Transmitting information by controlling nonlinear oscillators

Camilloni Andrea, Caponi Luca, Giri Matteo

15 luglio 2021

Sommario

La trasmissione sicura di informazioni può essere gestita attraverso la perturbazione di oscillatori non lineari caotici. Questo è possibile grazie all'utilizzo di un principio, detto "ITVC" (Information transmission via control) che afferma che ogni controller usato per sincronizzare perfettamente una coppia di oscillatori non lineari, inclusi quelli con andamento caotico, può essere usato come decodificatore nel processo del recupero dell'informazione. L'obiettivo del nostro progetto è quello di verificare che una coppia di oscillatori non lineari, che siano uguali o meno, possa essere effettivamente sincronizzata tramite i metodi descritti dagli autori della ricerca.

1 Introduzione

L'idea alla base del progetto è quella di utilizzare un controller che permetta di sincronizzare una coppia di oscillatori, un master in trasmissione e uno slave in ricezione, basandosi sul principio noto come ITVC.

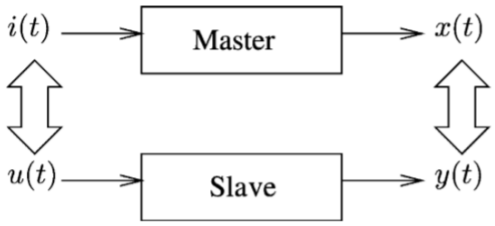

Figura 1: Sistema Master - Slave

1.1 Principio ITVC

Si consideri il seguente sistema di equazioni:

$$\begin{cases} \dot{x}(t) = f(x(t)) + i(t) \\ \dot{y}(t) = \hat{f}(y(t)) + u(t) \\ s(t) = h(x(t)) + \eta(t) \end{cases} \tag{1}$$

dove $x, y \in \Re^n$ sono vettori di stato, rispettivamente, del master e dello slave, $f, \hat{f} : \Re^n \to \Re^n$ sono funzioni vettoriali che descrivono i 2 oscillatori, $s \in \Re$ è il segnale trasmesso, $i(t)$ è l'informazione che si vuole trasmettere tramite il master, $u(t)$ è il segnale di controllo dello slave, $h(x) : \Re^n \to \Re$ è una funzione di "mapping" del vettore di stato in uno scalare, la quale andrà a costituire l'informazione trasmessa in condizioni ideali,cioè in assenza di rumore, ed infine $\eta(t)$ è il rumore presente in trasmissione.

Sia l'errore di sincronizzazione

$$e = x - y \tag{2}$$

e supponendo che la coppia di oscillatori sia identica, e quindi abbia stesso campo vettoriale, cioè $f = \hat{f}$, allora $\lim_{t\to\infty} \|e\| = 0 \Rightarrow \lim_{t\to\infty} \|i(t) - u(t)\| = 0$, dove $\|.\|$ denota la norma Euclidea.
Se l'errore è asintotico a 0, allora si può affermare che il segnale di controllo iniettato nello slave diviene identico all'informazione trasmessa, permettendo così il recupero di quest'ultima.

1.2 Circuito di Chua

Per creare la coppia di oscillatori master e slave abbiamo utilizzato il circuito di Chua, come mostrato nella Figura 2. Il circuito di Chua è un semplice circui-

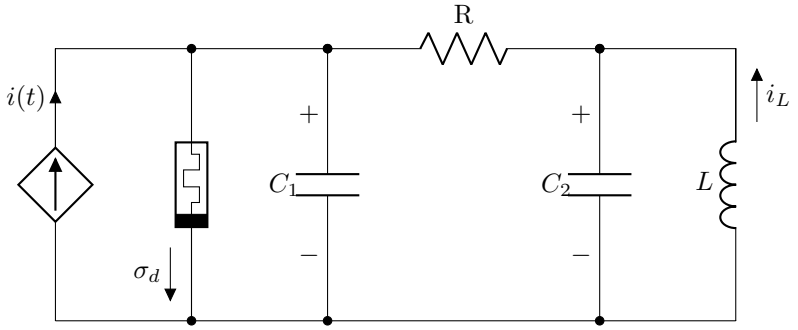

Figura 2: Circuito di Chua.

to elettronico che permette di ricreare il classico comportamento caotico (vedi Figura 3). Con comportamento caotico si intende che l'oscillatore genera onde non periodiche (che non si ripetono mai nel tempo), seppure deterministiche.

I bipoli presenti sono due condensatori (C_1 e C_2), un induttore L, un generatore interno di corrente ed un *memristore*. Il memristore è un bipolo elettrico introdotto nel 1971 e realizzato fisicamente solo 37 anni più tardi, nel 2008. L'idea primordiale era quella di trovare un componente circuitale che relazionasse tra loro flusso magnetico $\phi(t)$ e carica elettrica q(t): queste due variabili fondamentali erano infatti prive di correlazione prima degli studi condotti da Chua[3].

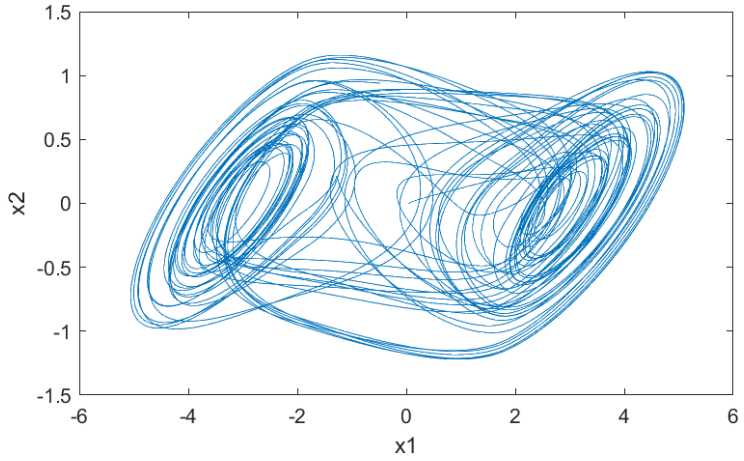

Figura 3: Andamento Caotico

La scelta del nome "memristore" è giustificata da una peculiare proprietà dell'elemento: mantenendo costante la corrente i(t) (o la tensione v(t)) che lo attraversa esso si comporta come un resistore tempo variante e la sua resistenza(conduttanza) dipende dallo stato presente ma anche dagli stati precedenti, che vengono quindi MEMORIZZATI (da qui il prefisso mem-). E' un componente passivo (in ogni istante di tempo t risulta l'energia istantanea e(t)≥0) se e solo se il parametro caratteristico M(ϕ) ("memristenza") che lega proporzionalmente tensione v(t) e corrente i(t) e maggiore o uguale a zero. Inoltre, il memristore è un elemento non lineare che può essere dunque implementato tramite l'utilizzo di amplificatori operazionali. Il memristore può essere realizzato progettando un circuito equivalente composto da 2 op-amp e delle resistenze, come mostrato in figura 4(maggiori dettagli sono presentati nel riferimento [4])

Un oscillatore costruito implementando il circuito di Chua può essere descritto dal seguente sistema:

$$\begin{cases} C_1\dot{x}_1 = \dfrac{x_2 - x_1}{R} + \sigma(x_1) + i \\ C_2\dot{x}_2 = \dfrac{x_1 - x_2}{R} + x_3 \\ L\dot{x}_3 = -x_2 \end{cases} \tag{3}$$

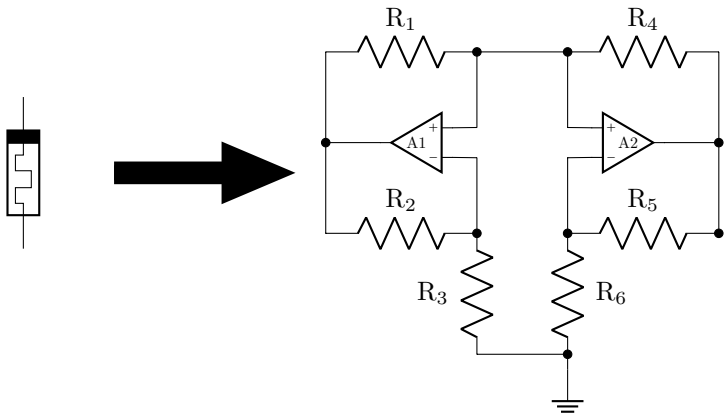

Figura 4: Circuito equivalente al memristore (Relazione costitutiva è $i = \sigma(v)$).

dove $\sigma(x) = m_0 x + 0.5(m_1 - m_0)[|x + B_p| - |x - B_p|]$ è la relazione costitutiva del memristore.

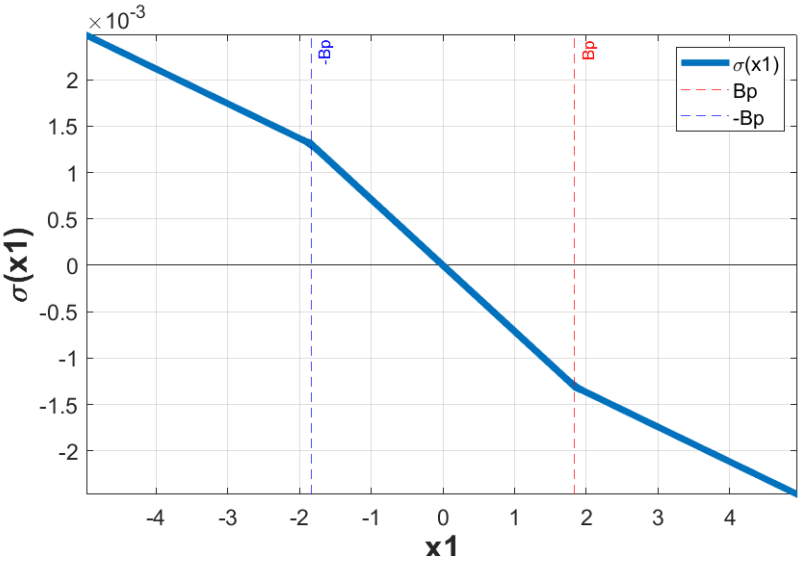

Figura 5: Relazione tensione - corrente sul memristore

In figura 5 viene mostrata la relazione che lega $\sigma(x)$ e x, cioè corrente e tensione sul memristore. I tre coefficienti m_0, m_1, B_p sono parametri caratteristici del memristore; m_0 e m_1 sono rispettivamente due conduttanze e rappresentano nel grafico (in figura 5) le 2 rispettive pendenze della relazione corrente-tensione, mentre B_p costituisce la tensione di "Breakpoint".

Il vettore di stato che consideriamo nelle simulazioni e nell'implementazione è il seguente:

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

dove x_1, x_2, x_3 rappresentano rispettivamente la tensione sul condensatore C_1, la tensione su C_2 e la corrente sull'induttore L.

1.3 Controller

Il controller è un dispositivo che monitora una variabile di processo e la confronta con un punto di riferimento. La differenza tra il valore attuale della variabile di processo e il valore desiderato, chiamato errore di sincronizzazione, è usato come feedback per portare la variabile allo stesso valore del punto di riferimento.
Nel corso del progetto utilizziamo due tipi di controller per implementare la sincronizzazione tra la coppia di oscillatori: un controller dipendente dalla struttura e un controller non dipendente dalla struttura.
Il controller dipendente dalla struttura (Structure-dependent controller) funziona solo nel caso in cui i due oscillatori hanno la stessa struttura fisica, e quindi solo nel caso in cui il trasmettitore e il ricevitore sono identici da un punto di vista circuitale e parametrico.
Il controller non dipendente dalla struttura (Structure-indipendent controller) invece non ha questa limitazione. Può essere utilizzato anche nel caso in cui i due oscillatori non abbiano la stessa struttura.

2 Implementazione

Indipendentemente dalla struttura che sfrutteremo abbiamo effettuato la simulazione su Matlab, implementando la funzione ode4 [2], la quale ci permette di risolvere le equazioni differenziali del sistema (3).
Una funzionalità introdotta da ode4, rispetto le tradizionali funzioni ode già incluse in Matlab, è la possibilità di passare variabili esterne alla funzione che poi utilizzeremo per effettuare il Plot dei grafici.
Come primo punto della nostra simulazione in Matlab abbiamo organizzato il nostro progetto in 2 grandi script:

- Command Window
- Oscillators

2.1 Command Window

Nel primo script dichiariamo inizialmente le variabili relative al controllore che implementeremo, il tempo di esecuzione della simulazione, ed il relativo passo che sfrutteremo per integrare le equazioni differenziali, e le condizioni iniziali

rispettive ai 2 oscillatori. Dichiariamo inoltre in questa prima sezione il segnale di disturbo che si presenterà sul nostro segnale trasmesso.
A questo punto chiamiamo la funzione ode4:

x = ode4('Oscillators',t,[x1,x2,x3,x4,x5,x6,e],tsync,tnoise,noise);

la quale risolve le equazioni del sistema (3) relative rispettivamente al Master e lo Slave, implementando un controller che ci permetterà il calcolo dell'errore (2) e inoltre la sincronizzazione stessa dei segnali.
La funzione ci restituisce una matrice x, la quale sarà composta da 7 colonne relative rispettivamente ai vettori di stato della coppia di oscillatori, e una colonna rappresentante l'errore.
Vista l'impossibilità di plottare il segnale di controllo generato nel secondo script, poiché è impossibile accedervi, dati i vettori di stato estrapolati della funzione ode abbiamo ricreato il segnale trasmesso e simulato nuovamente il controllo, così da poterlo riportare sui grafici e confrontarlo per verificare l'effettivo recupero dell'informazione(Vedi Fig.6).

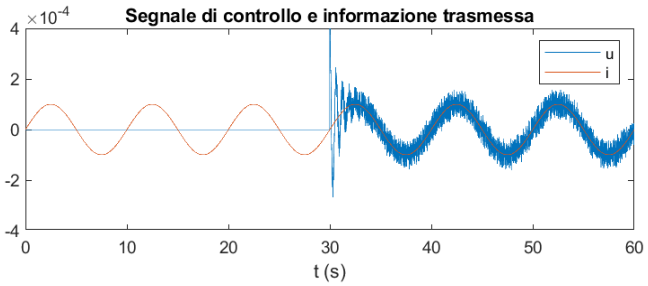

Figura 6: dove $u(t)$=segnale di controllo, $i(t)$=informazione trasmessa

Nella parte finale di questo script abbiamo effettuato la creazione dei grafici di cui discuteremo in seguito nelle conclusioni.

2.2 Oscillators

Nello script Oscillators si concentra "il cuore" del progetto. Dopo aver effettuato la dichiarazione dei parametri caratterizzanti i due oscillatori e il controller implementato, il programma si struttura nel seguente modo:

1. Si crea un segnale sinusoidale $i(t)$ che rappresenta l'informazione che si dovrà trasmettere, tramite l'oscillatore (Master) in trasmissione.

2. Si inizializza un vettore, denotato 'oscillators', a cui vengono assegnati i valori ottenuti risolvendo il sistema (3) ad ogni istante di tempo, sia per il master che per lo slave. Ad ogni iterazione della funzione ode4,

verranno restituiti i valori contenuti in 'oscillators', risolvendo le equazioni differenziali, così da poter riempire la matrice x presente nel primo script.

3. Prima di poter risolvere il sistema (3) dello Slave, il quale permette di recuperare l'informazione sincronizzandosi a partire da un determinato istante, viene generato un segnale di rumore il quale simula la trasmissione dell'informazione in presenza di disturbi.

4. Quindi nella parte centrale dello script viene calcolato l'errore (5), purchè siano note le condizioni iniziali, dal quale implementando un controller (Sezione 1.3), si pùo ricavare un segnale di controllo $u(t)$ che permette di sincronizzare lo slave, e di recuperare l'informazione stessa.

5. A questo punto noto il segnale di controllo, viene risolto il sistema (3) dell'oscillatore in ricezione.

2.3 Structure-dependent

Il controller di tipo structure-dependent richiede l'impiego di due oscillatori con stesse caratteristiche; quindi in simulazione utilizzeremo i seguenti parametri per entrambi i sistemi:

Bipoli	Valori numerici
C_1	24 μF
C_2	236 μF
L	42 H
R	1660 Ω

Parametri del memristore	Valori numerici
m_0	-0.374 mS
m_1	-0.711 mS
B_p	1.84 V

Parametri del controllore	Valori numerici
k_p	0.05
k_i	0.1

Abbiamo implementato l'oscillatore che fa da master tramite il sistema di equazioni (3), mentre possiamo descrivere lo slave tramite il seguente sistema:

$$\begin{cases} C_1\dot{y}_1 = \dfrac{y_2 - y_1}{R} + \sigma(y_1) + u(t) \\ C_2\dot{y}_2 = \dfrac{y_1 - y_2}{R} + y_3 \\ L\dot{y}_3 = -y_2 \end{cases} \tag{4}$$

I parametri circuitali dei due sistemi sono gli stessi (poichè l'implementazione è structure-dependent). Ciò che cambia è che nel sistema dello slave il

segnale trasmesso i(t) viene sostituito con il segnale di controllo u(t).

Per l'implementazione del controller structure-dependent si utilizzano le seguenti equazioni:

$$e_T(t) = s(t) - y_1(t), \tag{5}$$

$$\hat{m}(t) = k_p e_T(t) + k_i \int_0^t e_T(\theta) d\theta, \tag{6}$$

$$u(t) = -\frac{y_2 - y_1}{R} + \hat{m}(t) \tag{7}$$

Per prima cosa si calcola l'errore di sincronizzazione (5), definito come la differenza tra la variabile di stato in presenza di rumore $s(t) = x_1(t) + \eta(t)$ e la variabile di stato $y_1(t)$, relativi ai due sistemi, poi si calcola l'azione di controllo (6). Infine calcoliamo il segnale di controllo (7) che andrà a sincronizzare l'oscillatore in ricezione e che permetterà, a partire da un determinato istante, il recupero dell'informazione. C'e' una spiegazione del perche' $u(t)$ ha proprio questa espressione ? Forse si puo' ragionare in termini circuitali ? Un esempio di tale segnale si può osservare in Fig. 6.

2.4 Structure-independent

Un controller di tipo structure independent non richiede sistemi identici per la sincronizzazione. Per la simulazione abbiamo comunque utilizzato parametri molto simili fra loro, poiché non forniti [1].

Le equazioni caratterizzanti il controller sono:

$$e_T(t) = s(t) - y_1(t), \tag{8}$$

$$k_p(e_T) = k_p^{min} + k_p^{max}\left(1 - \frac{1}{1 + \delta_p e_T^2}\right), \tag{9}$$

$$u(t) = k_p(e_T(t))e_T(t), \tag{10}$$

dove $k_p(t)$ è l'azione di controllo ($k_p^{min} = 0.0007$; $k_p^{max} = 0.01$; $\delta_p = 1$).

3 Conclusioni

I risultati ottenuti dalle simulazioni sono pressoché simili. In entrambe le simulazioni siamo riusciti ad implementare i due controller, avviando l'azione di controllo a partire da un istante $t = 30$ e come si può vedere nella figura 7 , gli oscillatori assumono da $t = 30$ man mano lo stesso stato $x_1(t)$.

Implementando le due diverse strutture abbiamo riscontrato un importante risultato: abbiamo osservato che nell'implementazione di un controller structure dependent e quindi, di due oscillatori identici, l'errore di sincronizzazione, a partire dall'avvio del controllore, **"corre" più velocemente a 0**, e così rimane(vedi Fig. 8); mentre nella structure independent non va mai a 0, ma oscilla fra valori compresi tra 0.01 e -0.01(vedi Fig. 9).

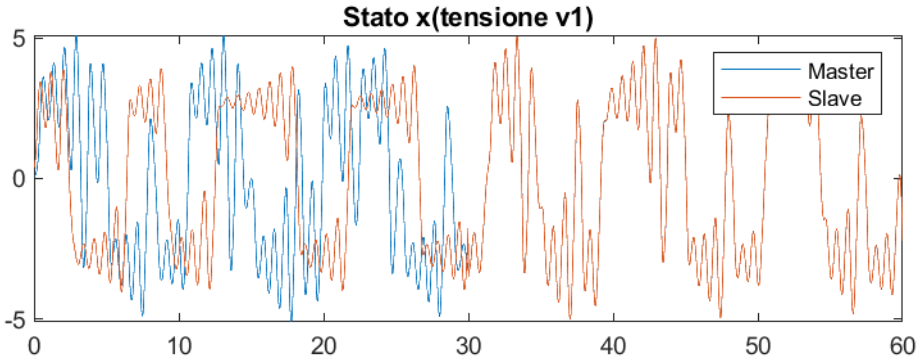

Figura 7: rispettive tensioni sul condensatore C_1 del master e dello slave

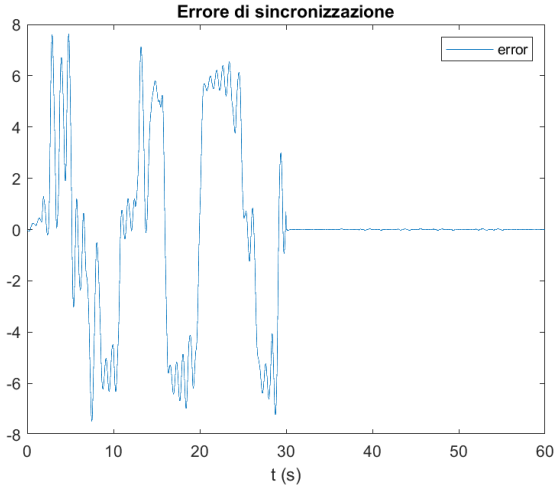

Figura 8: Errore di sincronizzazione in una struttura dependent

Come risultato finale mostriamo rispettivamente il recupero dell'informazione trasmessa come segnale sinusoidale $i(t) = 0.1 \sin\left(\frac{2\pi}{10}t\right) mA$, a partire da $t = 30$, nei due casi, structure dependent in figura 10 e structure independent in figura 11

Riferimenti bibliografici

[1] L.A.B. Tôrres and L.A. Aguirre, "Transmitting information by controlling nonlinear oscillators", *Physica D: Nonlinear Phenomena*, Vol. 196, No. 3–4, pp. 387 – 406, 2004

[2] ODE4: https://it.mathworks.com/matlabcentral/fileexchange/59044-ode4-gives-more-accurate-results-than-ode45-ode23-ode23s

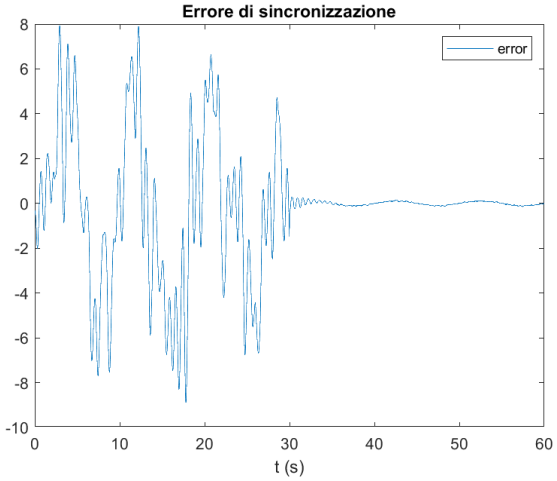

Figura 9: Errore di sincronizzazione in una structure independent

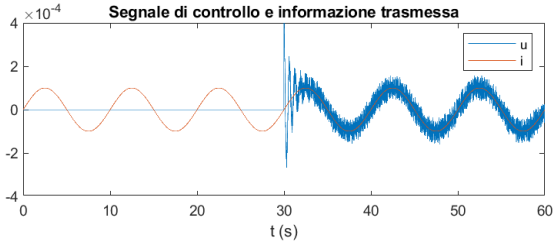

Figura 10: Recupero dell' informazione nello structure-dependent

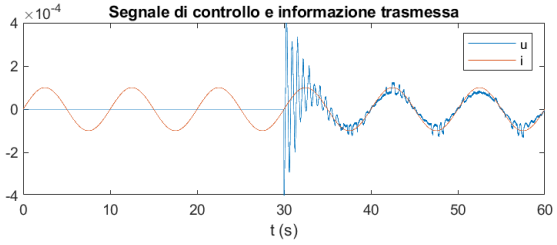

Figura 11: Recupero dell' informazione nello structure-independent

[3] Leon O. Chua, "Memristor-The missing circuit element"

[4] L.A.B. Tôrres and L.A. Aguirre "Inductorless Chua's circuit"