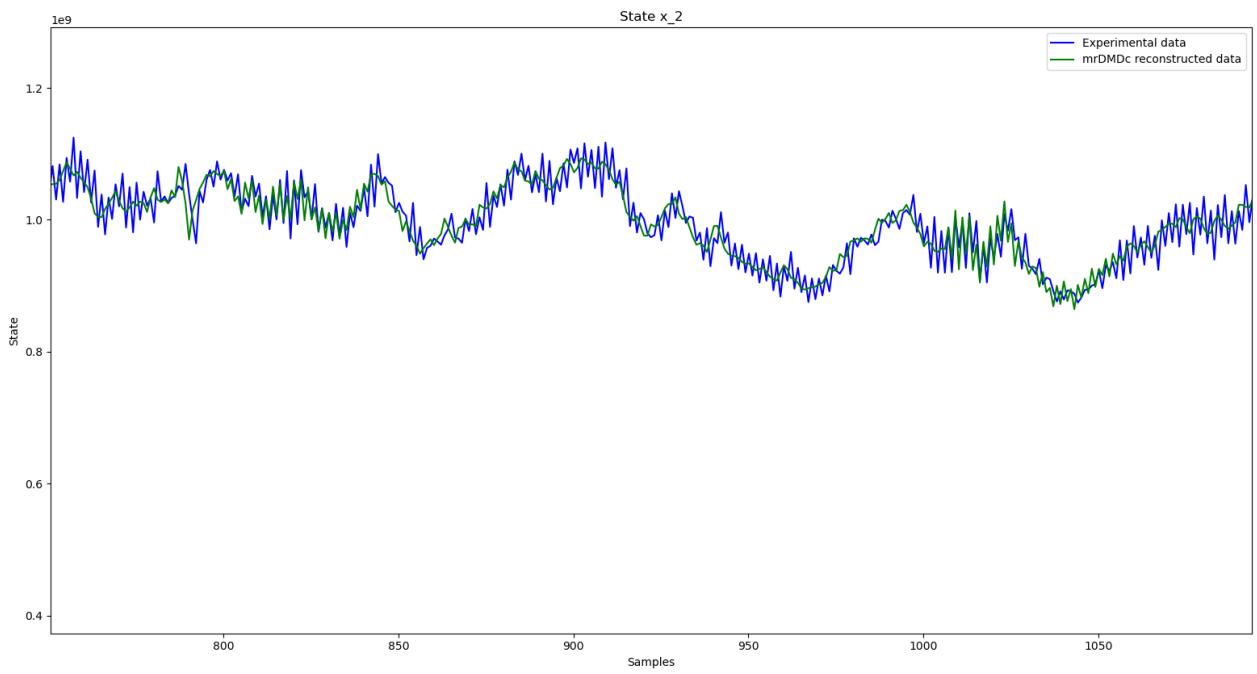


Figura 4.62

La ricostruzione come riportato in figura non è perfetta, questo è probabilmente dovuto alla penuria di dati passati all'algoritmo per testare le sue capacità.

Errore:

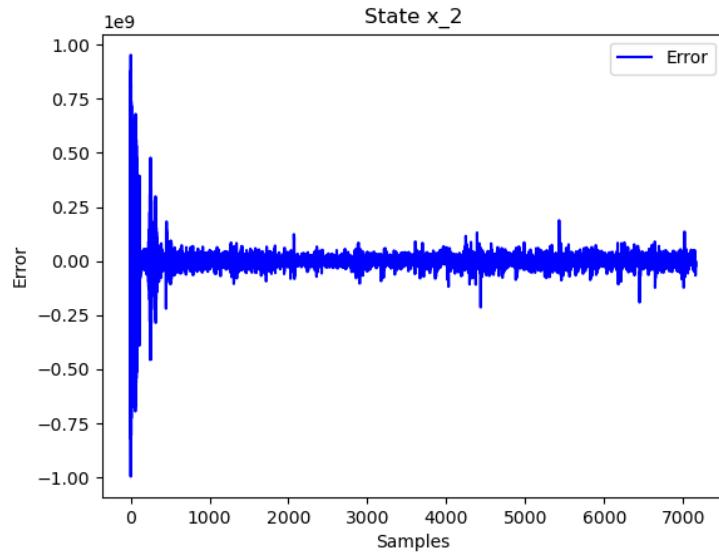


Figura 4.63

Risultati dei KPI calcolati considerando l'intero dataset e non solo una colonna:

MSE	9.92e+14
MAPE	2.67e+01
MAE	4.78e+06
RMSE	3.15e+07
R <sup>2</sup>	-10.76

Filtrazione per le slow feature in ogni livello:

Filtration level 0	0%
Filtration level 1	0%
Filtration level 2	0%
Filtration level 3	0%
Filtration level 4	0%
Filtration level 5	0%
Filtration level 6	0%

#### 4.3.2.4 V2G

##### 4.3.2.4.1 Stato non ritardato ed ingressi non ritardati

4.3.2.4.1.1  $svd\ rank = -1$ ,  $max\ cycles = 60$ ,  $slow\ feature\ scale = 5$ ,  $reconstruction = 1$ :

```
#svd_rank for DMDc object
'''param svd_rank: the rank for the truncation; If 0, the method computes the optimal rank and uses it for truncation;
| if positive interger, the method uses the argument for the truncation; if float between 0 and 1, the rank is the number
| of the biggest singular values that are needed to reach the 'energy' specified by `svd_rank`; if -1, the method does
| not compute truncation.'''
svd_rank_set = -1

#max_cycles for mrDMDc
max_cycles_set = 60
...
there is a possibility that if the reconstruction is not good (an example if the reconstructed system is unstable),
changing the parameter will work.
No way has been figured out to decide whether to increase or decrease it in the event of a bad reconstruction
| ...

#index of filtration, needs to set the size of rho
slow_feature_scale = 5

'''selection of the reconstruction type
1 Reconstruction with A and B dt = step
2 Reconstruction with A and B dt = 1 (trasformation of A and B with Harold library)
3 Reconstruction with forced response (work in progress)
...
reconstruction = 1
```

Codice 4.7

Rappresentazione dell'evoluzione dello stato del dataset sperimentale:

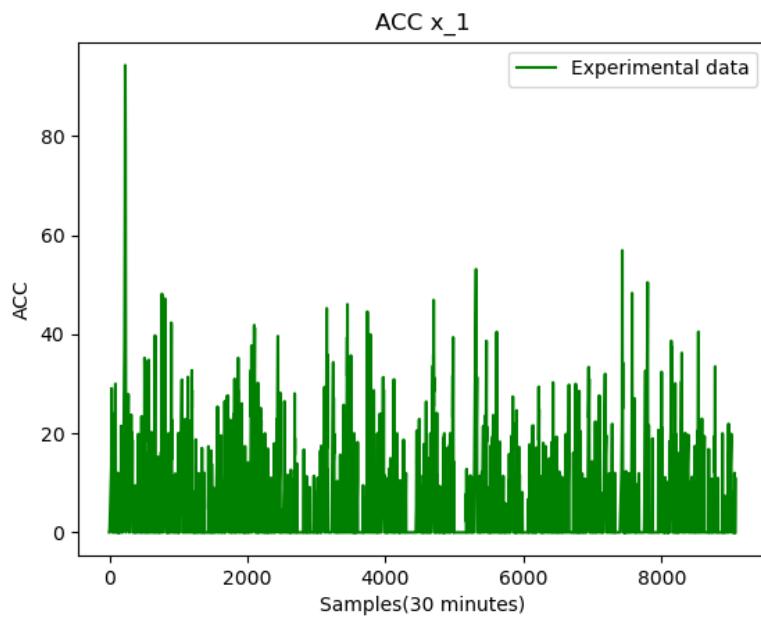
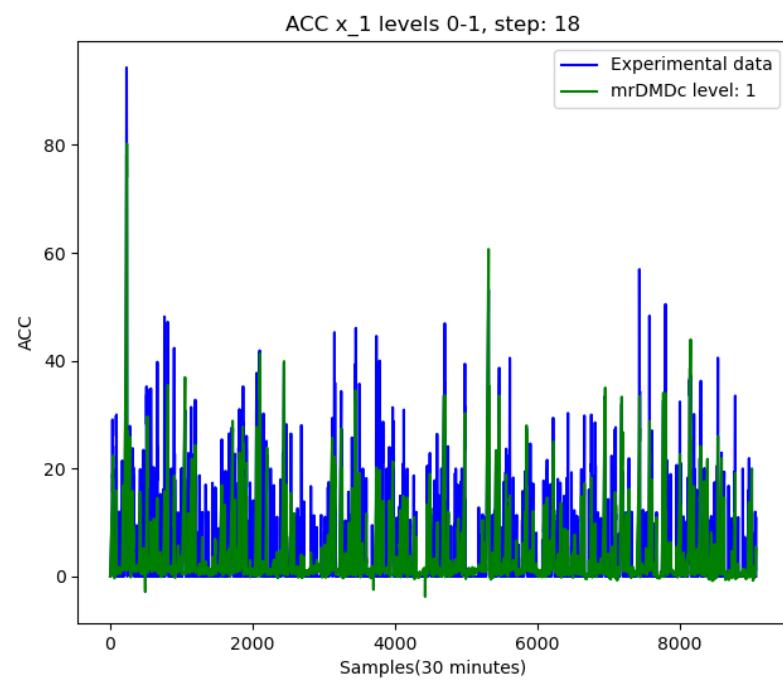
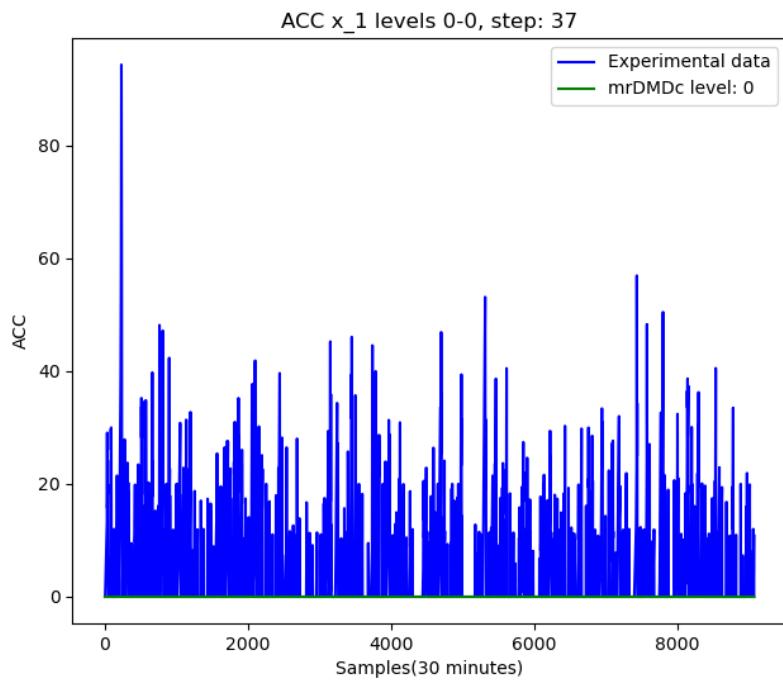
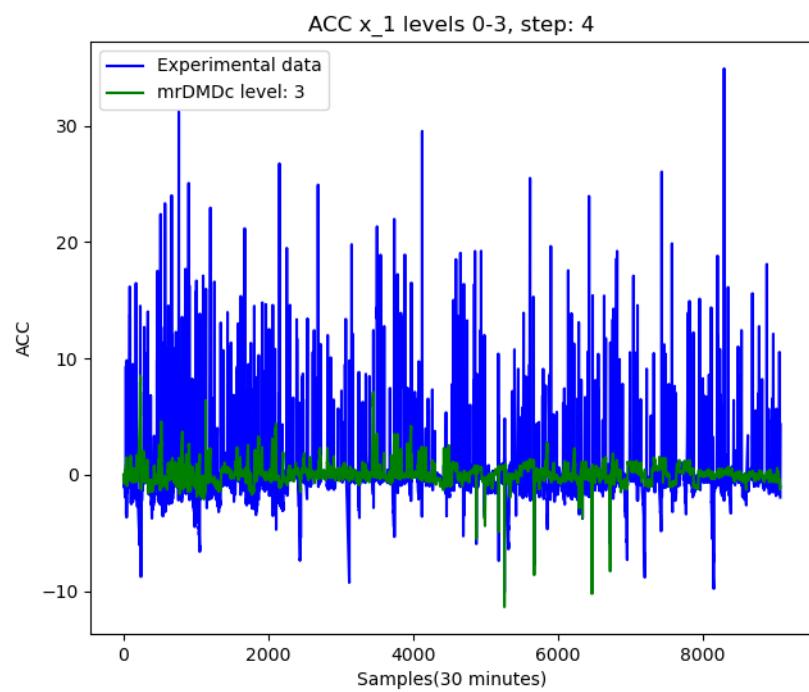
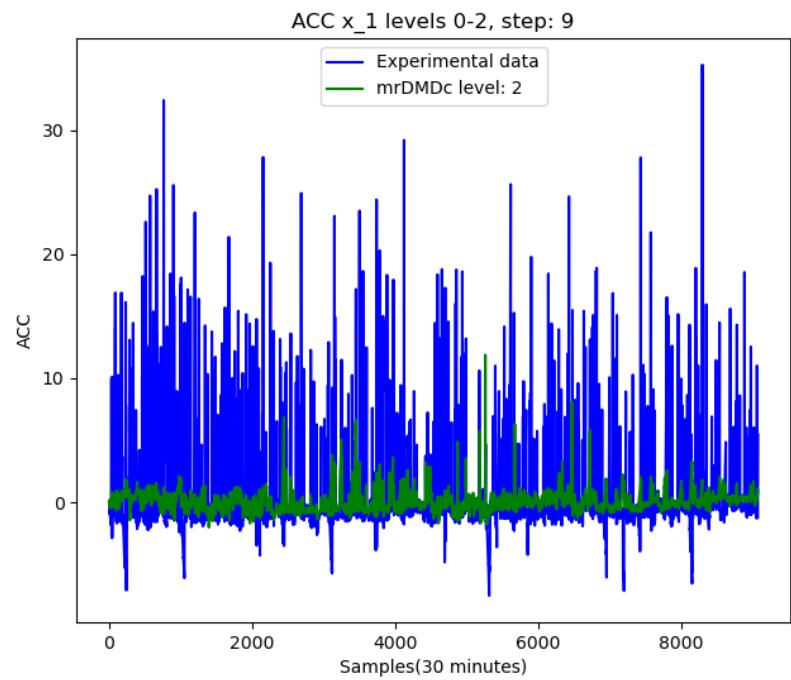


Figura 4.64

Ricostruzione di ogni livello:





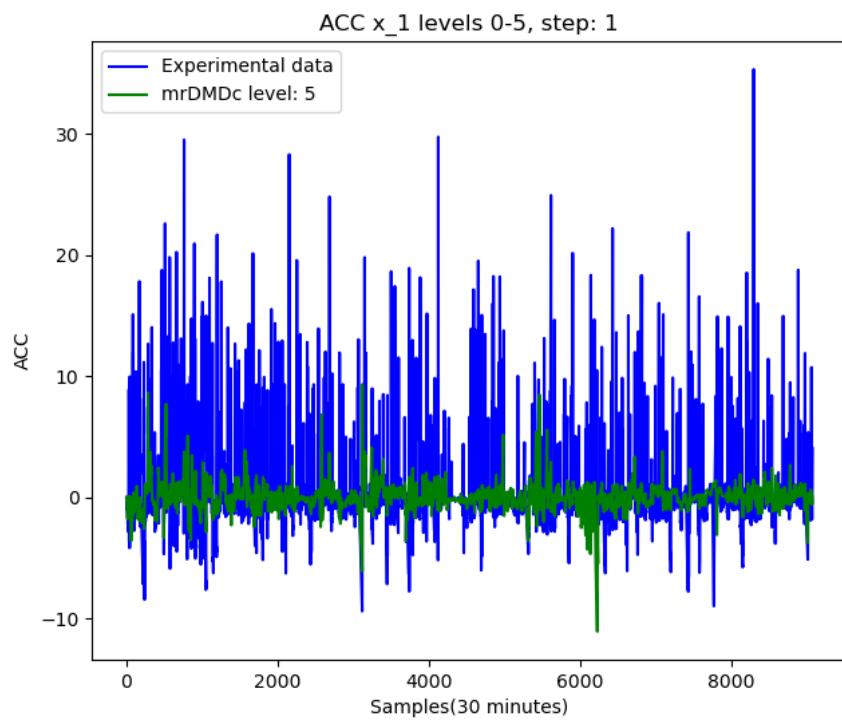
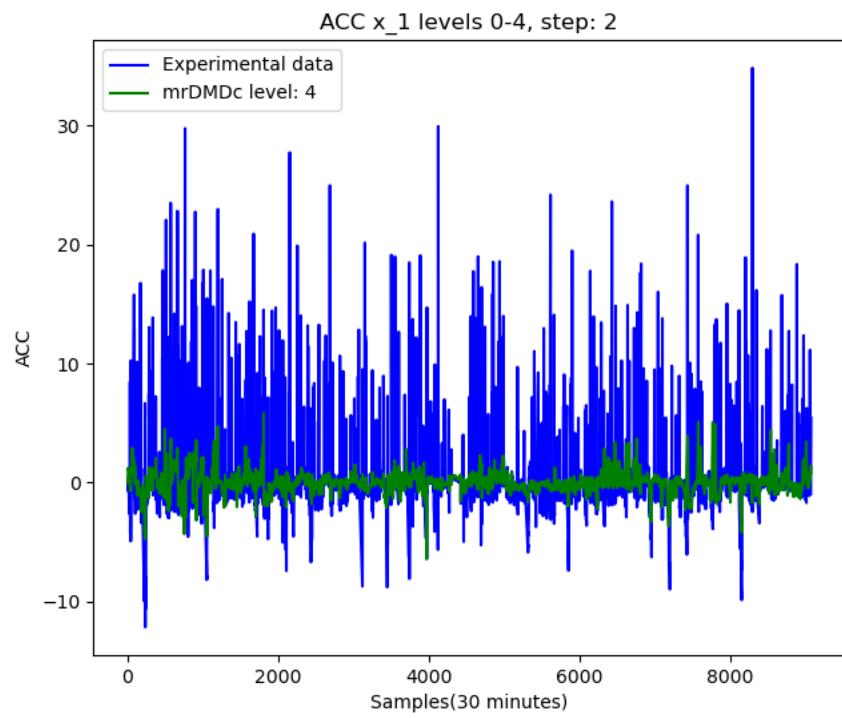
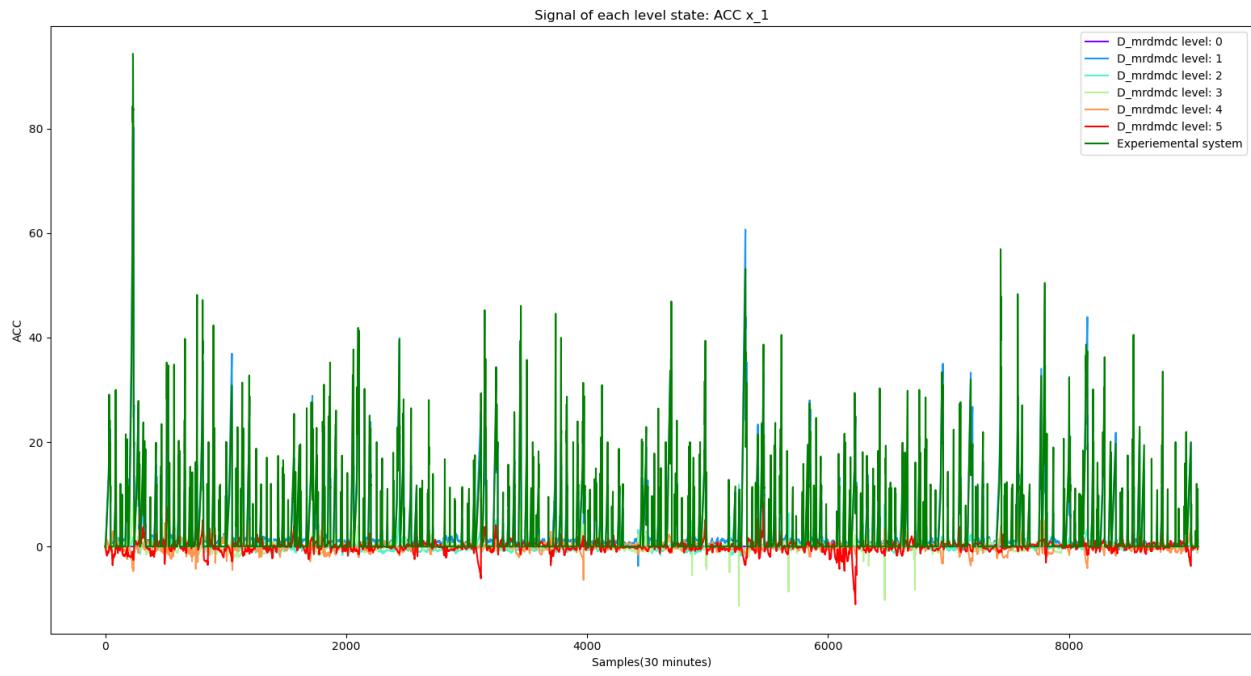


Figura 4.65

Contributi di ogni singolo livello:



Zoom:

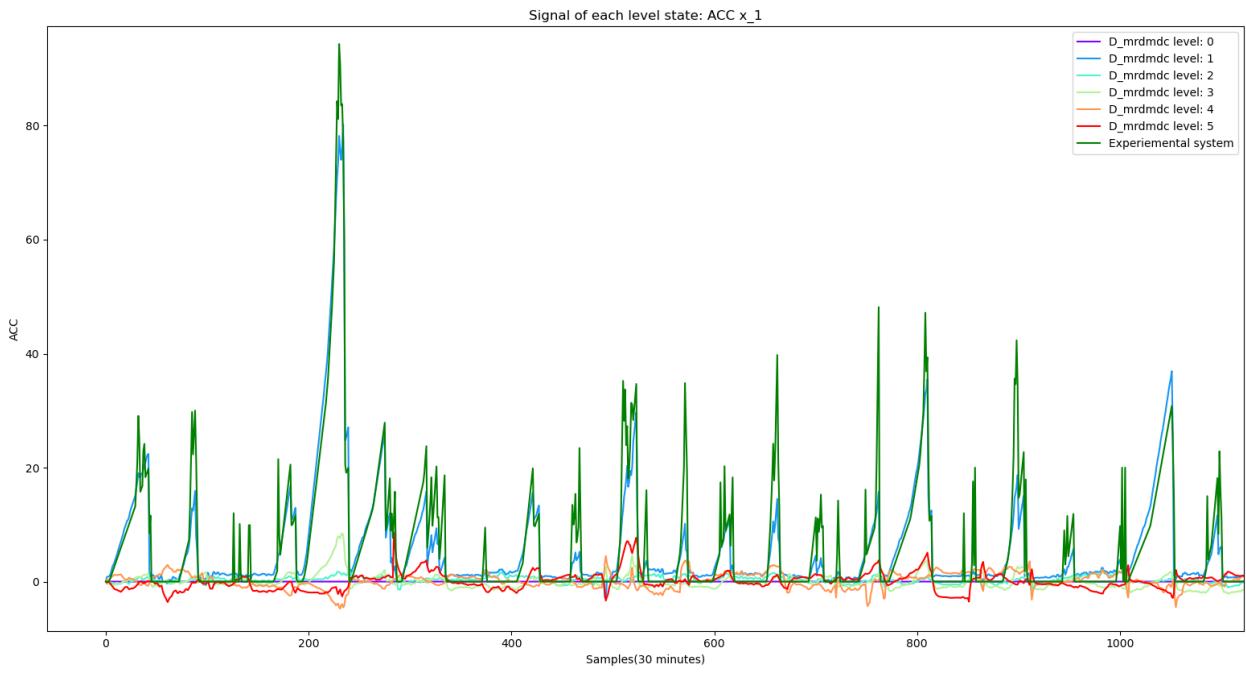
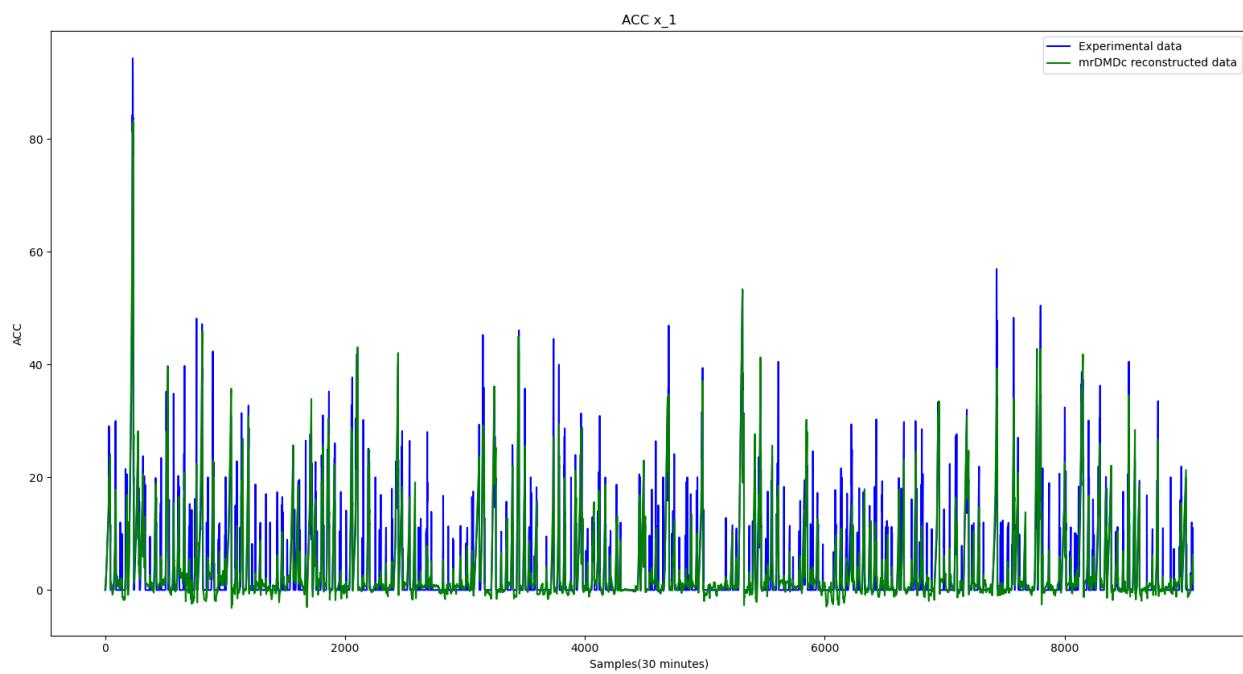


Figura 4.66

Ricostruzione:



Zoom:

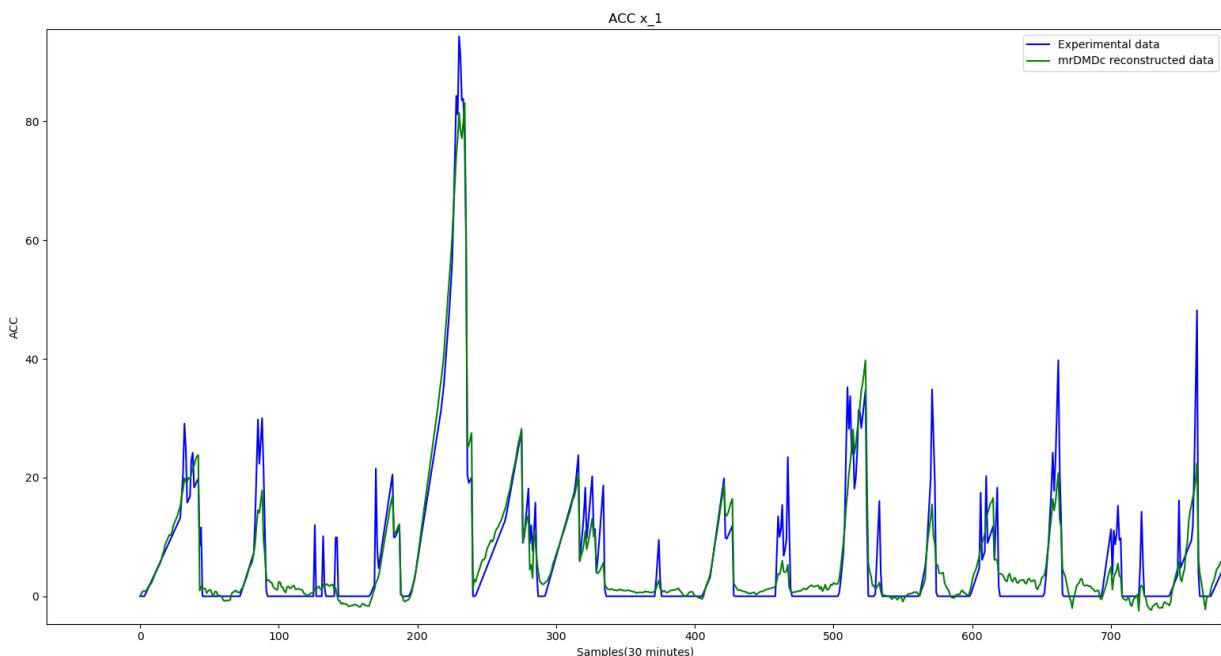


Figura 4.67

Errore:

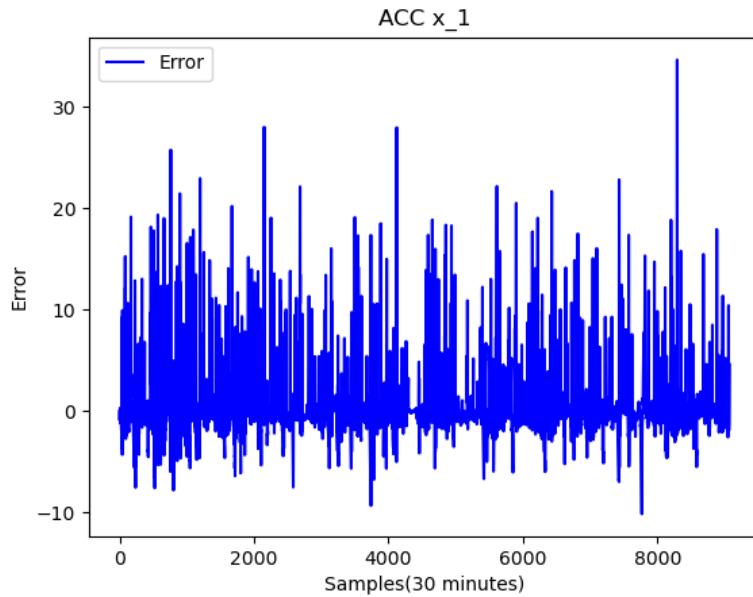


Figura 4.68

Risultati dei KPI calcolati considerando l'intero dataset e non solo una colonna:

MSE	8.40
MAPE	1.61
MAE	1.60
RMSE	2.90
R^2	0.86

Filtrazione per le slow feature in ogni livello:

Filtration level 0	Non trova dinamiche
Filtration level 1	0%
Filtration level 2	0%
Filtration level 3	0%
Filtration level 4	0%
Filtration level 5	0%
Filtration level 6	0%

#### 4.3.2.4.2 Stato ritardato ed ingressi non ritardati

Sono stati selezionati la D (serie temporale di stati) e la U (serie temporale di ingressi) dal dataset V2G.

4.3.2.4.2.1  $svd\ rank = -1$ ,  $max\ cycles = 30$ ,  $slow\ feature\ scale = 5$ ,  $reconstruction = 1$ :

```
#svd_rank for DMDc object
'''param svd_rank: the rank for the truncation; If 0, the method computes the optimal rank and uses it for truncation;
if positive interger, the method uses the argument for the truncation; if float between 0 and 1, the rank is the number
of the biggest singular values that are needed to reach the 'energy' specified by `svd_rank`; if -1, the method does
not compute truncation.'''  
svd_rank_set = -1

#max_cycles for mrDMDc
max_cycles_set = 30
'''  
there is a possibility that if the reconstruction is not good (an example if the reconstructed system is unstable),
changing the parameter will work.  
No way has been figured out to decide whether to increase or decrease it in the event of a bad reconstruction  
#index of filtration, needs to set the size of rho
slow_feature_scale = 5

'''selection of the reconstruction type
1 Reconstruction with A and B dt = step
2 Reconstruction with A and B dt = 1 (trasformation of A and B with Harold library)
3 Reconstruction with forced response (work in progress)
'''  
reconstruction = 1
```

Codice 4.8

Rappresentazione dell’evoluzione dello stato del dataset sperimentale:

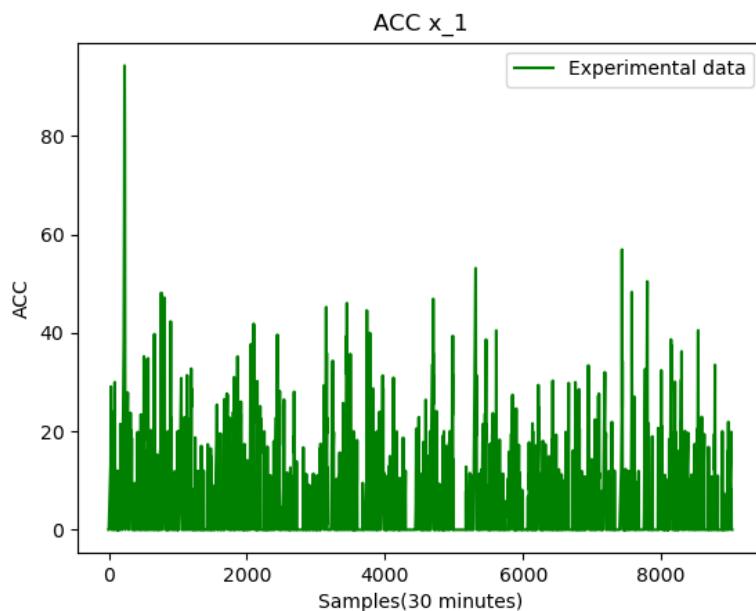
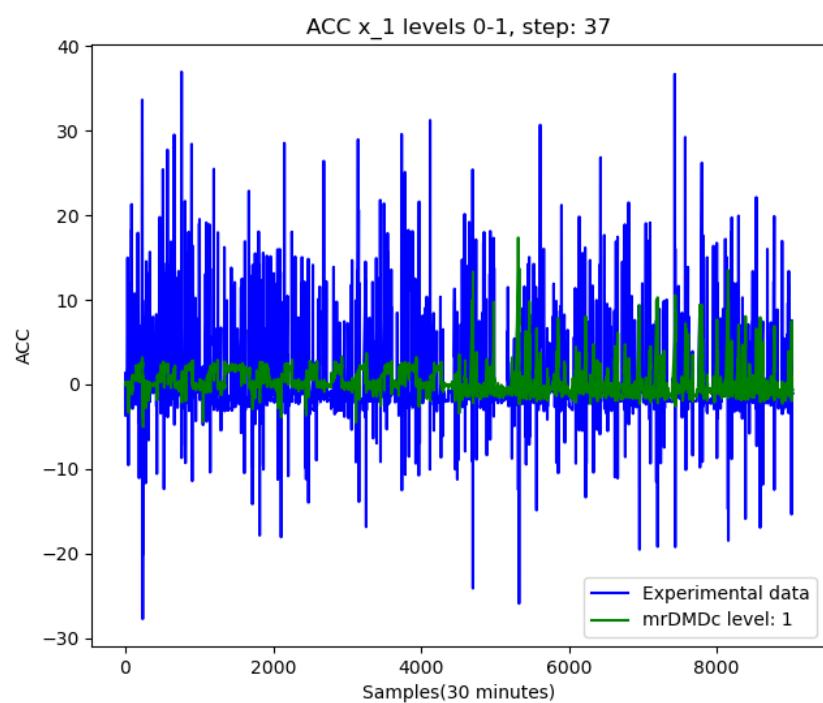
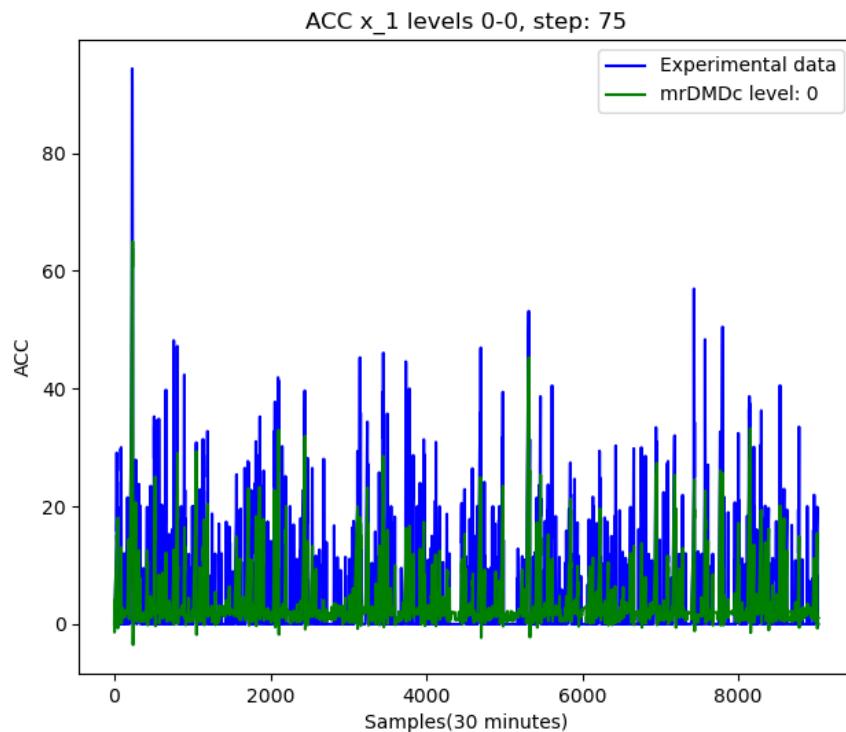
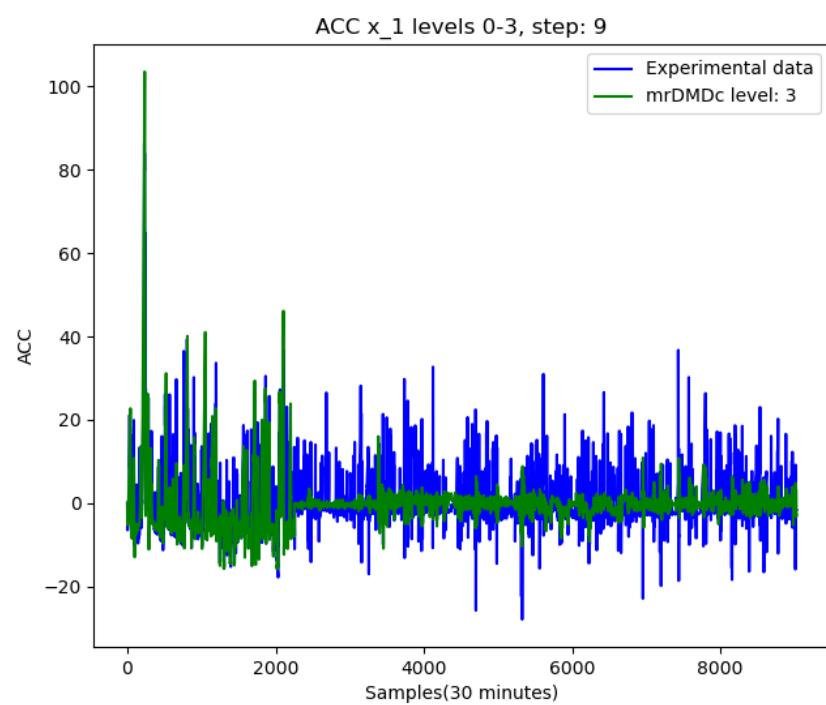
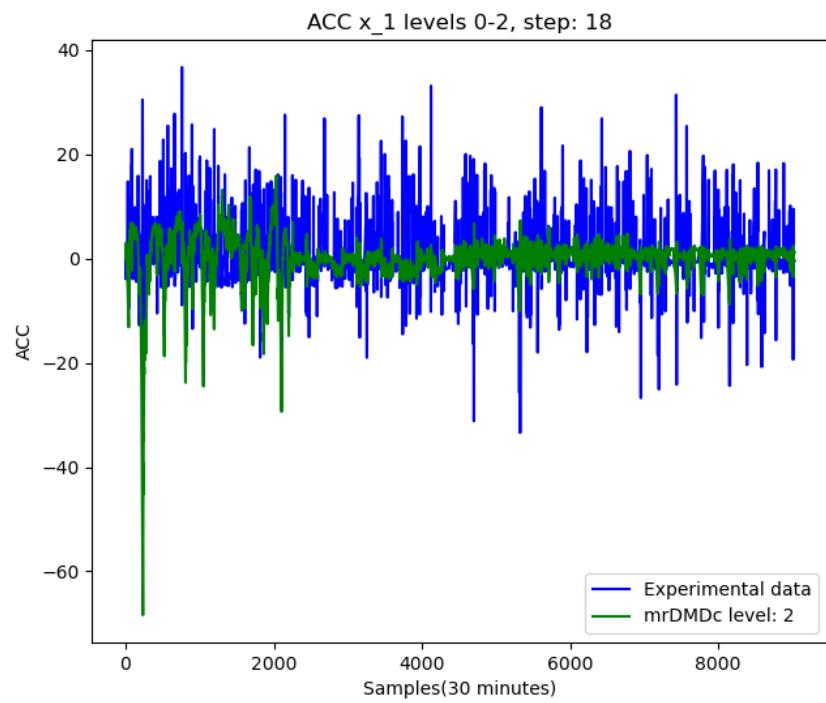
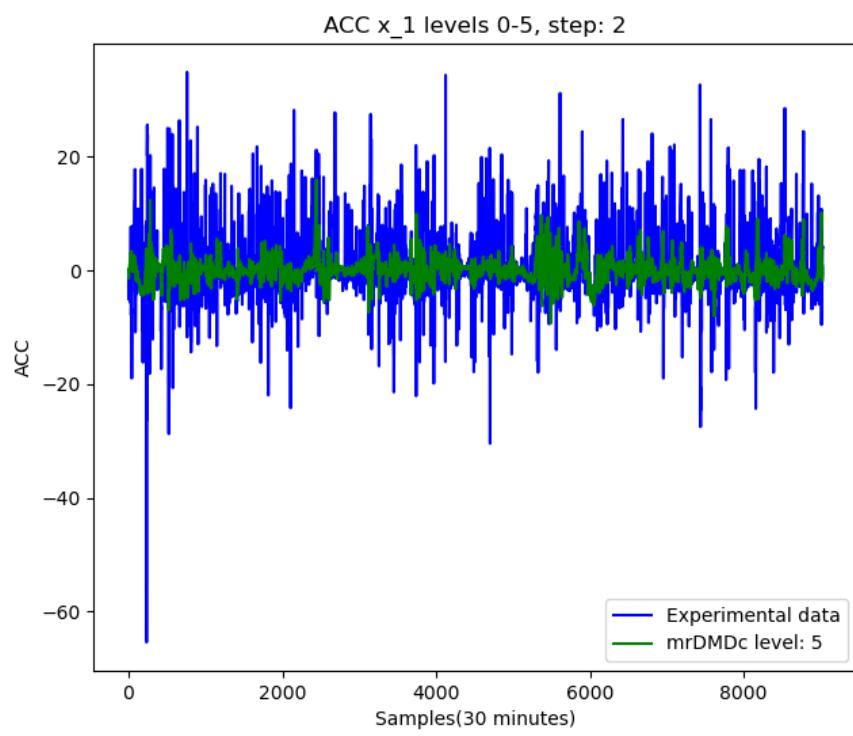
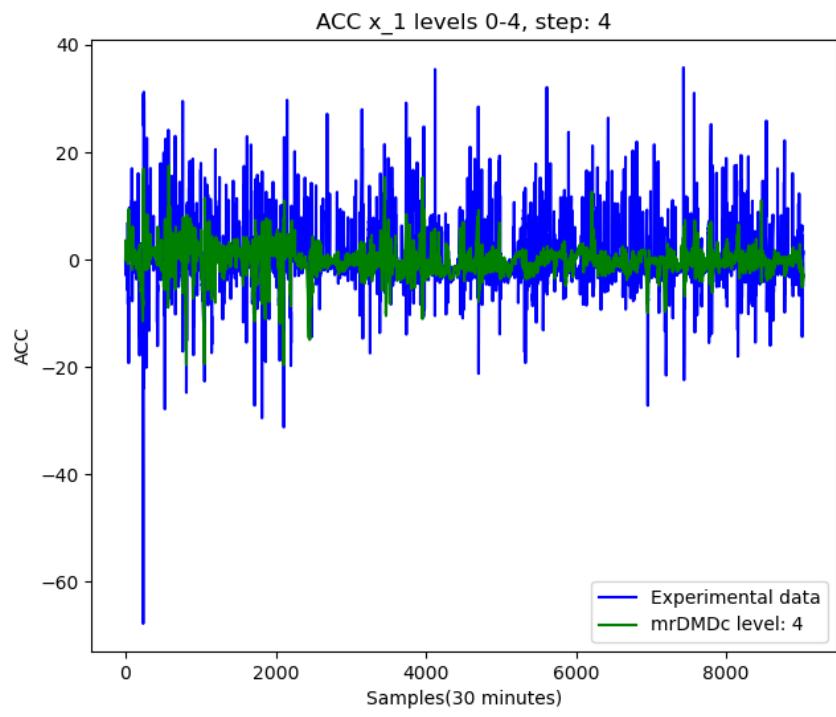


Figura 4.69

Ricostruzione di ogni livello:







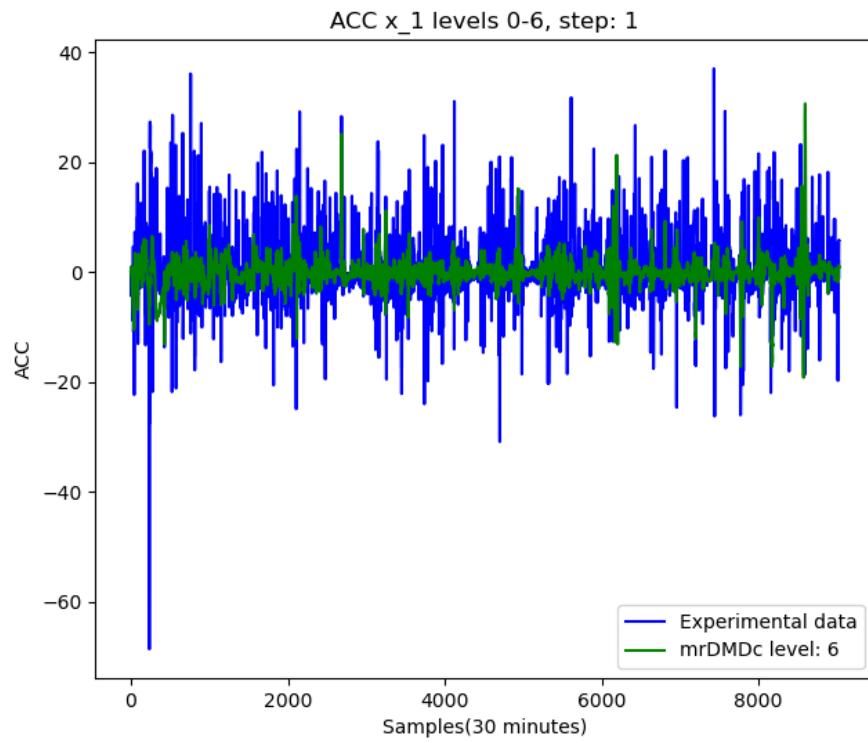
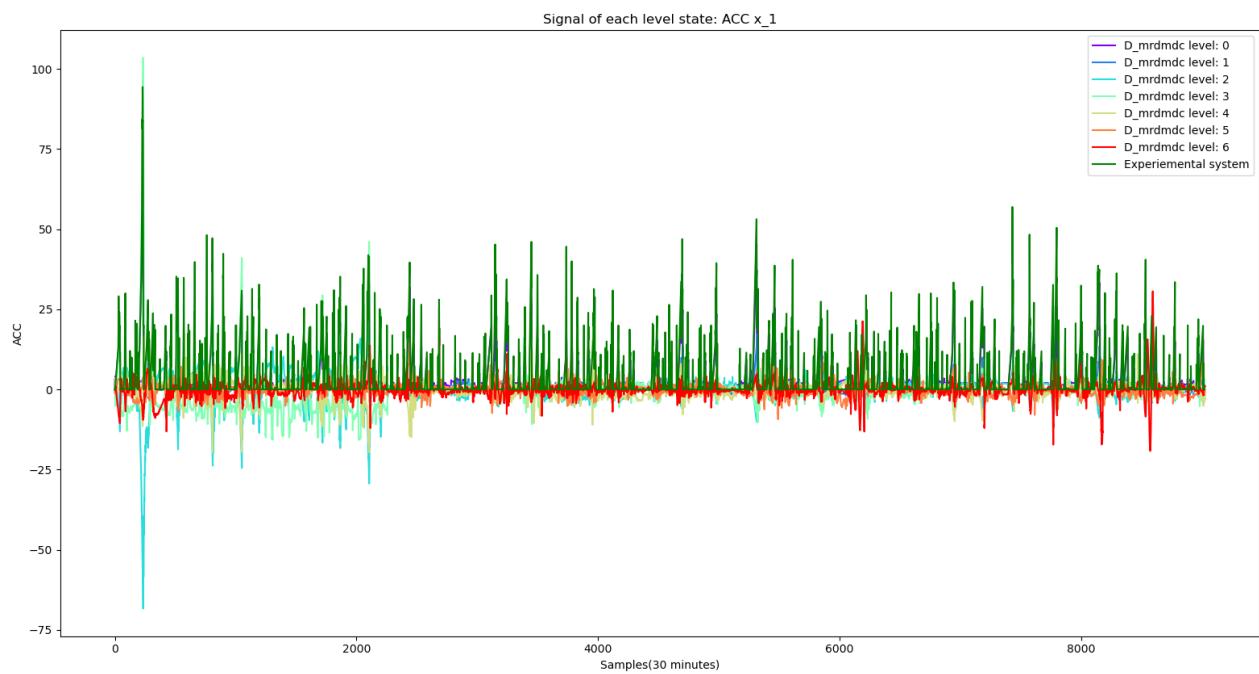


Figura 4.70

Contributi di ogni singolo livello:



Zoom:

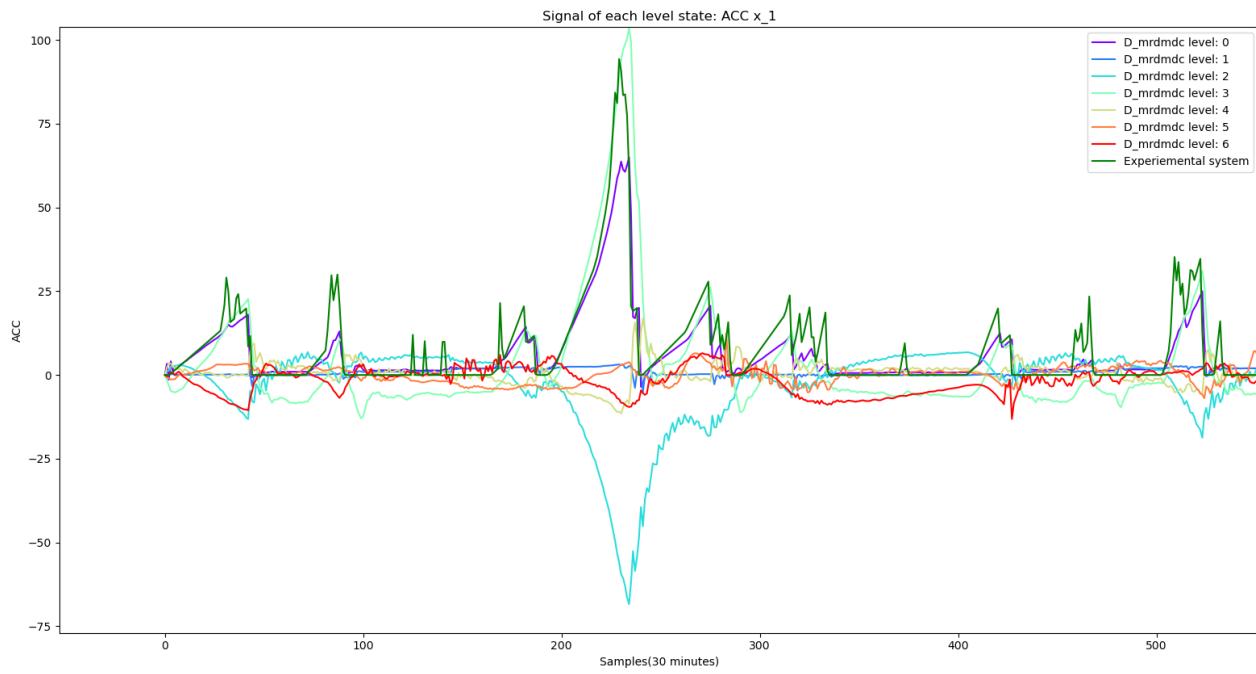
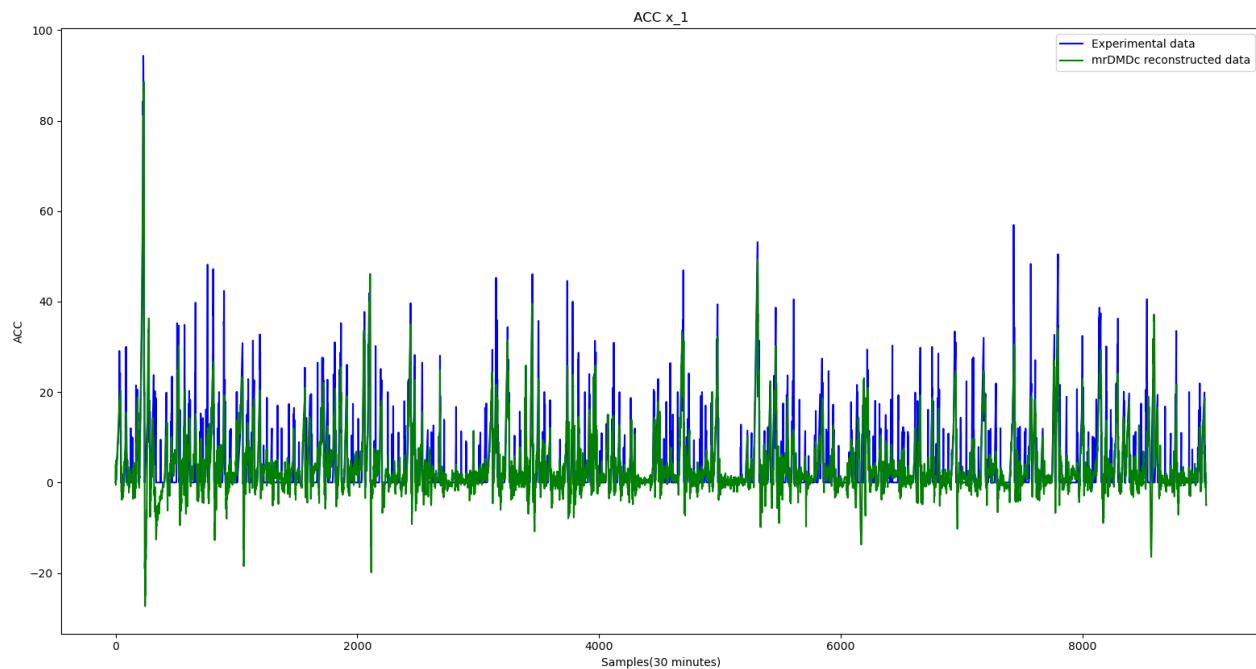


Figura 4.71

Ricostruzione:



Zoom:

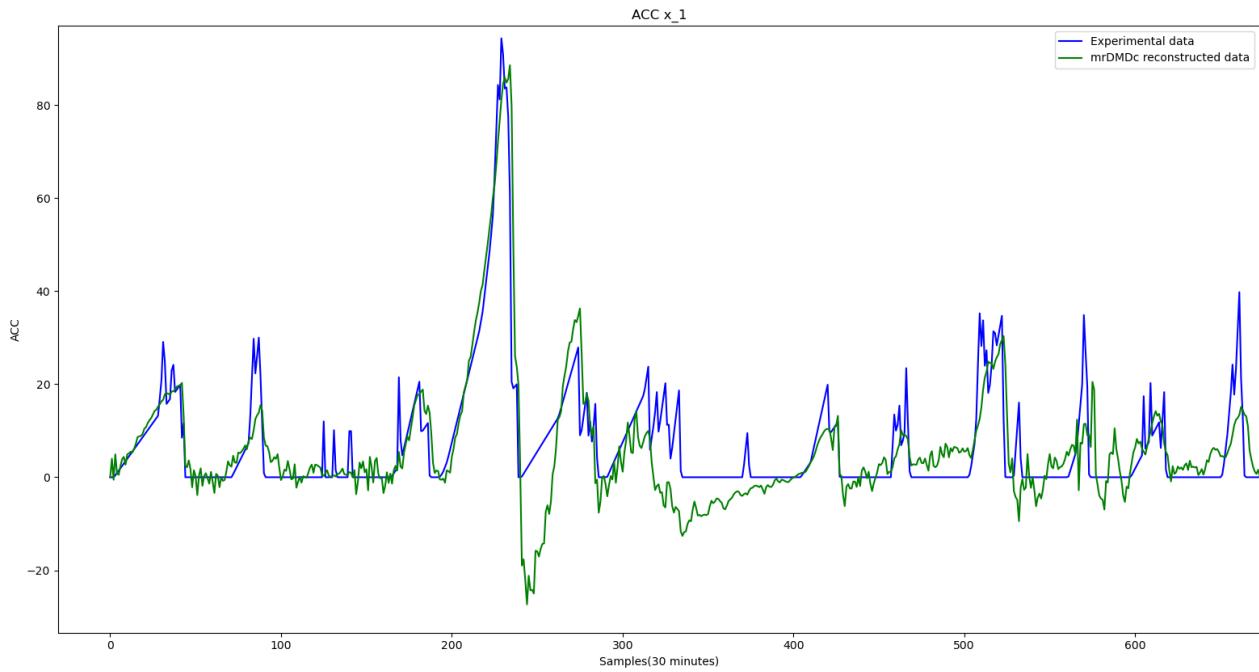


Figura 4.72

Errore:

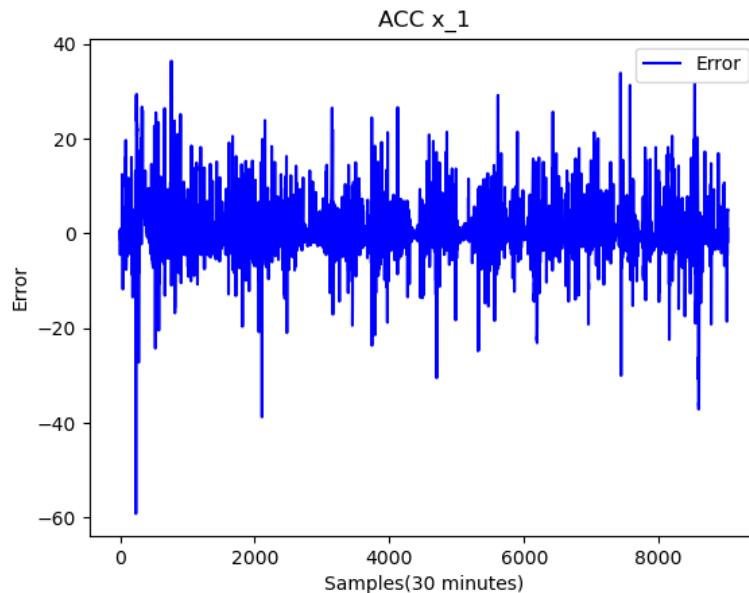


Figura 4.73

Risultati dei KPI calcolati considerando l'intero dataset e non solo una colonna:

MSE	1.06e+02
MAPE	3.34
MAE	6.22
RMSE	1.03e+01
R^2	-0.26

Filtrazione per le slow feature in ogni livello:

Filtration level 0	0%
Filtration level 1	0%
Filtration level 2	0%
Filtration level 3	0%
Filtration level 4	0%
Filtration level 5	0%
Filtration level 6	0%

#### 4.3.2.4.3 Stato ritardato ed ingressi ritardati

Come per il DMDc sono stati eseguiti più esperimenti nel caso dello stato ritardato e degli ingressi ritardati. Vengono mostrati i parametri impostati, i KPI, il plot con tutte le ricostruzioni dello stato, ed in più vengono riportate le filtrazioni per ogni livello. Di seguito vediamo i parametri impostati:

##### 4.3.2.4.3.1 $svd\ rank = -1$ , $max\ cycles = 10$ , $slow\ feature\ scale = 5$ , $reconstruction = 1$ :

```
#svd_rank for DMDc object
'''param svd_rank: the rank for the truncation; If 0, the method computes the optimal rank and uses it for truncation;
| if positive interger, the method uses the argument for the truncation; if float between 0 and 1, the rank is the number
| of the biggest singular values that are needed to reach the 'energy' specified by `svd_rank`; if -1, the method does
| not compute truncation.''''
svd_rank_set = -1

#max_cycles for mrDMDc
max_cycles_set = 10
...
there is a possibility that if the reconstruction is not good (an example if the reconstructed system is unstable),
changing the parameter will work.
No way has been figured out to decide whether to increase or decrease it in the event of a bad reconstruction
| ...

#index of filtration, needs to set the size of rho
slow_feature_scale = 5

'''selection of the reconstruction type
1 Reconstruction with A and B dt = step
2 Reconstruction with A and B dt = 1 (trasformation of A and B with Harold library)
3 Reconstruction with forced response (work in progress)
...
reconstruction = 1
```

Codice 4.9

- *meteo + aggregated*:

Risultati dei KPI calcolati considerando l'intero dataset e non solo una colonna:

MSE	0.32
MAPE	0.68
MAE	0.17
RMSE	0.57
R <sup>2</sup>	0.99

Filtrazione per le slow feature in ogni livello:

Filtration level 0	40%
Filtration level 1	27%
Filtration level 2	40%
Filtration level 3	39%
Filtration level 4	39%
Filtration level 5	38%
Filtration level 6	44%
Filtration level 7	61%

- *aggregated*:

Risultati dei KPI calcolati considerando l'intero dataset e non solo una colonna:

MSE	7.94
MAPE	1.03
MAE	1.20
RMSE	2.82
R^2	0.89

Filtrazione per le slow feature in ogni livello:

Filtration level 0	43%
Filtration level 1	27%
Filtration level 2	34%
Filtration level 3	32%
Filtration level 4	32%
Filtration level 5	32%
Filtration level 6	36%
Filtration level 7	43%

- *meteo (no rhum<sub>t</sub>) + aggregated*:

Risultati dei KPI calcolati considerando l'intero dataset e non solo una colonna:

MSE	0.67
MAPE	0.70
MAE	0.28
RMSE	0.82
R^2	0.99

Filtrazione per le slow feature in ogni livello:

Filtration level 0	40%
Filtration level 1	29%
Filtration level 2	40%
Filtration level 3	36%
Filtration level 4	35%
Filtration level 5	35%
Filtration level 6	39%
Filtration level 7	53%

- *meteo (no rhum<sub>t</sub>) + aggregated (no holidays)*:

Risultati dei KPI calcolati considerando l'intero dataset e non solo una colonna:

MSE	0.67
MAPE	0.73
MAE	0.28
RMSE	0.82
R <sup>2</sup>	0.99

Filtrazione per le slow feature in ogni livello:

Filtration level 0	40%
Filtration level 1	29%
Filtration level 2	40%
Filtration level 3	36%
Filtration level 4	35%
Filtration level 5	35%
Filtration level 6	39%
Filtration level 7	53%

- *aggregated (no holidays)*:

Risultati dei KPI calcolati considerando l'intero dataset e non solo una colonna:

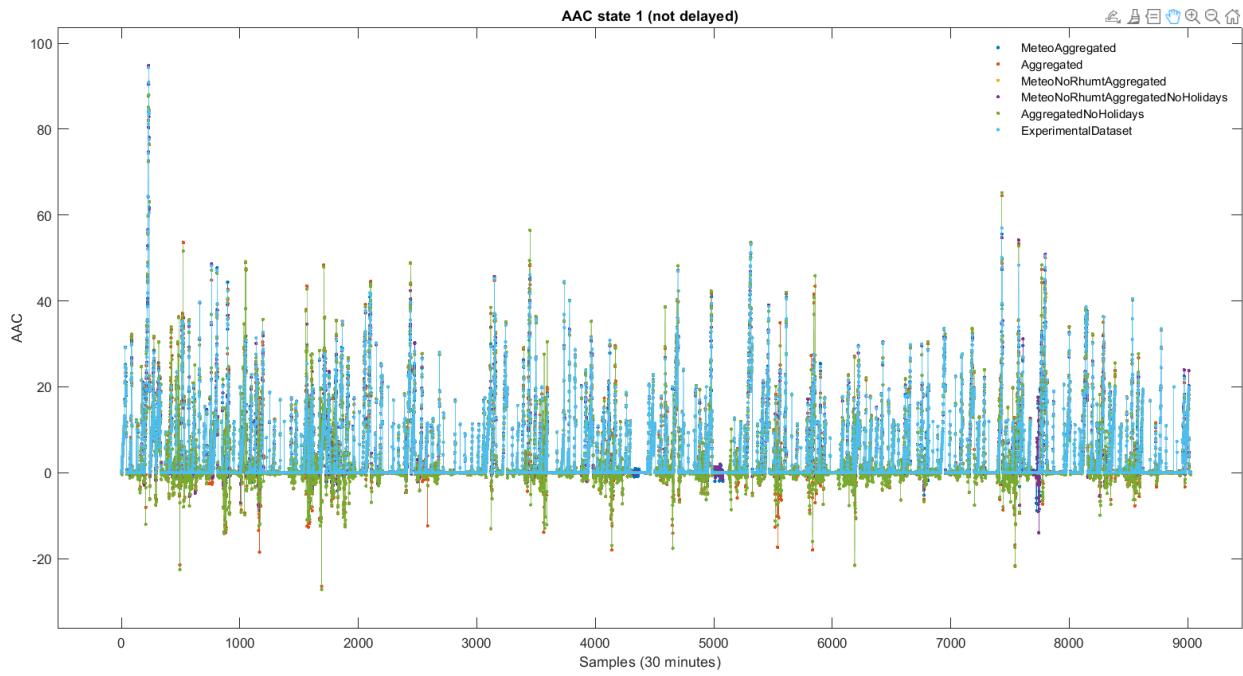
MSE	2.45e+01
MAPE	1.03
MAE	1.34
RMSE	4.95
R <sup>2</sup>	0.81

Filtrazione per le slow feature in ogni livello:

Filtration level 0	43%
Filtration level 1	27%
Filtration level 2	33%
Filtration level 3	31%
Filtration level 4	31%
Filtration level 5	32%
Filtration level 6	36%
Filtration level 7	43%

Viene presentato un plot con tutte le evoluzioni dello stato non ritardato (corrispondente alla prima colonna delle matrici ricostruite) per tutte le prove descritte in precedenza.

Grafico training:



Zoom:

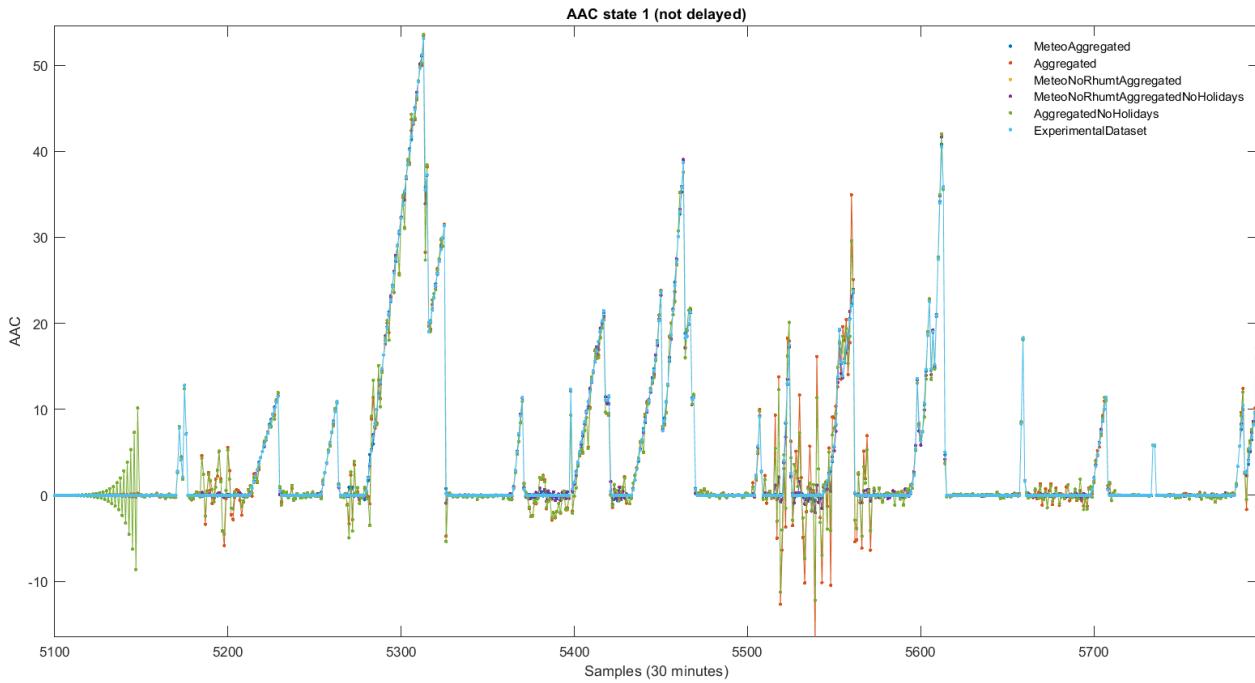


Figura 4.74

Si nota dal grafico e dai KPI come gli ingressi privati delle informazioni riguardanti il meteo vadano a peggiorare di parecchio i risultati della ricostruzione dello stato.

## 4.4 Confronto DMDc Vs. mrDMDc

Confronto con tabella KPI per i risultati ottenuti nel training (per il test ove specificato).

### 4.4.1 SRU

	<b>MSE</b>	<b>MAPE</b>	<b>MAE</b>	<b>RMSE</b>	<b>R2</b>
<b>DMDc</b>	6.95e-04	6.84e-02	1.80e-02	2.64e-02	0.63
<b>mrDMDc</b>	8.18e-19	1.19e-09	3.45e-10	9.04e-10	0.99

### 4.4.2 Sistema Sintetico con 5 Ingressi con ritardi dal Sistema SRU

	<b>MSE</b>	<b>MAPE</b>	<b>MAE</b>	<b>RMSE</b>	<b>R2</b>
<b>DMDc</b>	1.32e+15	2.23	1.12e+07	3.64e+07	0.33
<b>mrDMDc</b>	1.32e+13	0.34	3.79e+05	3.63e+06	0.98

### 4.4.3 Sistema sintetico con ingresso n.5 non ritardato dai dati SRU

	<b>MSE</b>	<b>MAPE</b>	<b>MAE</b>	<b>RMSE</b>	<b>R2</b>
<b>DMDc</b>	9.93e+14	1.06e+02	5.18e+06	3.15e+07	-1.04e+01
<b>mrDMDc</b>	9.92e+14	2.67e+01	4.78e+06	3.15e+07	-1.07e+01

### 4.4.4 V2G

#### 4.4.4.1 Stato non ritardato ed ingressi non ritardati

	<b>MSE</b>	<b>MAPE</b>	<b>MAE</b>	<b>RMSE</b>	<b>R2</b>
<b>DMDc</b>	1.15e+01	0.94	1.77	3.39	0.80
<b>mrDMDc</b>	8.40	1.61	1.60	2.90	0.86

#### 4.4.4.2 Stato ritardato ed ingressi non ritardati

	<b>MSE</b>	<b>MAPE</b>	<b>MAE</b>	<b>RMSE</b>	<b>R2</b>
<b>DMDc</b>	7.09e+01	2.34	5.77	8.42	-1.21e+01
<b>mrDMDc</b>	1.06e+02	3.34	6.22	1.03e+01	-0.26

#### 4.4.4.3 Stato ritardato ed ingressi ritardati

- *meteo + aggregated:*

	Training					Test				
	MSE	MAPE	MAE	RMSE	R2	MSE	MAPE	MAE	RMSE	R2
<b>DMDc</b>	9.10	1.88	1.74	3.02	0.85	1.18e+01	1.01	2.02	3.44	0.82
<b>mrDMDc</b>	0.32	0.68	0.17	0.57	0.99					

- *aggregated:*

	Training					Test				
	MSE	MAPE	MAE	RMSE	R2	MSE	MAPE	MAE	RMSE	R2
<b>DMDc</b>	9.63	0.97	1.70	3.10	0.84	1.19e+01	1.43	1.92	3.45	0.82
<b>mrDMDc</b>	7.94	1.03	1.20	2.82	0.89					

- *meteo (no rhum<sub>t</sub>) + aggregated:*

	Training					Test				
	MSE	MAPE	MAE	RMSE	R2	MSE	MAPE	MAE	RMSE	R2
<b>DMDc</b>	9.20	1.09	1.73	3.03	0.85	1.19e+01	1.11	2.00	3.45	0.82
<b>mrDMDc</b>	0.67	0.70	0.28	0.82	0.99					

- *meteo (no rhum<sub>t</sub>) + aggregated (no holidays):*

	Training					Test				
	MSE	MAPE	MAE	RMSE	R2	MSE	MAPE	MAE	RMSE	R2
<b>DMDc</b>	9.27	1.16	1.72	3.04	0.85	1.18e+01	1.02	1.97	3.44	0.82
<b>mrDMDc</b>	0.67	0.73	0.28	0.82	0.99					

- *aggregated (no holidays):*

	Training					Test				
	MSE	MAPE	MAE	RMS E	R2	MSE	MAP E	MA E	RMS E	R2
<b>DMDc</b>	9.70	0.99	1.69	3.11	0.84	1.18e+0	1.12	1.89	3.44	0.82
<b>mrDMDc</b>	2.45e +01	1.03	1.34	4.95	0.81					

## 5 Conclusioni

Lo scopo di questa tesi era l'implementazione del multi resolution DMDc (mrDMDc). Tale scopo è stato raggiunto percorrendo un excursus implementativo che a partire dal DMD, poi modificato in DMD con il controllo (DMDc) ha permesso di implementare un algoritmo iterativo basato sul mrDMD che includesse segnali di ingressi esogeni di controllo.

La metodologia e lo sforzo implementativo sono stati documentati approfonditamente sia dal punto di vista algoritmico quanto di sviluppo del codice. Il codice del mrDMD è stato ampliato con aggiunta di funzionalità quali:

- il salvataggio delle informazioni relative ai nodi
- la filtrazione ed eliminazione delle dinamiche ad ogni iterazione,
- il metodo *fit* della classe DMDc utile per la modellizzazione del sistema dinamico e alla ricostruzione eseguita dal DMDc. .

In ogni caso, dai risultati ottenuti in precedenza, è possibile considerare questa attuale implementazione dell'mrDMDc come funzionante se presa in considerazione la ricostruzione a passo  $dt = step$ , ed in generale i risultati ottenuti sono sempre migliori del DMDc; chiaramente i risultati ottenuti dal mrDMDc dipendono dall'eventuale troncamento e da quanto si filtrano le dinamiche.

Gli sviluppi futuri riguardanti questo algoritmo, per verificare se esso lavora correttamente e se riesce a identificare correttamente le dinamiche riguardanti il sistema che si sta analizzando, sono:

- Un ulteriore analisi sull'implementazione della ricostruzione. E' necessario indagare sul motivo cui la ricostruzione tramite dinamiche ottenute con passo  $dt=step$  e poi riscalate a passo  $dt = 1$  con passaggio intermedio nel tempo-continuo, generi dei segnali con un'ampiezza eccessiva
- Implementazione della ricostruzione in un dataset di test, quindi una previsione su un dataset non utilizzato in fase di training, che prenda in ingresso nuovi input e ne fornisca un risultato basato sul modello ottenuto nel training.

L'implementazione avanzata del DMDc e del mrDMDc è stata applicata su tre set di dati diverse:

- Un set di dati sintetico con dinamiche note
- Un set di dati proveniente da un sistema di Sulfur Recovery Unit come applicazione di soft sensors
- Un set di dati per la previsione della disponibilità aggregata a cedere energia nel contesto del Vehicle-to-Grid

Nelle tre applicazioni sono state effettuate diverse prove al variare di parametri algoritmici critici e sono state confrontate le performance mediante dei key performance indicators standad quali MAPE%, MAE, R^2 ed il confronto diretto delle serie temporali sperimentaliste ricostruite dai modelli identificati.

## 6 Bibliografia

- [1] Wikipedia contributors. *Dynamic Mode Decomposition*. URL: [https://en.wikipedia.org/wiki/Dynamic\\_mode\\_decomposition](https://en.wikipedia.org/wiki/Dynamic_mode_decomposition)
- [2] Kutz, Jose Nathan, Brunton, Steven L., Brunton, Bingni W. And Proctor, Joshua L.. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*: SIAM, 2016.
- [3] Wikipedia contributors. *Moore–Penrose pseudoinverse*. URL: [https://en.wikipedia.org/wiki/Moore%20%93Penrose\\_inverse](https://en.wikipedia.org/wiki/Moore%20%93Penrose_inverse)
- [4] Wikipedia contributors. *Singular Value Decomposition*. URL: [https://en.wikipedia.org/wiki/Singular\\_value\\_decomposition](https://en.wikipedia.org/wiki/Singular_value_decomposition)
- [5] Robert Taylor. *Optimal Singular Value Hard Threshold*. URL: <https://humaticlabs.com/blog/optimal-svht/>
- [6] Robert Taylor. *Sparsity-promoting DMD*. URL: <https://humaticlabs.com/blog/spdmd-python/>
- [7] Nicola Demo et al. *PyDMD*. URL: <https://github.com/mathLab/PyDMD>
- [8] Ilhan Polat. *Harold library, a control systems package for Python*. URL: <https://pypi.org/project/harold/>
- [9] *Control Systems Library for Python*. <http://github.com/python-control/python-control>
- [10] Gabriele Rinaldi. *Data-Driven-Methods*. <https://github.com/GabrieleRinaldi/Data-Driven-methods.git>
- [11] Luca Patanè e Maria Gabriella Xibilia. «*Echo-state networks for soft sensor design in an SRU process*». In: *Information Sciences* (2021), pp. 195–214.
- [12] Luigi Fortuna et al. «*Soft Sensors for Monitoring and Control of Industrial Processes*». In: *Soft Sensors for Monitoring and Control of Industrial Processes*. 2007.
- [13] Luca Patanè, Francesca Sapuppo e Maria Gabriella Xibilia “*A comprehensive data analysis for aggregate capacity forecasting in Vehicle-to-Grid applications*” In: *ICCAD24*.
- [14] Scikit-learn: *Machine Learning in Python*, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011. <https://scikit-learn.org/stable/about.html#citing-scikit-learn>
- [15] G. Oh, D. J. Leblanc, and H. Peng, *Vehicle energy dataset (VED), a large-scale dataset for vehicle energy consumption research*, IEEE Trans Intel Trans Sys, vol. 23, no. 4, pp. 3302–3312, 2022.

[16] *Meteo stat python api*, <https://dev.meteostat.net/python/>

[17] “*Ann arbour school breaks*” <https://annarborwithkids.com/articles/winterspring-break-schedules-2022/>

[18] “*Michigan ors non-business days*” <https://www.michigan.gov/psru/orsnon-business-days>