Università degli Studi di Padova,
corso di Laurea in Ingegneria dell'Automazione

Corso di Progettazione di Sistemi di Controllo, *a.a. 2012/2013*

Docente: Luca Schenato

Sincronizzazione di orologi in WSN mediante controllo PI con autotuning dei guadagni.

Fuser Federico
Chientaroli Roberto
Sacchetto Fabio

18 febbraio 2013

Abstract

In questo progetto viene proposto un innovativo controllo PI per la sincronizzazione temporale di dispositivi elettronici che fanno uso di un timer interno.
L'algoritmo è basato su un controllore Proporzionale-Integrativo atto alla risoluzione di un problema di consensus e richiede un protocollo di comunicazione broadcast tra i nodi della rete. La parte Proporzionale del controllore elimina la differenza di offset tra gli orologi mentre la parte Integrale compensa le loro diverse velocità di aggiornamento.
Viene inoltre studiata formalmente la convergenza del protocollo di sincronizzazione quando il grafo di comunicazione è noto e fissato. Per concludere viene simulato l'algoritmo di base in modo da confermarne la correttezza e successivamente vengono proposte descrizioni, simulazioni ed osservazioni per i molteplici confronti effettuati con le varie soluzioni proposte, atte a migliorare la robustezza, il tempo di convergenza e la stabilità.

Indice

Introduzione

Le reti di dispositivi elettronici sono attualmente sempre più utilizzate in qualsiasi campo riguardante i sistemi interconnessi, come ad esempio le reti di sensori utili ad effettuare misure distribuite su larga scala. La rilevazione di fenomeni fisici viene attualmente eseguita mediante l'uso di sensori che, grazie all'avanzamento della tecnologia, stanno diventando sempre più precisi, affidabili e sofisticati.
Il metodo che viene più spesso usato per lo scambio dei dati tra i sensori e con l'unità di controllo è il wireless, tecnologia che consente soprattutto l'abbandono appunto dell'ingombro dei cavi. Questo sistema consente di realizzare le Wireless Sensor Network, famose per il loro svariato utilizzo in molteplici settori industriali e informatici.
Il riferimento temporale è un'importante informazione che accompagna i dati gestiti da un dispositivo, per esempio ai fini del controllo (in real time) con l'ausilio di sensori decentralizzati e/o wireless. Essenziale quindi che gli agenti che compongono la rete siano globalmente coordinati e sincronizzati, cioè in particolare abbiano un comune ed omogeneo riferimento temporale.
La nascita di questo fatto è dovuta essenzialmente alla dinamica di ogni singolo orologio la quale in evoluzione libera viene governata da un proprio oscillatore interno di frequenza diversa per ogni orologio (dovuta a fabbricazione, status ambientale o usura) e caratterizzata da un'offset iniziale (dovuto ai diversi istanti iniziali degli orologi). Il problema fondamentale che quindi scaturisce dall'uso di queste WSN è quello della loro sincronizzazione temporale.

Mediante algoritmi atti alla risoluzione formale di questo problema si è riusciti in gran parte a soddisfare le richieste di mercato, presentando soluzioni di vario carattere, prestazione e affidabilità. Il lavoro svolto con questo progetto non consiste nello studio e progettazione di algoritmi innovativi in concorrenza ai molteplici già presentati ma vuole concentrarsi sulla messa a punto, miglioramento e ampliamento di metodologie già formalmente consolidate sotto il punto di vista teorico.

Stato dell'arte

Un classico approccio per risolvere il problema di sincronizzazione è quello della creazione di una struttura gerarchica ad albero nella quale tutti i nodi (foglie) si sincronizzano rispetto ad un nodo (radice).
Un approccio simile consiste nella suddivisione della rete in cluster, ognuno caratterizzato da una struttura gerarchica ad albero. Questi algoritmi però presentano il problema di non essere robusti e scalabili, infatti se fosse necessario aggiungere o togliere nodi dalla rete bisognerebbe aggiornare l'intera struttura ricostruendola ex novo. Esempi di protocolli che sfruttano queste caratteristiche sono le Network Time Protocol e le Precision Time Protocol.
Approcci alternativi consistono in algoritmi totalmente distribuiti in cui i nodi della WSN comunicano solo con i vicini eseguendo lo stesso tipo di codice per giungere al consensus, come per esempio il protocollo Average Time Synch. Con questa tecnica si ottiene una notevole robustezza e adattabilità.
Altri approcci sono basati su strategie non lineari quindi di difficile interpretazione e studio, come quello composto da una cascata di due algoritmi "least-squared" distribuiti oppure sulla cascata di due algoritmi di consensus di primo ordine.
Attualmente è stato presentato un protocollo distribuito basato sulla modellizzazione dell'orologio come un doppio integratore al quale si applica un controllo Proporzionale-Integrativo di tipo lineare. Semplicemente vengono compensati gli offset mediante la parte proporzionale e la velocità interna degli orologi mediante la parte integrale.
Su quest'ultimo approccio è basato il lavoro qui proposto.

Approccio proposto

La progettazione di un efficiente algoritmo di sincronizzazione si basa su molteplici fattori come la scelta del tipo di implementazione, del tipo di protocollo e di controllore da usare e delle prestazioni da raggiungere.
In questo progetto è stato scelto di utilizzare un controllo di tipo distribuito dove ogni nodo (agente) presente sulla rete esegue lo stesso tipo di algoritmo basandosi esclusivamente sui dati a propria disposizione[1] invece che sfruttare la conoscenza dell'intera rete.
Inoltre il protocollo di comunicazione su cui si basa l'algoritmo è di tipo asincrono in modo tale da coinvolgere nell'aggiornamento solamente una parte della rete alla volta, in particolare viene scelto un protocollo di tipo broadcast in modo tale che ogni nodo invia le informazioni contemporaneamente a tutti i propri vicini.
Il lavoro svolto si focalizza quindi nella progettazione dell'algoritmo basato sull'autotuning distribuito dei parametri usando il modello a doppio integratore dell'orologio in spazio di stato. In particolare sarà presentata una pratica versione dell'algoritmo di sincronizzazione PI basata su un protocollo di comunicazione broadcast e saranno proposti nuovi possibili approcci per effettuare l'autotuning distribuito dei parametri del controllore PI, con l'intento di massimizzare la velocità di convergenza al consenso, sia in termini di comportamento nel transitorio che in termini di comportamento a regime.
Il punto principale sul quale si è basato l'approccio proposto è stato il rilevamento dei parametri da cui dipendono i guadagni del controllore PI al fine di ottimizzarne e sfruttarne la dipendenza in favore del miglioramento dell'algoritmo. Come si vedrà questo approccio è sembrato intuitivamente corretto ed ha offerto i miglioramenti sperati.
In secondo luogo è stata addirittura complicata la struttura del controllore inserendo un termine Derivativo. Ne sono quindi state studiate le caratteristiche e il comportamento. Essendo quest'ultimo un nuovo approccio nel campo della sincronizzazione temporale, sono state comparate le prestazioni in relazione al classico controllore PI al fine di valutarne convenienza ed efficacia.

Sommario

- Capitolo 1: Formulazione del problema dalla quale si derivano i modelli matematici per lo studio formale del problema. Vengono qui descritte le ipotesi di partenza e le ipotesi semplificative, oltre ai metodi con il quale viene impostato ed affrontato il progetto.
- Capitolo 2: Descrizione del controllo Proporzionale-Integrativo applicato per la risoluzione del problema di sincronizzazione temporale, con analisi della convergenza dell'algoritmo. Vengono inoltre proposte e commentate le relative simulazioni dell'algoritmo. Inoltre si introduce l'implementazione dello stesso nell'architettura *SensLab*.
- Capitolo 3: Introduzione e derivazione dei nuovi algoritmi proposti, corredati da numerose simulazioni e commenti alle stesse. Inizialmente si procede con una semplice modifica dell'algoritmo base introdotto nel capitolo 2 e successivamente si attueranno dei nuovi metodi per effettuare auto-tuning mirato al parametro della parte Integrale del controllore.
- Capitolo 4:Viene proposta l'aggiunta del termine Derivativo al controllore e vengono confrontati i risultati con quelli ricavati in precedenza.
- Conclusioni: Con i vari risultati ottenuti si potrà globalmente commentare la conclusione del lavoro effettuato, evidenziando i pro e contro di ogni algoritmo studiato ed applicato. Inoltre vengono discusse interessanti varianti per il futuro sviluppo dell'algoritmo.

[1] come le informazioni ricevute esclusivamente dai propri vicini.

Capitolo 1

Formulazione del problema

1.1 Interpretazione del clock nelle WSN

Il modello ideale di clock digitale è costituito generalmente da un contatore che viene incrementato a frequenza fissa da un oscillatore interno (tipicamente al quarzo). In questo modo è possibile misurare intervalli di tempo costanti.
Nella realtà però la frequenza dell'oscillatore è una grandezza variabile funzione per esempio di agenti esterni, invecchiamento e usura.
Dato che queste variazioni rimangono tuttavia limitate durante il ciclo di funzionamento del clock interno, risulta possibile approssimare linearmente quest'ultimo, per ogni sensore, come $\tau(t) = \gamma t + \omega$, dove γ è lo skew (pendenza) mentre ω è l'offset.
Di conseguenza è possibile confrontare il clock interno tra nodi qualsiasi della rete per valutare la differenza tra offset e skew.
Se i nodi confrontati risultano perfettamente sincronizzati il loro skew relativo sarà pari a 1 cioè la loro frequenza di aggiornamento sarà la medesima mentre il loro offset relativo sarà nullo.

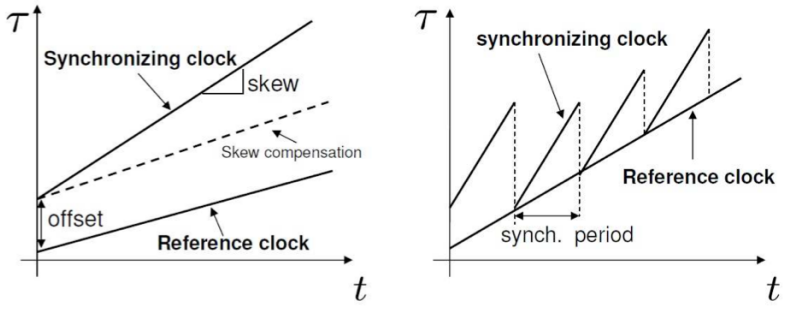


Figura 1.1: esempi di compensazione dello skew (a sinistra) e compensazione dell'offset (a destra)

1.2 Modello matematico del clock

La funzione comulativa

$$s(t) = \int_{-\infty}^{t} f(\rho) d\rho \tag{1.1}$$

descrive bene l'evoluzione temporale del contatore implementato nell'orologio, dove

$$f(t) = \sum_{k=0}^{\infty} \delta(t - t(k)) \tag{1.2}$$

e $t(k), k = 0, 1, 2, ...$ sono gli istanti in cui il contatore viene incrementato.
Si parte dal presupposto per cui il tempo scandito dall'orologio è una variabile incognita e quindi bisogna stimarla.
Per stimare il tempo si può quindi usare la funzione comulativa ricavando

$$\hat{t}(t_1) = \hat{t}(t_0) + \hat{\Delta}(t_1)[s(t_1) - s(t_0)] = \hat{t}(t_0) + \hat{\Delta}(t_1) \int_{t_0}^{t_1} f(\rho)d\rho \tag{1.3}$$

dove $\hat{\Delta}(t)$ è la stima del periodo di oscillazione reale $\frac{1}{f(t)}$[1].
Per facilitare la trattazione, dato che il periodo di clock è teoricamente breve, è conveniente approssimare la funzione $s(t)$ con una sua versione regolarizzata. In sostanza si pone inizialmente

$$f(t) = \frac{1}{t(k+1) - t(k)}, t \in [t(k), t(k+1)[\tag{1.4}$$

e successivamente si assumerà che tale valore di frequenza sia costante $f_i(t) = \bar{f}_i$ per ogni istante.
Entrambi i valori $\hat{t}(t)$ e $\hat{\Delta}(t)$ possono essere modificati quando sono presenti informazioni che potrebbero migliorare la stima del tempo e del periodo di oscillazione. Stabilito che queste modifiche vengano attuate all'istante di tempo T_{up}, si può facilmente esprimere la (1.3) come

$$\hat{t}(t) = \hat{t}(T_{up}^+) + \hat{\Delta}(T_{up}^+)\bar{f}(t - T_{up}^+) \tag{1.5}$$

dove

$$\hat{t}(T_{up}^+) = \hat{t}(T_{up}^-) + u' \tag{1.6}$$

e T_{up}^- e T_{up}^+ indicano rispettivamente l'istante di tempo precedente e successivo di T_{up}.
Ovviamente anche la stima del periodo di oscillazione sarà governata da una sua evoluzione temporale del tipo

$$\hat{\Delta}(t) = \hat{\Delta}(T_{up}^+) \tag{1.7}$$

dove

$$\hat{\Delta}(T_{up}^+) = \hat{\Delta}(T_{up}^-) + u'' \tag{1.8}$$

L'ingresso $u = [u'\ u'']^T$ caratterizzerà il controllo[2] applicato rispettivamente alle due stime.

Per semplificare la trattazione si può definire $x(t) := x(T_{up}^+)$ e di conseguenza si possono riscrivere le notazioni (1.5) e (1.7) seguendo una rappresentazione in spazio di stato ottenendo

$$x(t+1) = \begin{bmatrix} x'(t+1) \\ x''(t+1) \end{bmatrix} = \begin{bmatrix} \hat{t}(t+1) \\ \hat{\Delta}(t+1) \end{bmatrix} = \begin{bmatrix} 1 & \delta(t) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \hat{t}(t) \\ \hat{\Delta}(t) \end{bmatrix} \tag{1.9}$$

dove $\delta(t)$ è l'intervallo di tempo trascorso tra due istanti di aggiornamento.

1.3 Protocollo di comunicazione

Per poter analizzare come i dispositivi collegati alla rete aggiornano il proprio stato in funzione delle informazioni inviate e/o ricevute a valle di una comunicazione con altri nodi della rete e/o ricevute da altri nodi della rete, è bene definire le modalità con cui questi interagiscono tra di loro.
Questo progetto è fondato sull'utilizzo di un protocollo di comunicazione asimmetrico caratterizzato quindi da un nodo che invia ad uno o più vicini dell'informazione. Il nodo o i nodi destinatari aggiornano il proprio stato in funzione dell'informazione ricevuta.
Il *Gossip Asimmetrico* è un protocollo di comunicazione basato sull'invio di un pacchetto da parte di un nodo della rete verso un solo nodo vicino. Il *Broadcast* invece si basa sull'invio del pacchetto contemporaneamente a tutti i vicini del nodo mittente. Intuitivamente si può capire come il secondo protocollo sia estremamente più veloce del primo, quindi nel nostro caso si può immaginare come la convergenza al consensus possa avvenire con velocità maggiore.

[1] Risulta una buona scelta inizializzare $\hat{\Delta}(t) = \frac{1}{\bar{f}}$.

[2] questo aspetto verrà trattato nel capitolo seguente.

Proprio da questo fatto è stato scelto di utilizzare il protocollo di tipo *broadcast* come modalità di comunicazione tra i dispositivi della nostra WSN.

E' utile svolgere un'ulteriore ragionamento per capire in che modo verrà scelto il dispositivo che ad un certo istante invia dell'informazione sulla rete e soprattutto in quale momento questo accadrà. E' ben noto a livello teorico che la scelta puramente randomizzata del nodo che dovrà inviare l'informazione sulla rete risulta essere quella ottima mentre, per quanto riguarda la tempistica di trasmissione, gli istanti di comunicazione sono stati definiti come i campioni di un processo di Poisson di intensità $\lambda > 0$. Quest'ultimo simula il manifestarsi di eventi tra loro indipendenti e che accadano continuamente nel tempo.
Viene fissata una finestra temporale $[0\ T], T > 0$, durante la quale viene effettuata l'analisi, e degli istanti di tempo $t_{k,i}$ relativi al nodo i dove $k \in \mathbb{N}$ è il numero di eventi (le trasmissioni effettuate) nella finestra temporale. Per fare in modo che le trasmissioni siano viste come N processi di Poisson è bene definire l'istante di aggiornamento globale come:

$$t_k = \bigcup_{i=1}^{N} t_{i,k} \tag{1.10}$$

che può essere considerato a sua volta un processo di Poisson di intensità $TN\lambda$.
Di conseguenza anche l'intervallo di tempo $\delta(t)$ tra l'istante di aggiornamento e il precedente verrà d'ora in poi considerato come il tempo di "arrivo" del processo di Poisson di intensità $N\lambda$.
Viene inoltre aggiunta l'ipotesi che durante tutto il proseguo del lavoro non si considereranno la perdita d'informazione e i ritardi dovuti al tempo di trasmissione, eventi che nella realtà sono invece sempre presenti (anche se a volte possono essere compensati, per esempio, calcolando il tempo di trasmissione e ricezione di un pacchetto tramite comunicazione wireless).

A valle di queste ipotesi è bene definire la modalità di aggiornamento dello stato dei nodi che ricevono l'informazione in un certo istante temporale.
In generale al tempo t il nodo i invia il proprio stato $x_i(t) = \hat{t}_i(t)$ ai propri vicini. Uno dei vicini, il nodo j, aggiorna il proprio stato come $x_j(t+1) = \delta x_j(t) + (1-\delta)x_i(t)$ dove $\delta = [0\ 1]$ è il "peso dell'aggiornamento". L'intuizione suggerisce di porre $\delta = \frac{1}{2}$ in modo tale che il destinatario si porti alla media esatta tra il proprio e lo stato del mittente. Questo accorgimento verrà mantenuto per tutto il seguito del lavoro.
I nodi che non ricevono informazione non vengono aggiornati, quindi $x_j(t+1) = x_j(t)$.

Utilizzando quanto detto si può anticipare la definizione di un'importante matrice $K(t) \in \mathbb{R}^{N\times N}$, con N il numero dei nodi sulla rete:

$$K(t) = (1-\delta)\sum_{j\in\mathcal{N}_i} e_j(e_j - e_i)^T = \frac{1}{2}\sum_{j\in\mathcal{N}_i} e_j(e_j - e_i)^T \tag{1.11}$$

dove e_n è il vettore canonico n-esimo e $\mathcal{N}_i$ è l'insieme dei vicini del nodo i.
Questa matrice verrà successivamente usata durante l'applicazione della legge di controllo[3] per adeguarla ai soli nodi della rete che in un particolare momento stanno trasmettendo o ricevendo informazioni.

[3] si veda il capitolo successivo.

Capitolo 2

Controllo Proporzionale-Integrativo

Per quanto riguarda la progettazione del controllore utilizzato in questo lavoro, è bene d'ora in poi considerare l'intera rete di dispositivi e, seguendo quanto anticipato nella (1.9), vengono definiti i vettori:

$$x' = [x'_1 \cdots x'_N] \tag{2.1}$$
$$x'' = [x''_1 \cdots x''_N] \tag{2.2}$$
$$x = [x'\, x''] = [x'_1 \cdots x'_N\, x''_1 \cdots x''_N] \tag{2.3}$$

La legge di controllo lineare che descrive il controllore Porporzionale-Integrativo è della forma:

$$u(t_k) = \begin{bmatrix} u'(t_k) \\ u''(t_k) \end{bmatrix} = \mu \begin{bmatrix} 1 \\ \alpha \end{bmatrix} (x'_i(t_k) - x'_j(t_k)) \tag{2.4}$$

dove μ[1]$= \frac{1}{2}$ e $\alpha > 0$ è il parametro di controllo dell'azione integrale. Inoltre si ricorda che u' e u'' rappresentano, rispettivamente, gli ingressi di controllo della parte proporzionale e della parte integrale.
Applicando questa formulazione alla nota legge di aggiornamento dello stato $x(t+1) = x(t) + u(t)$ segue:

$$x'_i(t_{k+1}) = \frac{1}{2}(x'_i(t_k) + x'_j(t_k)) \tag{2.5}$$
$$x''_i(t_{k+1}) = x''_i(t_k) + \frac{\alpha}{2}(x'_j(t_k) - x'_i(t_k)) \tag{2.6}$$

che riscritta in forma matriciale, utilizzando le notazioni (2.1), (2.2), (2.3), porta alla formulazione:

$$x(t_{k+1}) = \begin{bmatrix} x'(t_{k+1}) \\ x''(t_{k+1}) \end{bmatrix} = \begin{bmatrix} I_N - K(t_k) & \delta(t_k)D \\ -\alpha K(t_k) & I_N \end{bmatrix} x(t_k) \tag{2.7}$$

dove $\delta(t_k) = t_{k+1} - t_k$ è l'intervallo di tempo tra due istanti di aggiornamento[2] e la matrice $D \in \mathbb{R}^{N \times N}$ è definita come

$$D = diag(f_1, \cdots, f_N) \tag{2.8}$$

dove f_i con $i = 1, \cdots, N$ sono le frequenze interne dei clocks considerate costanti e pari a $f_i \in [1-\epsilon, 1+\epsilon]$ con $0 \leq \epsilon < 1$.

Come già anticipato ci si aspetterà che il clock interno di tutti i nodi presenti sulla rete, evolverà seguendo l'aggiornamento descritto dall'equazione (2.7), dunque l'obiettivo di questo lavoro consiste nel progettare un algoritmo atto all'autotuning del parametro α relativo alla parte integrale del controllore.

Per una trattazione più semplice, soprattutto per la parte simulativa del progetto, viene utilizzata la legge di aggiornamento

$$x(t+1) = \begin{bmatrix} x'(t+1) \\ x''(t+1) \end{bmatrix} = \begin{bmatrix} I_N - K(t) & \delta(t)D \\ -\alpha K(t) & I_N \end{bmatrix} x(t) \tag{2.9}$$

[1] peso dell'aggiornamento, introdotto nel capitolo 1.
[2] visto anche come il tempo di "arrivo" del processo di Poisson di intensità $N\lambda$.

2.1 Analisi in media quadratica e convergenza dell'algoritmo

Definito il modello del sistema ed il relativo controllo, è utile un'analisi del tipo teorico a fronte delle simulazione implementate in Matlab. Presa in considerazione la struttura di un grafo completo, l'obbiettivo del paragrafo risiede nello studio del comportamento della matrice di covarianza dell'errore di sincronizzazione, dipendente dal parametro α.
Per semplificare la trattazione si preferisce considerare l'utilizzo di un protocollo di comunicazione del tipo gossip asimmetrico. In tal modo il sistema evolve secondo la seguente legge:

$$x(t+1) = \begin{bmatrix} x'(t+1) \\ x''(t+1) \end{bmatrix} = \begin{bmatrix} I_N - \frac{1}{2}E_{i\to j} & \delta(t)D \\ -\frac{\alpha}{2}E_{i\to j} & I_N \end{bmatrix} x(t) \tag{2.10}$$

simile alla (2.9) vista in precedenza, ma dove $E_{i\to j} = e_j(e_i - e_j)^T$ definisce una comunicazione tra il nodo i e il nodo j.

Denominata P_i la matrice di consenso del grafo in questione, si può definire la matrice di comunicazione E_i come:

$$E_i = W_i - I = \sum_{j\in\mathcal{N}_i} E_{i\to j} \tag{2.11}$$

dove la matrice W_i è tale da essere composta da tutti zeri esclusa la colonna i-esima composta da uni. Quindi la matrice P_i può essere definita come:

$$P_i = I + (1-\beta)E_i \tag{2.12}$$

Volendo evidenziale la matrice di retroazione, si ha:

$$K_i = I - P_i = -(1-\beta)E_i = -(1-\beta)(W_i - I) \tag{2.13}$$

Assumiamo, come fatto in precedenza, $\mu = \frac{1}{2}$ al fine di aver un aggiornamento in media degli stati e in tal modo l'equazione diventa:

$$K_i = -\frac{1}{2}(W_i - I) \tag{2.14}$$

Prima di avanzare con l'analisi è utile evidenziare alcune proprietà necessarie per semplificare le formulazioni:

- ad ogni k-passo la matrice $E(k)$ è scelta nell'insieme $\mathcal{S} = E_i = \sum_{j\in\mathcal{N}_i} E_{i\to j}$, quindi:

$$P[E(k) = E_i] = \frac{1}{N} \tag{2.15}$$

- il valore atteso della matrice di comunicazione sarà:

$$E[E[k]] = \frac{1}{N}\sum_{i=1}^{N} E_i = \frac{1}{N}\left(\sum_{i=1}^{N}(W_i - I)\right) \tag{2.16}$$

$$= \frac{1}{N}\left(\sum_{i=1}^{N} W_i - \sum_{i=1}^{N} I\right) = \frac{1}{N}\left(11^* - NI\right) = -\Omega \tag{2.17}$$

- proprietà di Ω:

$$\Omega = \Omega^* \tag{2.18}$$

$$\Omega\Omega = \Omega \tag{2.19}$$

$$E(k)\Omega = E(k) \tag{2.20}$$

$$\Omega E_i = -\Omega \tag{2.21}$$

- gli autovalori della matrice di aggiornamento di stato (2.10) del sistema variano a seconda della variazione della matrice D rispetto all'identità: in seguito si considererà $D = I$, per semplificare la trattazione.
- i momenti di primo e secondo ordine degli intervalli di aggiornamento δ_k del sistema valgono rispettivamente:

$$E[\delta_k] = \frac{1}{N\lambda} \tag{2.22}$$

$$E[\delta_k^2] = \frac{2}{N^2\lambda^2} \tag{2.23}$$

  calcolati in base alle proprietà della variabile aleatoria esponenziale di parametro λ.

Si definiscono

$$y(k) = \Omega x'(k) = [I - \frac{1}{N}11^*]x'(k) \tag{2.24}$$

l'errore di sincronizzazione e la nuova variabile

$$z(k) = \Omega D x''(k) = [I - \frac{1}{N}11^*]Dx''(k) \tag{2.25}$$

Si può quindi riscrivere la (2.10) come:

$$\begin{bmatrix} y(k+1) \\ z(k+1) \end{bmatrix} = \begin{bmatrix} \Omega & 0 \\ 0 & \Omega D \end{bmatrix} \begin{bmatrix} I + \frac{1}{2}E(k) & \delta_k D \\ \frac{\alpha}{2}E(k)I & I \end{bmatrix} \begin{bmatrix} x'(t_k) \\ x''(t_k) \end{bmatrix} \tag{2.26}$$

$$= \begin{bmatrix} \frac{1}{2}I & \delta_k I \\ \frac{\alpha}{2}\Omega D E(k) & I \end{bmatrix} \begin{bmatrix} y(k) \\ z(k) \end{bmatrix} \tag{2.27}$$

$$= \begin{bmatrix} A_{11}(k) & A_{12}(k) \\ A_{21}(k) & A_{22}(k) \end{bmatrix} \begin{bmatrix} y(k) \\ z(k) \end{bmatrix} = A(k) \begin{bmatrix} y(k) \\ z(k) \end{bmatrix} \tag{2.28}$$

Ponendo dunque $D = I$ si può analizzare la seguente matrice di covarianza:

$$\Sigma(k) = E\left[\begin{bmatrix} y(k) \\ z(k) \end{bmatrix} \begin{bmatrix} y(k)^* & z(k)^* \end{bmatrix} \right] = \begin{bmatrix} E[y(k)y(k)^*] & E[y(k)z(k)^*] \\ E[z(k)y(k)^*] & E[z(k)z(k)^*] \end{bmatrix} \tag{2.29}$$

$$= \begin{bmatrix} \Sigma_{yy}(k) & \Sigma_{yz}(k) \\ \Sigma_{zy}(k) & \Sigma_{zz}(k) \end{bmatrix} = \begin{bmatrix} \Sigma_{yy}(k) & \Sigma_{yz}(k) \\ \Sigma_{yz}(k)^* & \Sigma_{zz}(k) \end{bmatrix} \tag{2.30}$$

la cui evoluzione temporale seguirà la legge di aggiornamento

$$\Sigma(k+1) = E[A(k)\Sigma(k)A(k)^*] = F[\Sigma(k)] \tag{2.31}$$

Questa risulta una trasformazione lineare che si aggiorna mediante un algoritmo ricorsivo inizializzato dai valori $x'(k)$ e $x''(k)$. Noto il modello reale del clock è lecito definire $x'(0)$ come un vettore casuale di media nulla ($E[x'(0)] = 0$) e varianza δ_x^2 ($E[x'(0)x'(0)^*] = 0$), mentre $x''(0)$ come un vettore di tutti uni, quindi:

$$\Sigma(0) = \begin{bmatrix} \delta_x^2\Omega & 0 \\ 0 & 0 \end{bmatrix} \tag{2.32}$$

Le funzioni sopra descritte ora possono esser esplicate in funzione della matrice Ω come:

$$\Sigma_{yy}(k+1) = f_{yy}(k)\Omega \tag{2.33}$$

$$\Sigma_{yz}(k+1) = f_{yz}(k)\Omega \tag{2.34}$$

$$\Sigma_{zz}(k+1) = f_{zz}(k)\Omega \tag{2.35}$$

dove

$$f(k+1) = \begin{bmatrix} f_{yy}(k+1) \\ f_{yz}(k+1) \\ f_{zz}(k+1) \end{bmatrix} = \begin{bmatrix} \mu^2 & \frac{2\mu}{N\lambda} & \frac{2}{N^2\lambda^2} \\ -\alpha\mu(1-\mu) & \rho & \frac{1}{N\lambda} \\ \alpha^2(1-\mu)^2 & -2\alpha(1-\mu) & 1 \end{bmatrix} \begin{bmatrix} f_{yy}(k) \\ f_{yz}(k) \\ f_{zz}(k) \end{bmatrix} \tag{2.36}$$

$$= \begin{bmatrix} \frac{1}{4} & \frac{1}{N\lambda} & \frac{2}{N^2\lambda^2} \\ -\frac{\alpha}{4} & \frac{N\lambda-\alpha}{2N\lambda} & \frac{1}{N\lambda} \\ \frac{\alpha^2}{4} & -\alpha & 1 \end{bmatrix} \begin{bmatrix} f_{yy}(k) \\ f_{yz}(k) \\ f_{zz}(k) \end{bmatrix} = \Gamma(\alpha, N, \lambda) \tag{2.37}$$

dove $\rho = \mu - \frac{\alpha(1-\mu)}{N\lambda}$ e $\mu = \frac{1}{2}$, ottenute poichè applicando le condizioni iniziali in questo grafo completo si ha che l'insieme

$$J = \left\{ \Sigma : \Sigma = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \otimes \Omega \right\} \tag{2.38}$$

è invariante rispetto la trasformazione lineare di aggiornamento della matrice di covarianza.

Dalla matrice in (2.37) è possibile ottenere il polinomio caratteristico per studiarne l'andamento asintotico in funzione dei parametri α,N e λ:

$$\psi(z) = det(zI - \Gamma(\alpha, N, \lambda)) \tag{2.39}$$

$$= z^3 + \left(\frac{7}{4} + \frac{1}{2}\frac{\alpha}{N\lambda}\right) z^2 + \left(-\frac{1}{2}\left(\frac{\alpha}{N\lambda}\right)^2 + \frac{5}{8}\frac{\alpha}{N\lambda} + \frac{7}{8}\right) z + \tag{2.40}$$

$$+ \left(-\frac{1}{4}\left(\frac{\alpha}{N\lambda}\right)^3 - \frac{1}{2}\left(\frac{\alpha}{N\lambda}\right)^2 - \frac{3}{8}\frac{\alpha}{N\lambda} - \frac{1}{8}\right) = q_3 z^3 + q_2 z^2 + q_1 z + q_0 \tag{2.41}$$

Per verificarne la stabilità è possibile applicare il criterio di Routh nella forma a tempo continuo ottenuta applicando la trasformazione di Mobius $z = (1+s)/(1-s)$. Si ottiene dunque:

$$\psi(z)(s) = q_3' s^3 + q_2' s^2 + q_1' s + q_0' \tag{2.42}$$

$$= \left(\frac{1}{4}\left(\frac{\alpha}{N\lambda}\right)^3 + \frac{1}{2}\frac{\alpha}{N\lambda} + \frac{15}{4}\right) s^3 + \left(-\frac{3}{4}\left(\frac{\alpha}{N\lambda}\right)^3 - \left(\frac{\alpha}{N\lambda}\right)^2 - \frac{9}{4}\frac{\alpha}{N\lambda} + \frac{7}{2}\right) s^2 + \tag{2.43}$$

$$+ \left(\frac{3}{4}\left(\frac{\alpha}{N\lambda}\right)^3 + 2\left(\frac{\alpha}{N\lambda}\right)^2 + \frac{\alpha}{N\lambda} + \frac{3}{4}\right) s + \left(-\frac{1}{4}\left(\frac{\alpha}{N\lambda}\right)^3 - 1\left(\frac{\alpha}{N\lambda}\right)^2 + \frac{3}{8}\frac{\alpha}{N\lambda}\right) \tag{2.44}$$

Per ottenere la stabilità è necessario avere tutti i coiffecienti q_i' dello stesso segno. I parametri $\frac{\alpha}{N\lambda}$ sono positivi per le ipotesi iniziali fissate, di conseguenza q_3' e q_1' sono strettamente positivi, quindi è utile ricondurci solamente allo studio dei parametri q_2' e q_0', evidenziando il valore massimo di α accettabile in funzione del numero di nodi nella rete e del parametro della variabile aleatoria esponenziale.
Viene analizzata la rete nel caso $\lambda = 0.05$ differenziando i risultati ponendo $N = 2, 3, 4, 5, 10, 20, 100$ e successivamente estendendo al caso $\lambda = 0.15$ e $\lambda = 0.30$.
Dalle limitazioni ottenute sui parametri, si ottiene la seguente tabella in cui vengono tabulati i valori massimi di α ammissibili:

N	$\lambda = 0.05$	$\lambda = 0.15$	$\lambda = 0.30$
2	0.0346	0.1039	0.2077
3	0.0519	0.1558	0.3116
4	0.0692	0.2077	0.4155
5	0.0887	0.2957	0.5194
10	0.1731	0.5193	1.0000
20	0.3462	1.0000	1.0000
100	1.0000	1.0000	1.0000

Tabella 2.1: Valori del parametro α_{MAX} nei casi $N = 2, 3, 4, 5, 10, 20, 100$ e $\lambda = 0.05, 0.15, 0.30$.

L'analisi effettuata mette in luce il fatto che il parametro q_0' obblighi a restrizioni maggiori rispetto al parametro q_2' per quanto riguarda il valore massimo di α.
Come già detto i coefficienti q_1' eq_3' non vengono presi in considerazione poichè sempre positivi, mentre i valori di α_{MAX} calcolati per N=5 e N=20 saranno in seguito utili per effettuare le simulazioni.
I seguenti grafici mettono in evidenza che all'aumentare di λ è possibile incrementare il parametroα relativo alla parte integrativa del controllore, mantenendo in ogni caso la stabilità asintotica del sistema.

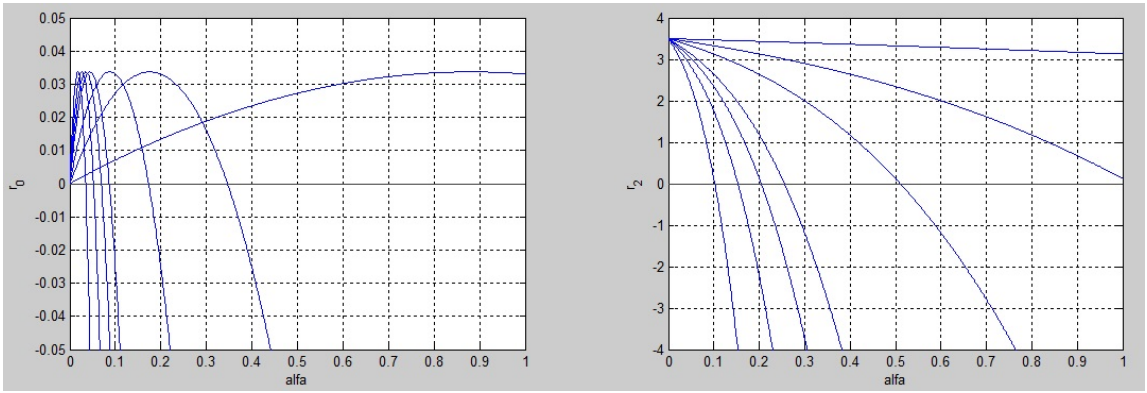


Figura 2.1: Valori dei coefficienti q_0' e q_2' con $\lambda = 0.05$ al variare di α nelle configurazioni con $N = 2, 3, 4, 5, 10, 20, 100$ nodi.

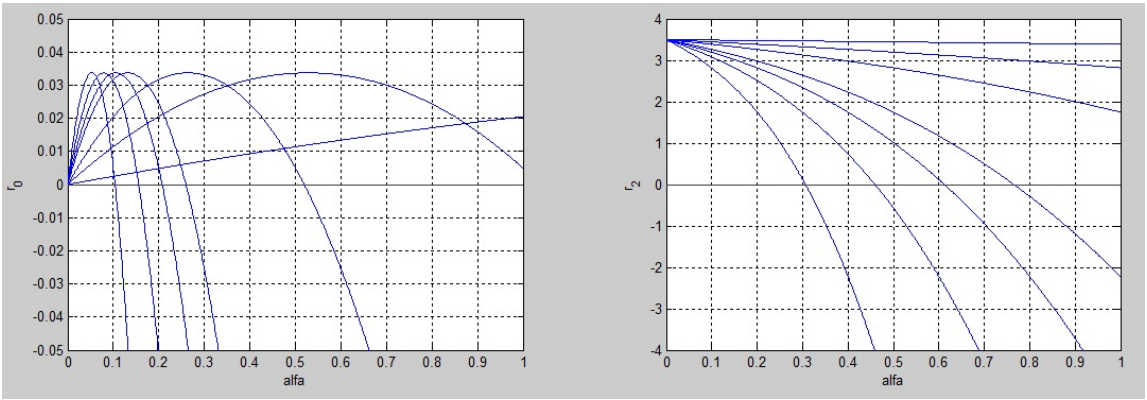


Figura 2.2: Valori dei coefficienti q_0' e q_2' con $\lambda = 0.15$ al variare di α nelle configurazioni con $N = 2, 3, 4, 5, 10, 20, 100$ nodi.

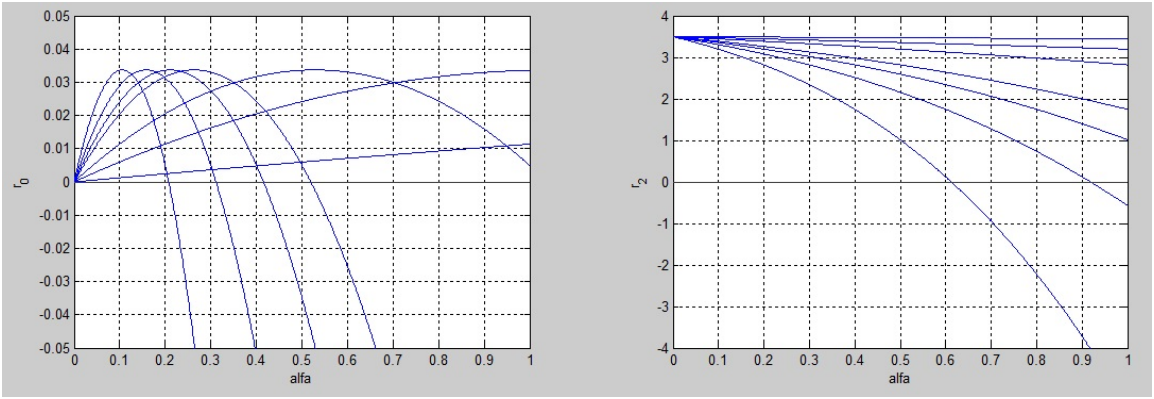


Figura 2.3: Valori dei coefficienti q_0' e q_2' con $\lambda = 0.3$ al variare di α nelle configurazioni con $N = 2, 3, 4, 5, 10, 20, 100$ nodi.

2.2 Simulazioni con α costante

Per effettuare le simulazione dell'algoritmo appena discuso viene usato il software *Matlab*. Fissato un insieme di N nodi che possono comunicare gli uni con gli altri, viene ad esso associata una matrice di comunicazione M, simmetrica[3], composta da uni nelle posizioni $[M]_{ij}$, che denotano la possibilità di comunicazione tra i nodi i e j, e da zeri quando i 2 nodi non sono comunicanti. Quindi attraverso questa matrice si riesce ad avere una corrispondenza biunivoca tra reti di nodi e matrici di comunicazione. Grazie ad una funzione appositamente scritta è anche possibile visualizzare graficamente la rete di nodi a partire dalla matrice di comunicazione M scelta.
Per esempio, data la matrice di comunicazione $M = 1^T 1$ di dimensione 6, si ha la seguente rete di comunicazione (grafo completo) di Fig. 2.4.

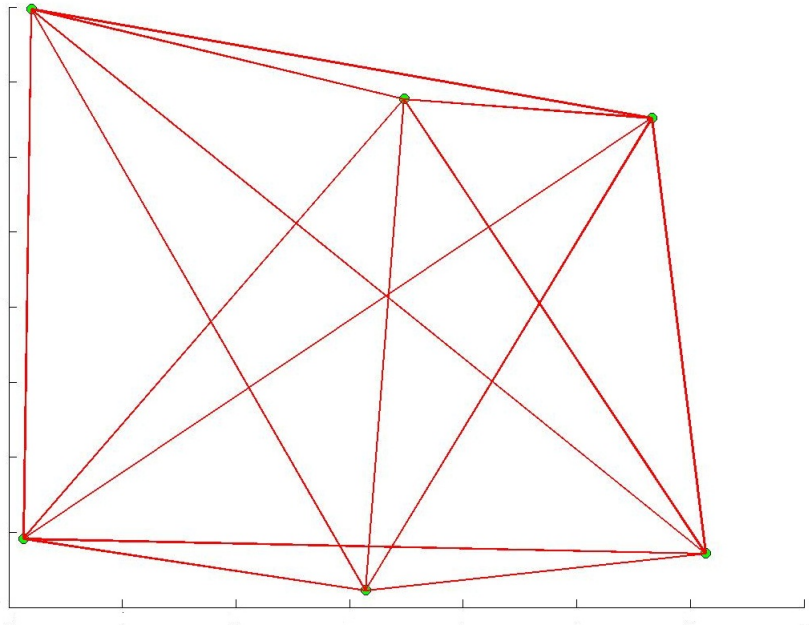


Figura 2.4: Esempio di rete a 6 nodi

Questa funzione risulta molto comoda per capire su che tipo di rete si sta eseguendo la simulazione, infatti la sua visualizzazione grafica rende più intuitiva la struttura e quindi le comunicazione tra i nodi di un grafo.

Una volta definita la rete di nodi sulla quale si vuole eseguire le simulazioni, sono state impostate le condizioni iniziali: per ogni singolo nodo è stata scelta in maniera casuale la sua frequenza "reale" del clock interno f_i, tale per cui $f_i \in [1-\epsilon\ 1+\epsilon]\ Hz$ con $\epsilon \in [0,1]\,Hz$, mentre la frequenza nominale è stata posta a $f_{i,nom} = 1\ Hz$ (quindi i valori iniziali delle stime dei periodi reali sono pari a $\triangle_i(0) = \frac{1}{f_{i,nome}} = \frac{1}{f_i}$); inoltre viene scelto casualmente il valore iniziale (offset) dell'orologio interno $x'(0)$ per ogni nodo. Questi valori appartengono uniformemente all'intervallo $[0, c]$ secondi e $c \in [0, 200]$ secondi, in modo tale da riuscire a simulare orologi che partono da offset molto differenti.

Una volta imposte le condizioni iniziali ai nodi non è stato scelto di applicare subito il controllo, ma di lasciarli "evolvere liberamente" per i primi secondi della simulazione per poi procedere applicando il controllo.

Purtroppo simulando con il software *Matlab* non è possibile applicare l'algoritmo indipendentemente ad ogni nodo quindi è necessario usare degli artifizi per inserire le condizioni presentate precedentemente: ogni nodo si accende (e comunica) seguendo una variabile aleatoria che descrive un processo di Poisson di parametro $\lambda = 1$. Per far ciò viene usata una base dei tempi comune a tutti i nodi

[3]Per avere archi bidirezionali nel grafo.

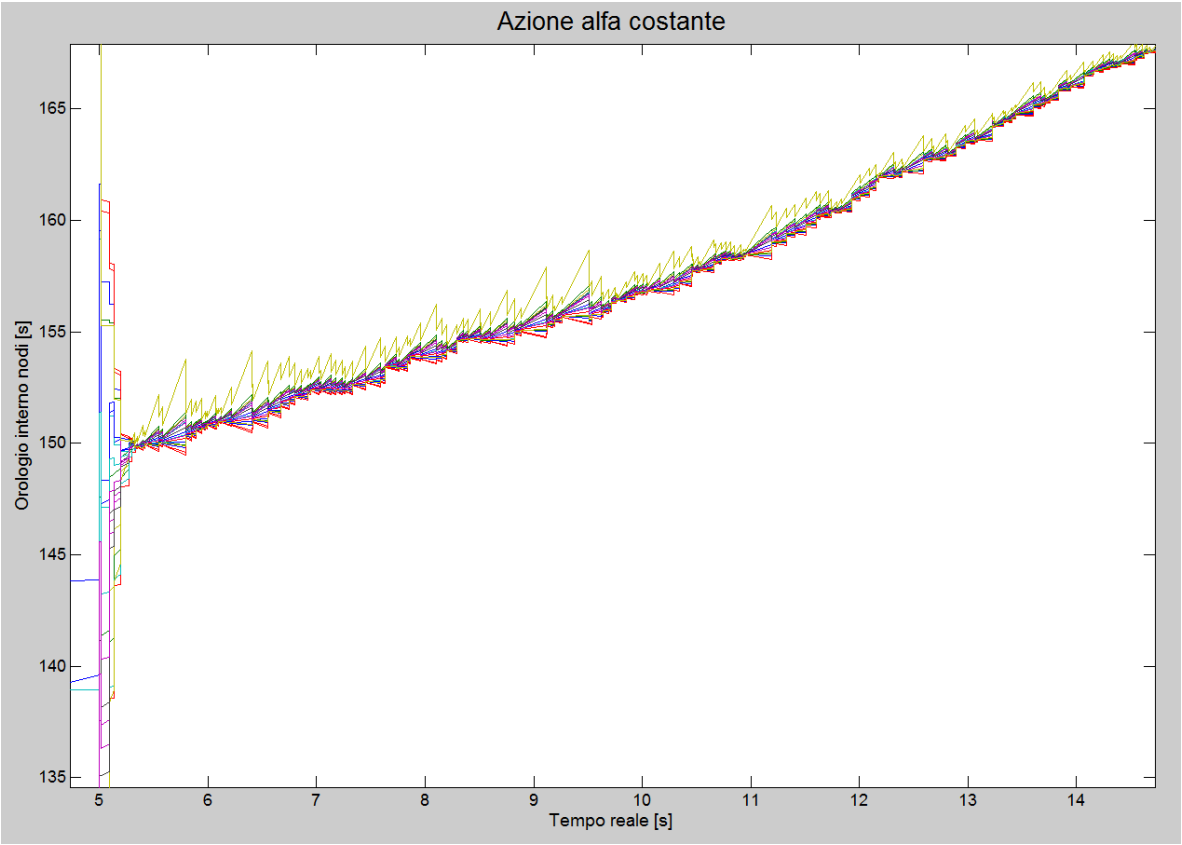
che stabilisce gli istanti di accensione di quest'ultimi secondo una variabile di Poisson[4] di parametro $\lambda = N$. Viene quindi simulata una sequenza temporale che segue questa distribuzione per individuare gli istanti di accensione dei nodi. L'algoritmo del controllo PI viene quindi fatto funzionare per ogni nodo, ciclicamente (con un ciclo while), per tutti gli istanti di accensione dei nodi.

2.2.1 Simulazioni Matlab, osservazioni e considerazioni

Le simulazioni, mantentendo fisso $\lambda = 1$, hanno risultati particolarmente variegati a seconda dei seguenti parametri iniziali scelti:

- α, "guadagno" del controllo integrale;
- ϵ, scostamento massimo delle frequenze degli orologi rispetto alla frequenza nominale;
- c, valore massimo di offset iniziale;
- tipo di grafo di comunicazione tra nodi[5].

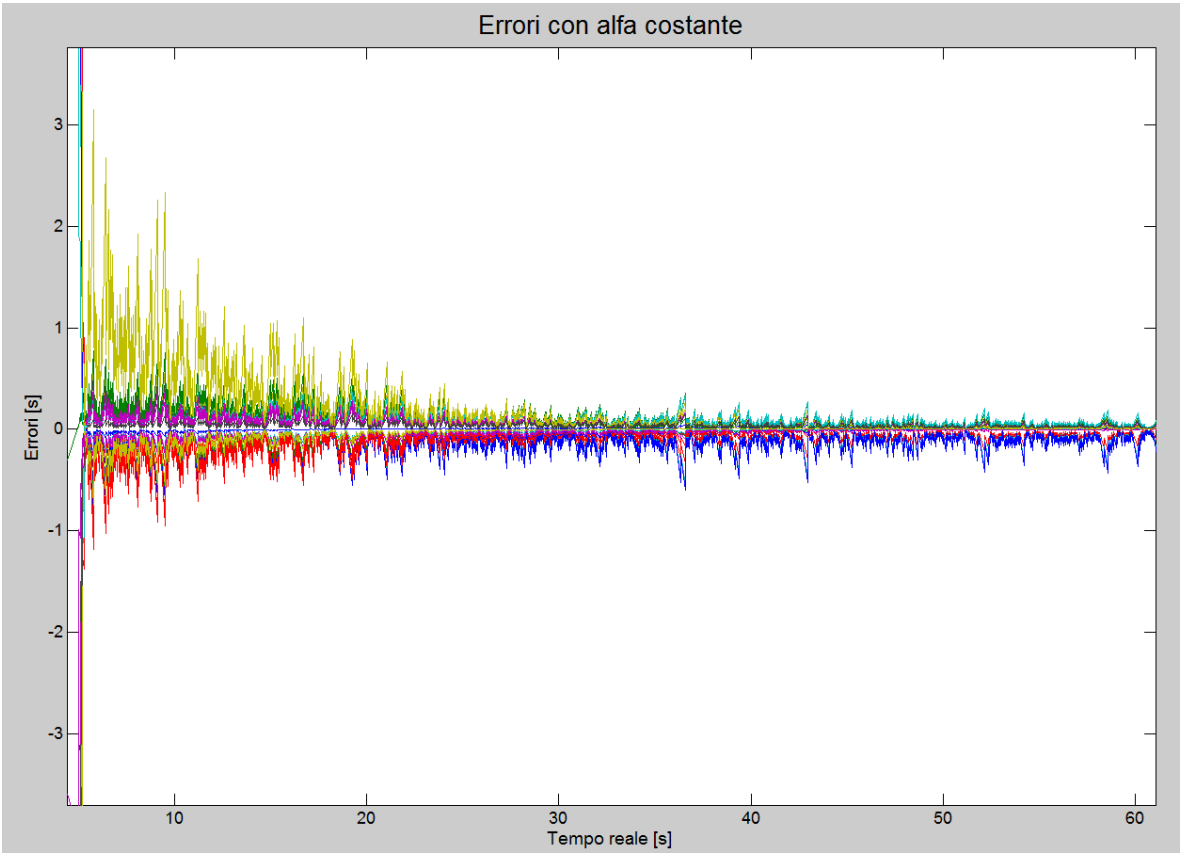
Come è lecito aspettarsi, più i parametri ϵ e c aumentano, più bisogna diminuire il guadagno α per riuscire ad avere stabilità ed un buon comportamento dinamico. Viceversa se ϵ e c sono piccoli si può aumentare il guadagno α.
Ponendo per esempio $\epsilon = 1\,Hz$ e $c = 200\,secondi$, in una rete di 20 nodi, è risultato evidente imporre il guadagno massimo pari a $\alpha = 0.06$ per mantenere stabilità ed evitare uno skew negativo, come rappresentato in Figura 2.5(a).



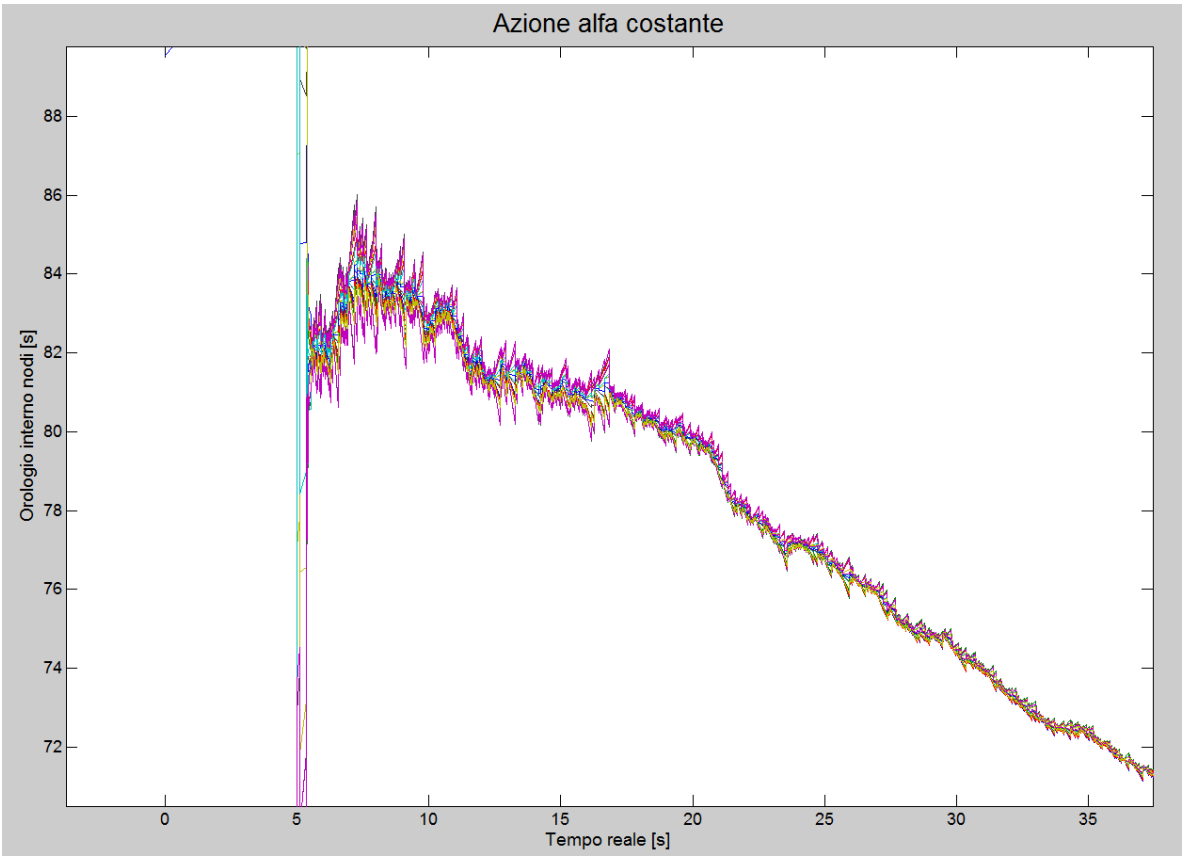
(a)

[4]Considerando gli istanti di accensione di ogni nodo come processi di Poisson i.i.d. con parametro $\lambda = 1$, sulla base dei tempi dei tempi comune l'accensione di un nodo seguirà una variabile di Poisson di parametro $\lambda = N$.

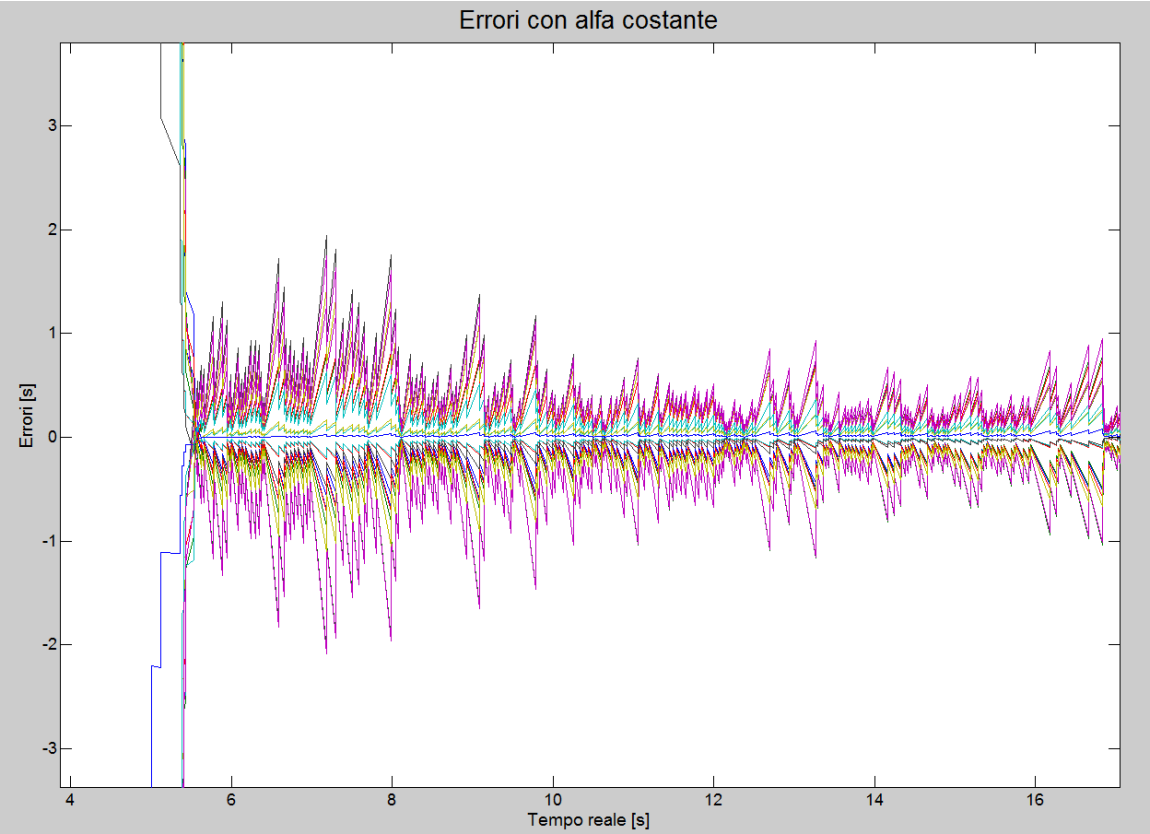
[5]Per tutte le simulazioni che seguiranno, si assumerà in realtà di avere un grafo completo. Con grafi non completi si nota semplicemente una maggiore propensione all'instabilità ed un maggior tempo di convergenza. L'andamento tipico degli errori risulta invece simile in entrambe le tipologie di grafo.


Errori con alfa costante
Errori [s]
Tempo reale [s]
3
2
1
0
-1
-2
-3
10
20
30
40
50
60

(a)


Azione alfa costante
Orologio interno nodi [s]
Tempo reale [s]
88
86
84
82
80
78
76
74
72
0
5
10
15
20
25
30
35

(b)



(b)

Figura 2.5: Simulazione con $\epsilon = 1\,Hz$, $c = 200\,secondi$ e $N = 20$ con (a) $\alpha = 0.06$ e (b) $\alpha = 0.1$

Si può notare come si riesce ad avere sincronismo, ma nella Fig. 2.6 (particolare della Fig. 2.5(a)) si vede come un basso valore di α faccia si che gli errori abbiano un andamento simile a quello di un semplice controllo proporzionale. In questo caso, per avere stabilità, bisogna quindi quasi totalmente annullare il controllo integrativo. Notiamo invece che ponendo un valore di α leggermente superiore, per esempio $\alpha = 0.1$, si possono ottenere i risultati di figura 2.5(b) dove è presente uno skew negativo.

Partendo invece da condizioni iniziali più realistiche, quali per esempio $\epsilon = 0.2\,Hz$ e $c = 30\,secondi$, si può portare il guadagno α fino al valore massimo di 0.3 riuscendo a mantenere sempre la stabilità.

A valle delle prime simulazioni effettuate è stato notato un primo aspetto negativo di questo tipo di controllo: la propensione all'instabilità del sistema nasce durante le fasi iniziali del controllo, quando sono presenti offset ancora consistenti e i $\triangle(t)$ sono corretti da un parametro α tendenzialmente elevato. Si osserva quindi che il controllo potrebbe subire un notevole miglioramento applicando un α inizialmente basso e successivamente maggiore. Anche a causa di queste osservazioni si proverà a modificare il controllore per migliorarne le prestazioni.

2.2.2 Aspetti computazionali

Si vuole rendere nota una modifica necessariamente apportata all'algoritmo al fine di risolvere lo spiacevole e poco realistico inconveniente che capita durante gli istanti di tempo in cui il controllo integrale restituisce un valore $\triangle(h+1) < 0$. Questo fatto coincide nella realtà con l'evoluzione negativa del tempo, fatto che non rende plausibili le simulazioni che si farebbero senza un qualche tipo di accorgimento. Per evitare il problema è stato semplicemente integrato un controllo sul parametro interessato che lo limita ad essere nullo nel caso si presentasse negativo. Ci si è resi conto che questa modifica ha inoltre reso possibile aumentare ulteriormente il parametro α senza compromettere la stabilità anche nelle condizioni più pessimistiche, arrivando ad un valore limite di 0.3.

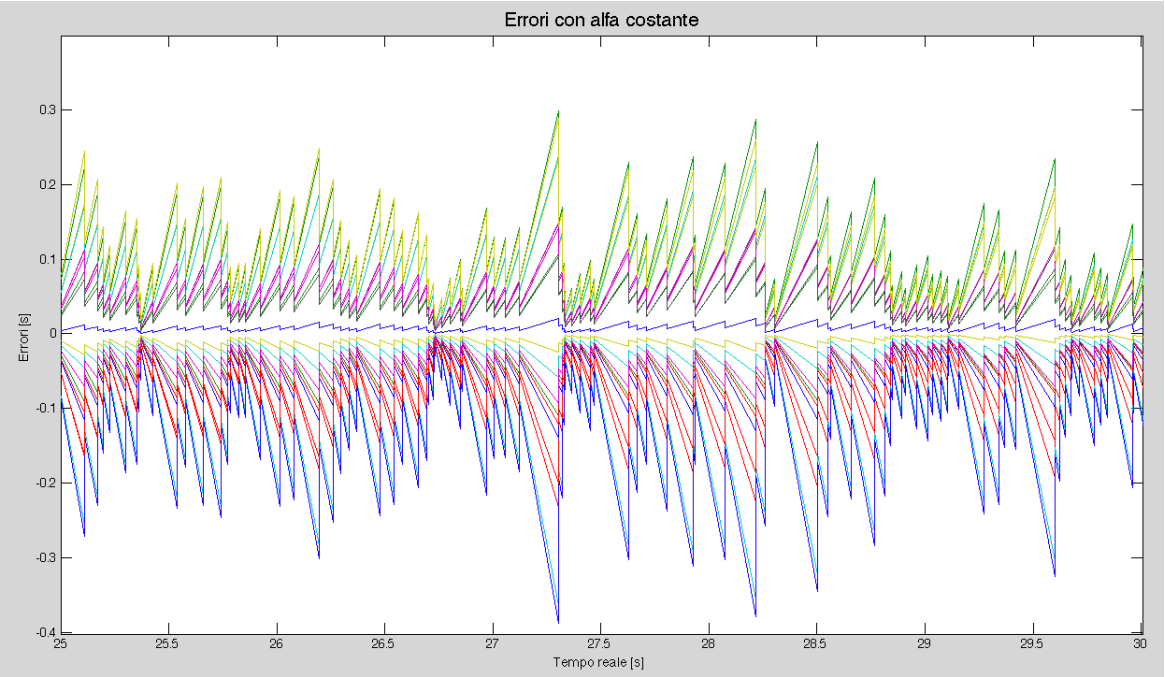


Figura 2.6: Rappresentazione in maggior dettaglio dell'andamento degli errori

Per quanto riguarda i tempi di convergenza, questi risultano relativamente abbastanza elevati. Provando per semplicità ad impostare delle condizioni di partenza ottimistiche, $\epsilon = 0.2\,Hz$ e $c = 30\,secondi$, il controllo funziona egregiamente anche per valori molto elevati di α (prossimi all'unità).
Dovendo però il controllo garantire stabilità per ogni condizione iniziale, si può capire come questo tipo di controllore sia poco robusto perchè dipende fortemente dalle condizioni iniziali. Pensando al caso peggiore bisognerebbe impostare un valore di α molto basso per assicurare convergenza in ogni occasione, ovviamente a discapito delle prestazioni. Un esempio si può vedere nella figura 2.7 dove è stata eseguita una simulazione partendo da condizioni iniziali pessimistiche ($\epsilon = 1\,Hz$ e $c = 200\,secondi$) adottando invece un valore di α ritenuto buono per delle condizioni iniziali ottimistiche ($\epsilon = 0.2\,Hz$ e $c = 30\,secondi$, $\alpha = 0.45$): non si riesce ad avere convergenza neanche dopo 70 secondi.
Nel caso in cui la particolare applicazione, sulla quale dev'essere implementato un algoritmo di sincronizzazione, richieda bassi tempi di sincronizzazione dei vari dispositivi, risulta non sufficiente scegliere solamente di innalzare il parametro α. A fronte dei limiti imposti dall'analisi in media quadratica, nel Capitolo 3 verranno analizzate nuove metodologie favorevoli ad un incremento prestazionale.

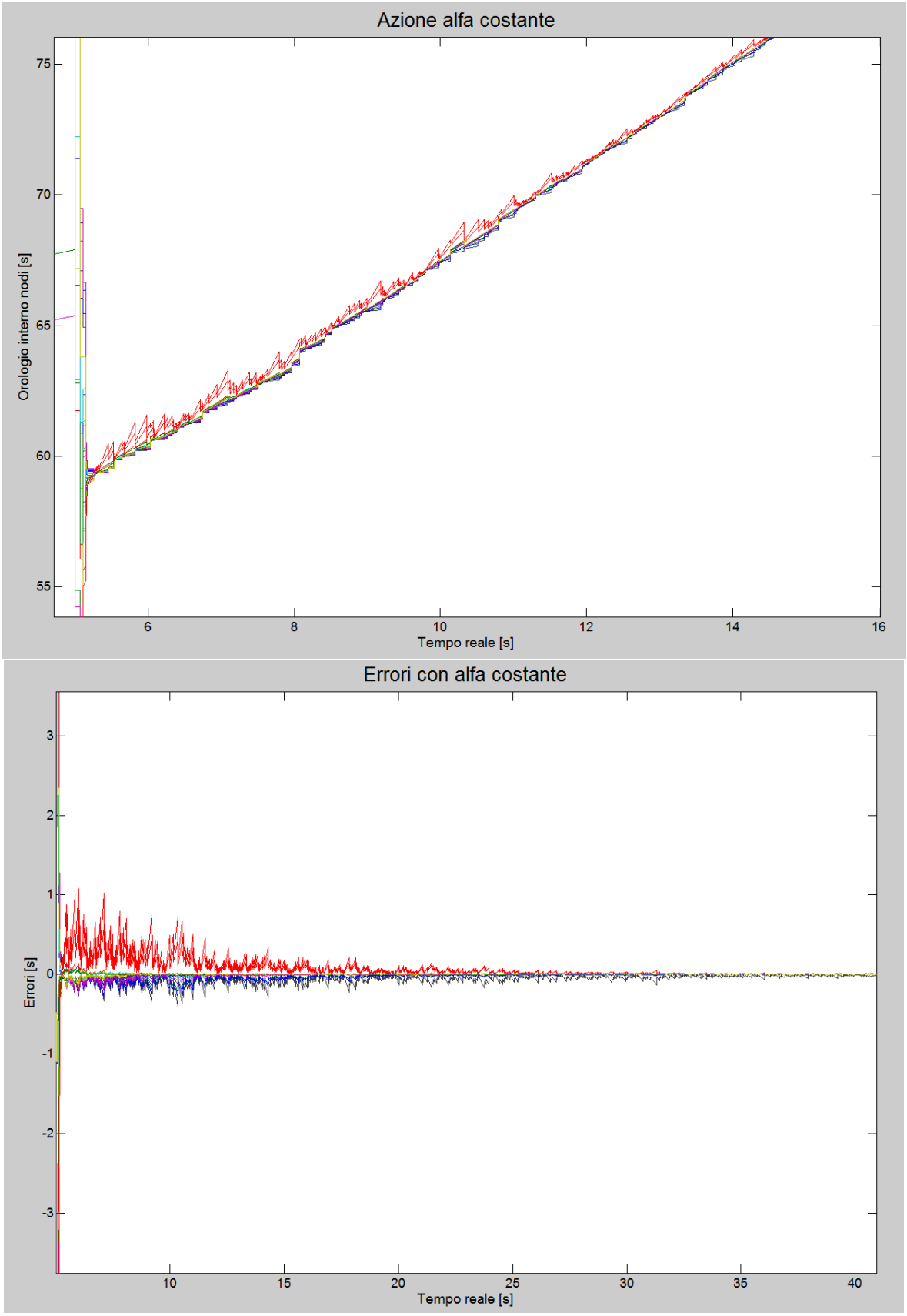


Figura 2.7: Simulazione con il bound di positività per $\triangle(h)$, $\alpha = 0.45$, $\epsilon = 1\,Hz$, $c = 200\,secondi$ ed $N = 20$

2.3 Implementazione sulla piattaforma *SensLab*

2.3.1 Introduzione a *SensLab*

Per implementare concretamente l'algoritmo di sincronizzazione con controllo PI e quindi passare dalla simulazione teorica a all'implementazione vera e propria, abbiamo sfruttato la piattaforma *SensLab*. Questa piattaforma mette a disposizione un grosso quantitativo di nodi, dotati di clock interno e con la possibilità di comunicare tra di loro (in broadcast) attraverso protocollo wireless. Data la struttura ed il protocollo disponibili è possibile dunque applicare l'algoritmo studiato finora. Questi 1024 nodi si trovano in Francia, e sono suddivisi equamente in 4 diversi laboratori: Lille, Strasbourg, Greanable e Rennes (Fig.2.8). L'accesso al server principale di ogni laboratorio si può fare direttamente da remoto collegandosi all'apposito sito e accedendovi usando le credenziali fornite ad ogni utente. Una volta connessi al server principale è possibile eseguire le operazioni preliminari per iniziare l'esperimento:

- selezionare su quali e quanti nodi (dei 256 totali) si desidera svolgere l'esperimento, avendo prima avuto l'accortezza di verificare quali, quando e per quanto tempo siano disponibili;
- selezionare la durata dell'esperimento e quando si vuole farlo partire;
- caricare per ogni nodo il programma che si vuole venga eseguito. Si può selezionare il "profilo" che si vuole far adottare al nodo[6]. Per questo progetto è stato usato un profilo standard, che attiva la comunicazione wireless e fa in modo che il nodo fosse alimentato dalla rete.

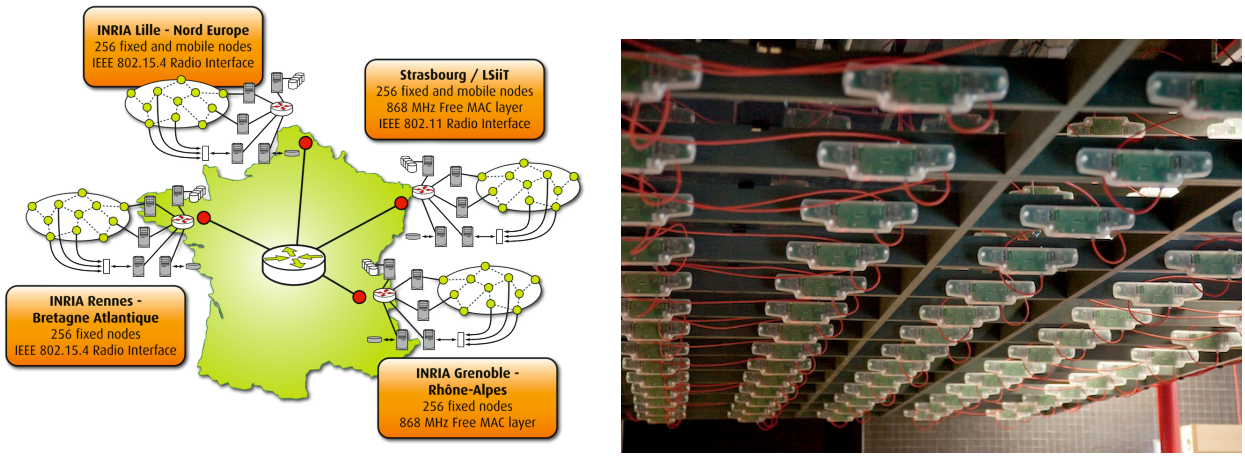


Figura 2.8: Ubicazione dei laboratori della piattaforma SenLab ed alcuni nodi.

Per le simulazioni effettuate sono stati usati alcuni dei 256 nodi ubicati nel laboratorio di Lille, con la possiblità di sfruttare l'interfaccia radio IEEE 802.15.4 presente in ogni nodo, così da riuscire a farli comunicare gli uni con gli altri (in broadcast) tramite canale wireless.

Ogni nodo dispone di un microprocessore a 16 bit con la possibilità di eseguire un programma ed ha una sua memoria interna (alquanto limitata). Il programma[7] che viene eseguito su ogni nodo è stato precedentemente scritto nel linguaggio di alto livello *C++* ed è stato poi compilato appositamente per il microprocessore di questi nodi, sfruttando il compilatore online messo a disposizione dal laboratorio; si può anche usufruire di alcune librerie fatte appositamente per i nodi (rese note tramite una guida), come per esempio quelle per acquisire il clock interno o per ricevere e inviare pacchetti tramite wireless.

Ogni singolo nodo, oltre alla comunicazione wireless, dispone di un collegamento tramite interfaccia seriale (che funge anche da alimentazione) per collegarsi direttamente con il server principale del laboratorio di Lille. Tramite questo accesso seriale è possibile, oltre a caricare i programmi in offline (quando l'esperimento non è in esecuzione), accedere ad ogni singolo nodo durante l'esecuzione dell'esperimento, tramite protocollo ssh (facilmente eseguibile da terminale unix). Questo permette di poter:

[6] per esempio se si vuole venga costantemente alimentato o se interessa provare l'alimentazione tramite batteria interna.

[7] in linea di principio può in realtà essere un programma diverso per ogni nodo.

- inviare comandi ad ogni singolo nodo durante lo svolgimento dell'esperimento, in modo da modificare l'esecuzione dell'algoritmo caricato in ogni nodo;
- ricevere (tramite interfaccia testuale del terminale) informazioni sugli orologi interni dei nodi e su come sta funzionando l'algoritmo. In questo modo è anche possibile raccogliere, a fine esperimento, una sequenza di dati utili a capire come ha funzionato l'algoritmo (anche attraverso la creazione di grafici esplicativi a partire dai dati).

2.3.2 Applicazione del controllo PI

Ogni nodo è dotato di un clock interno (ACKL), con una frequenza nominale di 32,768kHz, che fa incrementare automaticamente una variabile a 16 bit rappresentante il suo counter. Dato che $\frac{2^{16}}{f_{nominale}} \approx 2$ significa che, usando solamente questa variabile per il counter, ogni 2 secondi si avrebbe overflow e si azzererebbe, rendendo inutilizzabile questa variabile come "orologio interno". Per rimediare a ciò si utilizza una seconda variabile, sempre a 16 bit, che fungerà da MSW (Most Significant Word) per la variabile. Incrementando di 1 la variabile MSW ogniqualvolta si abbia overflow della LSW, si riesce così ad avere una variabile a 32 bit per il counter e quindi prima di avere overflow dei 32 bit bisogna aspettare circa $\frac{2^{32}}{f_{nominale}} \approx 131000$ secondi, cioè si può lasciare acceso il nodo per circa 36 ore prima di avere un overflow del counter.

Nell'esperimento, per ogni nodo è stato preso come valore dell'orologio interno proprio il valore della variabile counter (a 32bit) da quando è partito l'esperimento. Si riesce quindi ad effettuare il controllo proporzionale (offset correction) semplicemente facendo la media tra queste 2 variabili; bisogna poi però pensare a come poter variare la "stima della frequenza nominale" in modo tale che ogni orologio si aggiorni allo stesso modo (skew correction). Per far ciò viene decrementato il counter ogni 20000 tic del clock; questo preciso valore, che permette un buon livello di accuratezza, viene impostato inizialmente uguale per ogni nodo per poi poter essere modificato dall'algoritmo mediante il controllo integrativo, in modo tale da portare ogni nodo ad avere lo stesso aggiornamento temporale, cioè la stessa "pendenza" della rampa dei tempi.
Nel grafico di Fig. 2.9 si può visionare la simulazione di durata un'ora, usando 7 nodi sottoposti unicamente al controllo proporzionale.

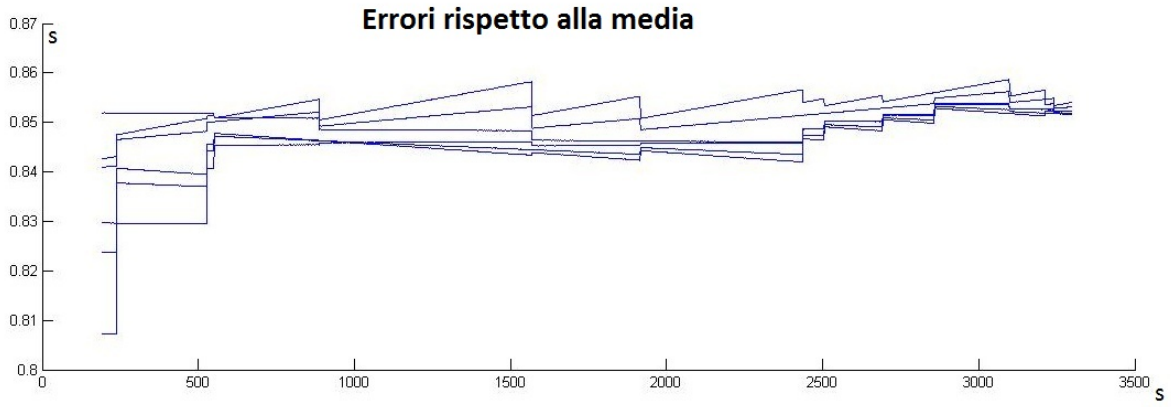


Figura 2.9: Esperimento di 1 ora con 7 nodi e solamente controllo proporzionale (offset correction).

Il grafico rappresenta gli errori, di ogni singolo nodo, rispetto ad un nodo di riferimento (che non varia il proprio counter). Come si può notare, l'andamento qualitativo rispecchia le simulazioni fatte in *Matlab*. I nodi riducono sempre più il loro offset e l'errore è limitato entro delle bande, però tendono comunque a divergere. Se si interrompesse quindi improvvisamente il controllo ogni orologio riprenderebbe con il proprio aggiornamento temporale, ognuno diverso dall'altro, e non si avrebbe sincronismo (convergenza).

È stato poi eseguito un successivo esperimento, usando 15 nodi, ma applicando il controllore PI. I risultati sono mostrati in Fig. 2.10.
Il grafico mostra, come per il precedente, gli errori rispetto ad un nodo di riferimento[8]. Si può notare come il controllore riesca effettivamente a portare a convergenza (sincronismo) gli orologi dei vari

[8]Dopo aver sottratto oppurtamente il trend lineare dovuto al diverso skew tra il nodo di riferimento e gli altri.

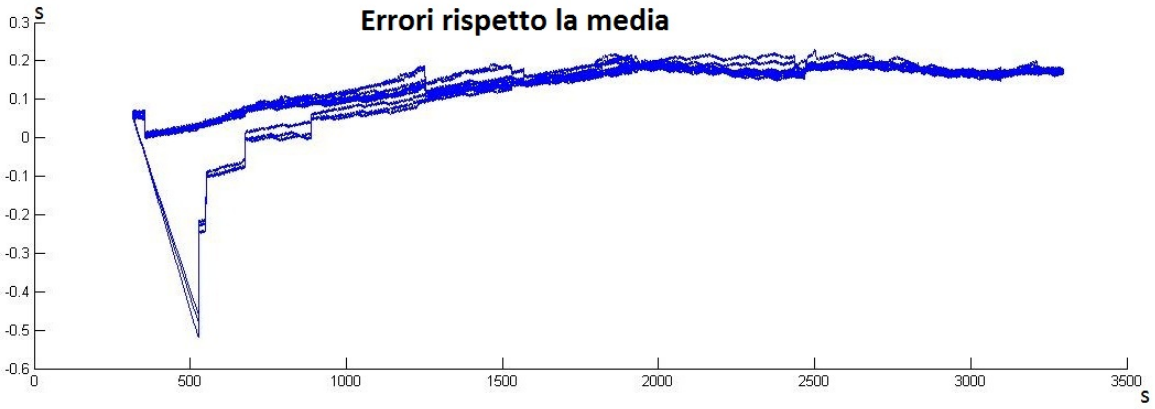


Figura 2.10: Esperimento di 1 ora con 15 nodi e controllo PI (offset and skew correction).

nodi, come ci si aspetta dalla teoria e dalle precedenti simulazioni. Si nota soprattutto che gli errori non tendono a divergere, dunque, all'arrestarsi dell'azione di controllo l'aggiornamento del sistema in evoluzione libera mantiene la sincronizzazione degli orologi. Bisogna anche sottolineare il fatto che quest'ultimo grafico presenta un andamento leggermente oscillatorio al posto di un andamento, come ci si aspetterebbe, rettilineo. Questo è causato dalla presenza di un errore di quantizzazione sul bit meno significativo, il quale oscilla tra 2 valori distinti e rappresenta il limite pratico in termine di precisione nella sincronizzazione.

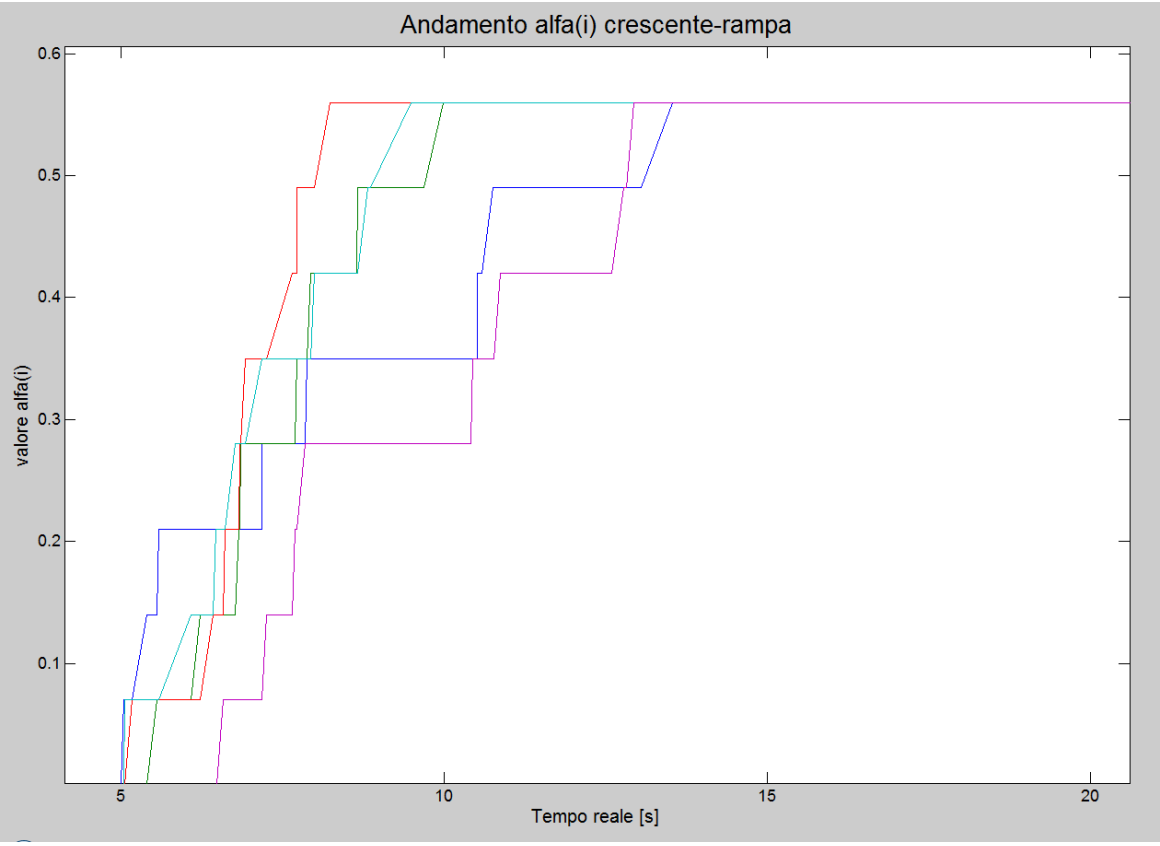
Capitolo 3

Autotuning del parametro α per controllori PI

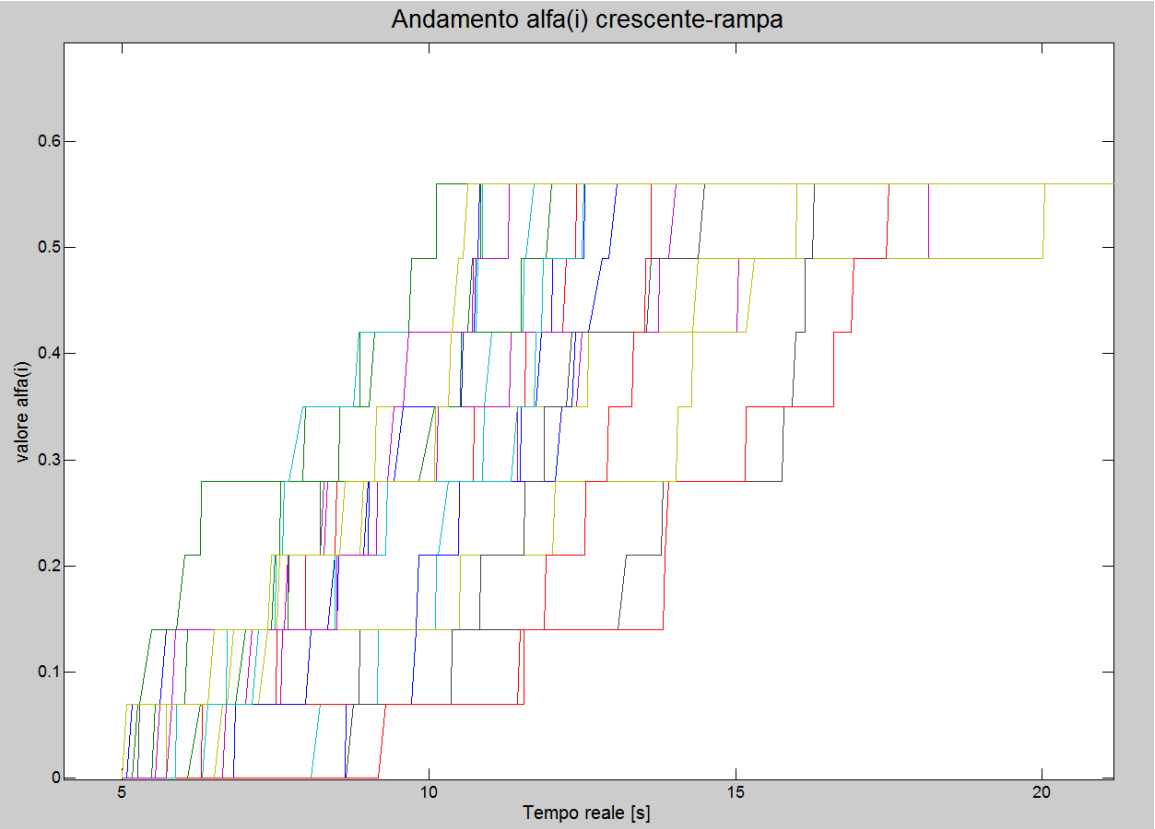
Dalle simulazioni finora effettuate è stato evidenziato come l'azione integrale sia necessaria per avere a regime un buon livello di accuratezza per il sincronismo ma risulta particolarmente "dannosa" e porta ad instabilità durante la prima fase di applicazione del controllo, dove si riscontrano valori di offset elevati. Vengono quindi proposti vari metodi per far si che, nei primi istanti di comunicazione e in caso di offset rilevante, l'azione integrativa sia limitata ed il controllore risulti totalmente proporzionale.

3.1 Crescita a rampa di α

Come prima strategia è stato deciso di far aumentare linearmente il guadagno α, partendo da un valore nullo fino ad arrivare ad un valore limite $\bar{\alpha}$. In questo caso il guadagno α risulta indipendente per ogni nodo e verrà definito come α_i, $i = 1 \dots N$.



(a)



(b)

Figura 3.1: Andamento di $\alpha_i(h)$ per una rete di 5 nodi (a) e 20 nodi (b).

Ad ogni accensione di un nodo (corrispondente all'invio di informazione sulla rete) si ottiene un aumento lineare della propria parte integrativa, espressa come $\alpha_i(h+1) = \alpha_i(h)+\rho$, fino ad arrivare al valore massimo $\bar{\alpha}$. Tutti i nodi vengono inizializzati con guadagno integrativo iniziale nullo, $\alpha\,(0) = 0$ ed un incremento ρ costante e comune per tutti i nodi.
In Fig. 3.1 si può notare un esempio dell'andamento tipico di α_i relativo ai vari nodi.

Prese a questo punto le condizioni iniziali pessimistiche ($\epsilon = 1\,Hz$, $c = 200\,secondi$) e applicando questa tipologia di controllo, ponendo $\bar{\alpha} = 0.5$ e $\rho = 0.03$, gli andamenti dei guadagni della parte integrativa vengono illustrati in Fig. 3.1(b), mentre gli andamenti generali degli orologi interni e degli errori vengono illustrati in Fig. 3.2.

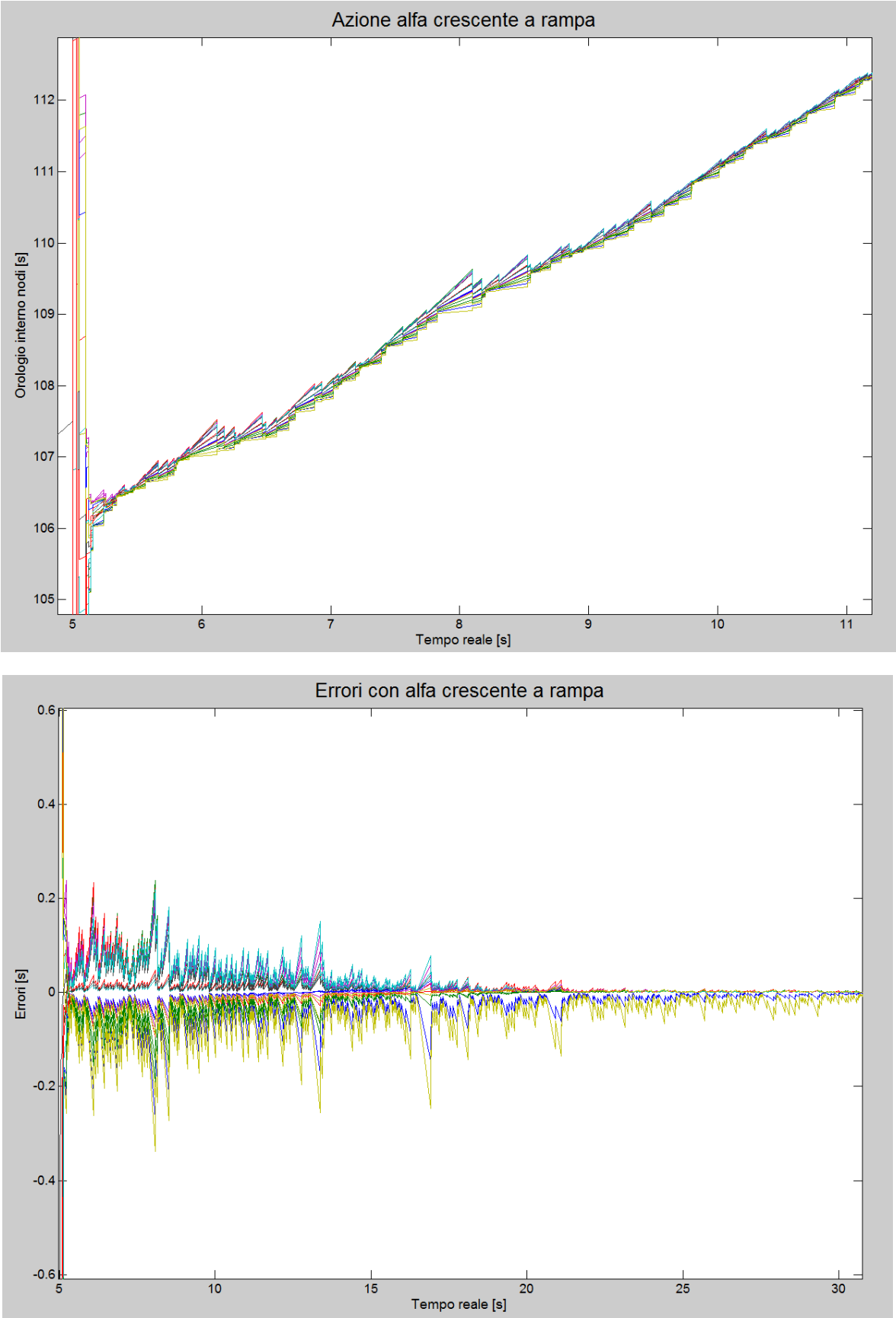


Figura 3.2: Simulazione con $\epsilon = 1\,Hz$, $c = 200\,secondi$, $\bar{\alpha} = 0.5$ e $\rho = 0.03$.

Dai diagrammi temporali si rilevano prestazioni nettamente migliori rispetto al caso con α costante trattato in precedenza, infatti gli errori convergono a zero molto più velocemente (si può anche visionare l'andamento degli errori nella Fig. 3.3 notando le condizioni iniziali in cui ci si è posti). Si riescono quindi ad ottenere migliori tempi di convergenza, migliore stabilità e miglior sincronismo a regime. Si nota anche come gli errori (dopo aver attivato il controllo) tendano a convergere e non siano tendenzialmente instabili, come invece accadeva nelle simulazioni con semplice α costante.

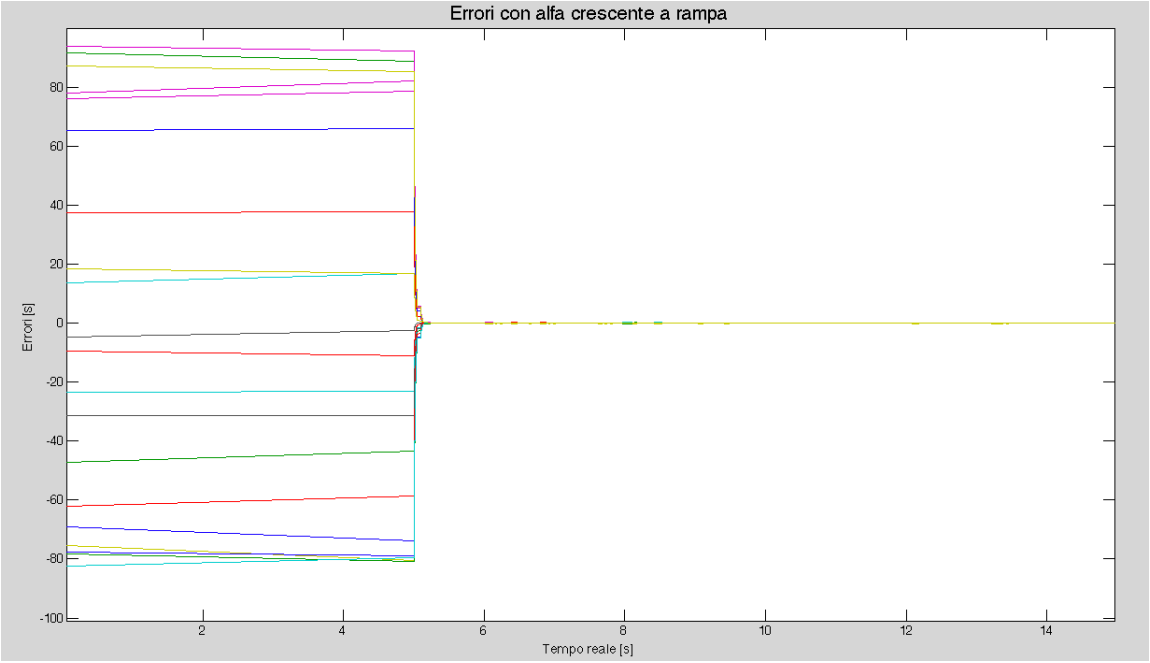


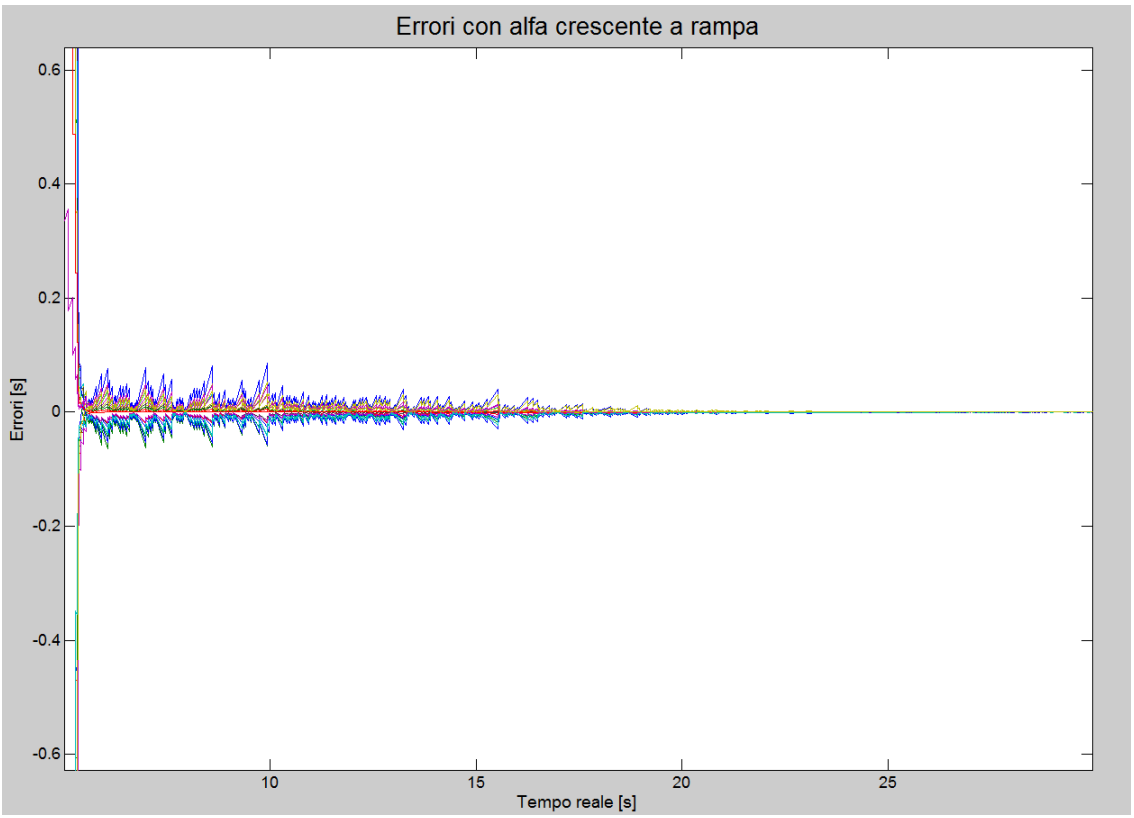
Figura 3.3: Visione estesa dell'andamento degli errori nella simulazione con $\epsilon = 1\,Hz$, $c = 200\,secondi$, $\bar{\alpha} = 0.5$ e $\rho = 0.03$.

Il problema di questo metodo è però la mancanza di robustezza in quanto i valori di crescita di α sono indipendenti dal tipo di sistema che si deve sincronizzare. Volendo sempre ottenere le prestazioni migliori in termini di tempo di convergenza, se da un lato si dovrebbe arrivare ad $\bar{\alpha}$ più velocemente poiché le condizioni iniziali presentano offset ed ϵ bassi (si veda Fig. 3.4(a)), viceversa se le condizioni iniziali fossero peggiori, è possibile che l'incremento di α sia troppo repentino (o $\bar{\alpha}$ troppo elevato) e questo porti ad instabilità o prestazioni inferiori (si veda Fig. 3.4(b)).

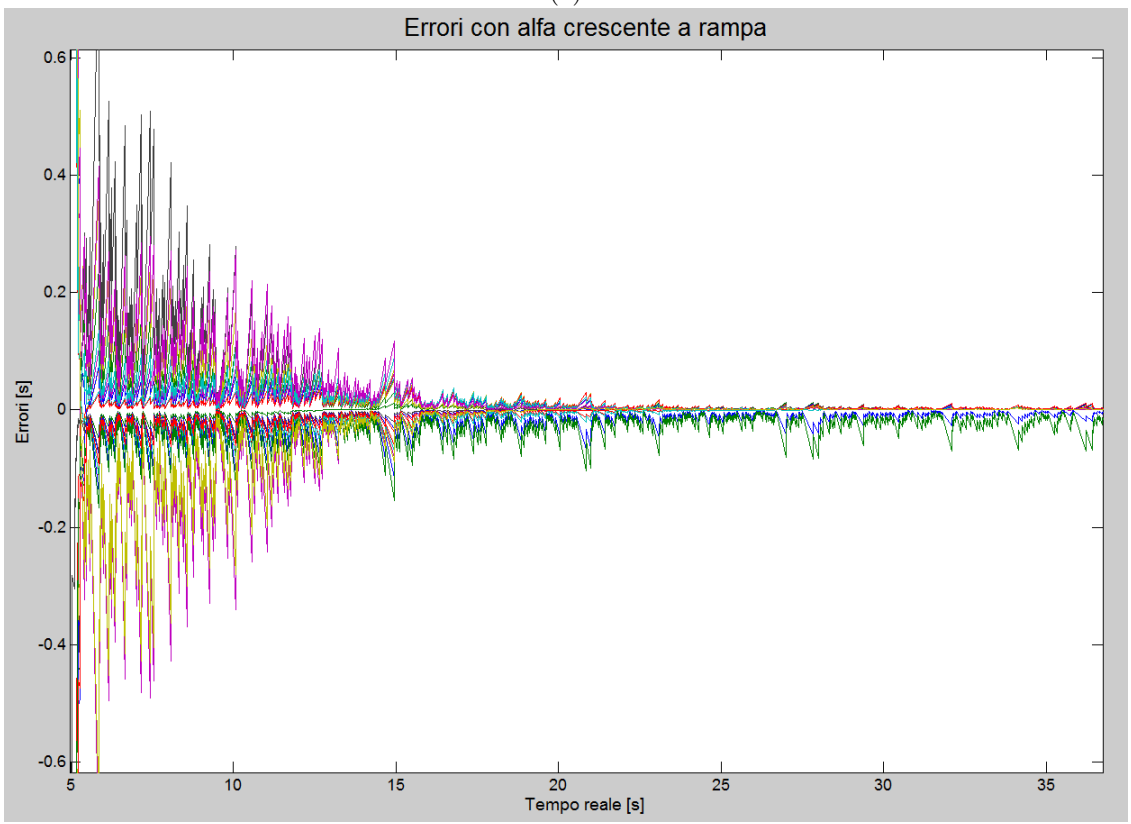
È necessario quindi trovare un metodo che in qualche modo faccia adattare il parametro α alla rete di nodi, riuscendo a determinare il giusto valore da applicare assicurando le migliori prestazioni, stabilità e robustezza.

Come ulteriori note a sfavore si rileva che con questo metodo gli andamenti dinamici dell'errore (per esempio) risultino diversi cambiando le condizioni iniziali o le condizioni della rete.
Inoltre aumentando il numero di nodi sulla rete viene sì garantita robustezza in quanto si ha un aumento del tempo di crescita del parametro α in corrispondenza di un aumento di tempo di convergenza per la rete, ma per contro non si ottiene un buon adattamento del parametro alla rete sulla quale viene applicato quest'algoritmo.

Nelle simulazioni evidenziate dalle seguenti figure, ci si è posti nelle condizioni tali per avere un'ottimizzazione delle prestazioni riferite alle condizioni pessimistiche $\epsilon = 1\,Hz$ e $c = 200\,secondi$.
Per quanto riguarda la Fig. 3.4 (a) ci si rende conto che il parametro α raggiunge il valore massimo (0.5) dopo circa 20 secondi mentre, notati gli offset iniziali posti per questa specifica simulazione (si veda la didascalia), si potrebbe raggiungere il valore massimo anche in tempo estremamente più ridotto utilizzando dei parametri ottimizzati ad hoc.
Viceversa nella Fig. 3.4 (b) sono state incrementati gli offset iniziali per rendere ancor più l'idea della poca robustezza del metodo.



(a)



(b)

Figura 3.4: Simulazioni con (a) $\epsilon = 0.3\,Hz$, $c = 50\,secondi$, $\bar{\alpha} = 0.5$ e $\rho = 0.03$ e con (b) $\epsilon = 1\,Hz$, $c = 500\,secondi$, $\bar{\alpha} = 0.5$ e $\rho = 0.03$.

3.2 Termine integrativo applicato in ritardo

Questo metodo è ancora più semplice del precedente. Si tratta di far intervenire in ritardo la parte integrativa in modo tale da avere inizialmente solo ed esclusivamente un controllo proporzionale e successivamente l'intero controllore PI. In questo modo si riesce ad evitare che l'aggiornamento di $\triangle(h)$ porti ad instabilità quando, all'inizio del controllo, si hanno valori di offset molto elevati. Superata la fase di transitorio, ottenuti quindi dei valori limitati di offset, si può inserire la parte di controllo integrativa usando un valore di α costante.

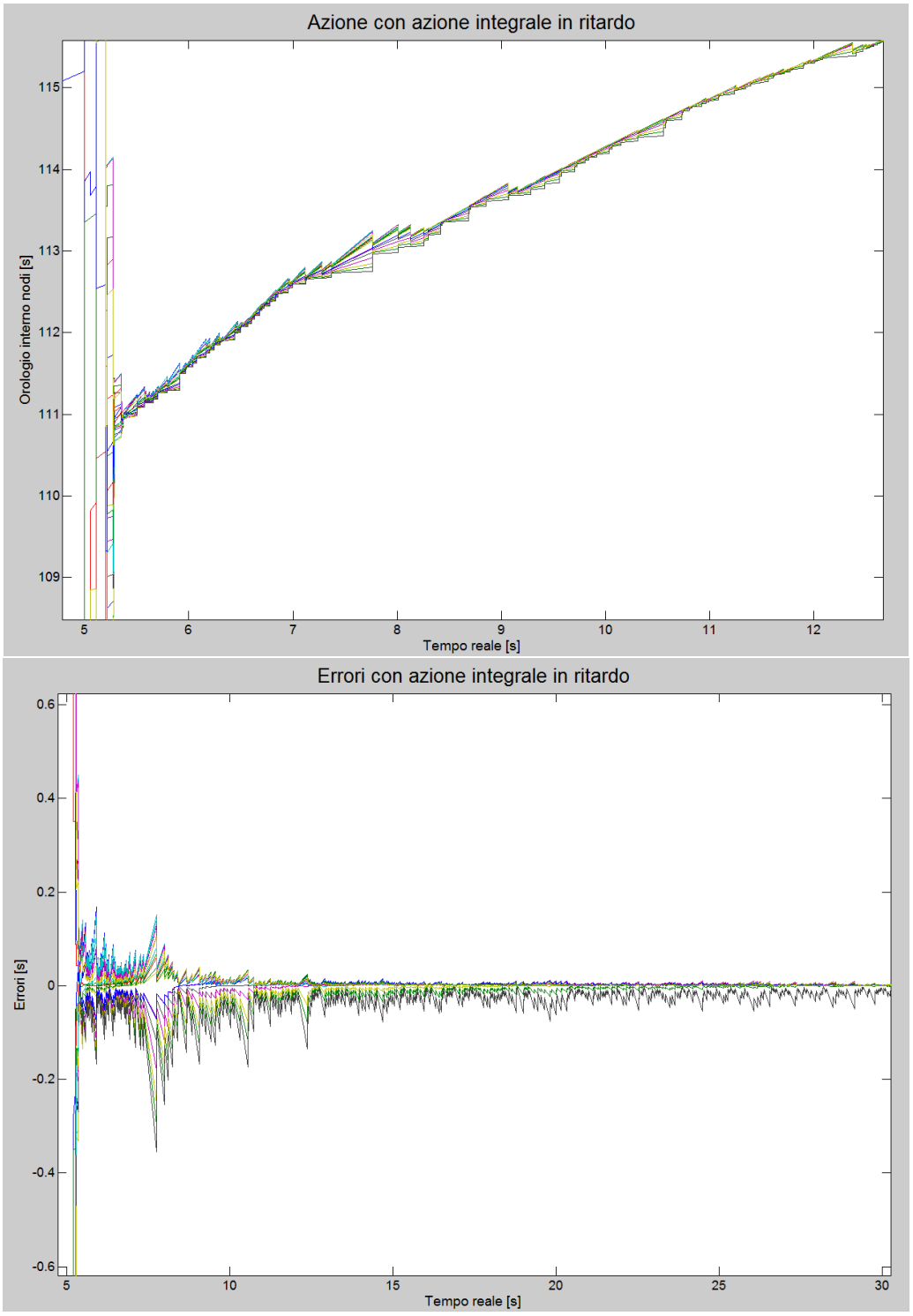


Figura 3.5: Dettaglio dell'andamento della stima del tempo e degli errori usando il metodo dell'azione integrale in ritardo di 1 secondo, con $\epsilon = 1\,Hz$, $c = 200\,secondi$, $\alpha = 0.5$.

Applicando questo metodo, in Fig. 3.5 è visualizzabile in dettaglio l'andamento degli errori partendo dalle condizioni iniziali "pessimistiche" e si possono trarre considerazioni completamente analoghe a quelle descritte per il metodo precedente.
Si hanno prestazioni profondamente migliori rispetto alla sola α costante (nella simulazione $\alpha = 0.5$) ma si continuano ad avere gli stessi problemi elencati per il metodo con α che cresce "a rampa".
Il problema prevalente risulta che questo approccio è particolarmente poco robusto: se cambiassero le condizioni iniziali si potrebbe avere un controllo PI con un valore di α troppo elevato che pregiudicherebbe la stabilità del sistema oppure, al contrario, con condizioni iniziali vantaggiose che renderebbero utile, ai fini della velocità di convergenza, l'applicazione fin dal principio del controllore PI.
Equivalentemenre al caso precedente si ha che il controllo dipende fortemente da com'è strutturata inizialmente la rete dei nodi (differenza di offset, differenza di skew e archi di comunicazione).

3.3 α "intelligente"

E' stato studiato un metodo che riuscisse a far variare in continuazione i parametri α_i relativi ad ogni nodo in funzione delle informazioni disponibili sulla rete a seguito delle comunicazioni locali tra i nodi. L'unica informazione che ogni nodo riceve periodicamente e sulla quale si baserà il calcolo dell'α ottima, è l'offset tra il proprio orologio interno e quello del nodo comuncante, visto come "posizione" relativa, nella dimensione temporale, di un nodo rispetto ai propri vicini.

3.3.1 Motivazione teorica e scelta dell'α ottima

Per provare a fornire una motivazione teorica (oltre a quelle pratiche viste precedentemente) sull'idea che stà alla base di utilizzare un parametro α funzione della distanza temporale (offset) tra nodo mittente e destinatario, ci si pone in una semplice condizione fornendo le seguenti ipotesi:

- 2 nodi, con frequenza di clock interno rispettivamente f_1 e f_2 e stima del periodo di clock interno $\triangle_1 = \frac{1}{\hat{f}_1}$ e $\triangle_2 = \frac{1}{\hat{f}_2}$;
- protocollo di comunicazione gossip simmetrico con peso $\beta = \frac{1}{2}$.

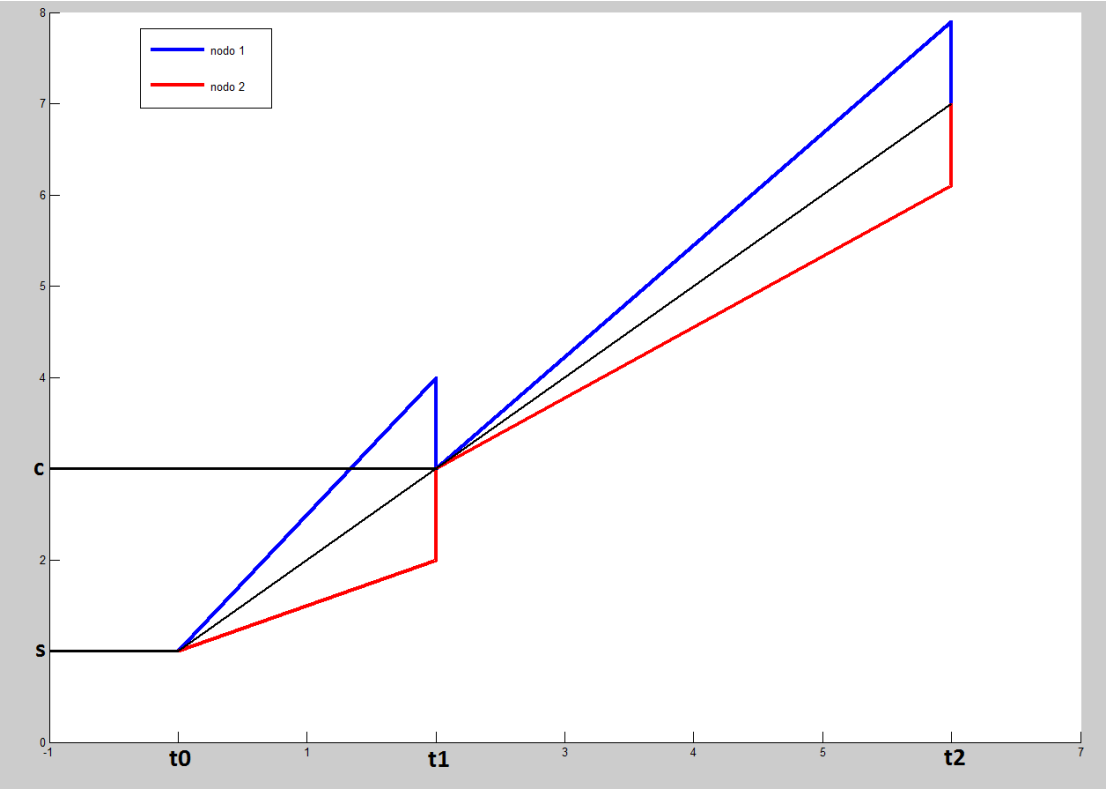


Figura 3.6: Evoluzione dei 2 nodi usati per la motivazione teorica.

Adottando il modello di aggiornamento (2.5) e (2.6) visto nel secondo capitolo si trova:

$$\begin{cases} x_1(t_2^-) = & \frac{1}{2}\left(x_1\left(t_1^-\right) + x_2\left(t_1^-\right)\right) + \triangle_1(t_1)\, f_1(t_2 - t_1) \\ x_2(t_2^-) = & \frac{1}{2}\left(x_2\left(t_1^-\right) + x_1\left(t_1^-\right)\right) + \triangle_2(t_1)\, f_2(t_2 - t_1) \end{cases} \tag{3.1}$$

dove

$$\begin{cases} x_1\left(t_2^+\right) = & \frac{1}{2}\left(x_1\left(t_2^-\right) + x_2\left(t_2^-\right)\right) \\ x_2\left(t_2^+\right) = & \frac{1}{2}\left(x_1\left(t_2^-\right) + x_2\left(t_2^-\right)\right) \end{cases} \tag{3.2}$$

e definendo $c = \frac{1}{2}\left(x_1\left(t_1^-\right) + x_2\left(t_1^-\right)\right)$, si ottiene

$$\begin{cases} x_1(t_2^-) = & c + \triangle_1(t_1) f_1(t_2 - t_1) \\ x_2(t_2^-) = & c + \triangle_2(t_1) f_2(t_2 - t_1) \end{cases} \tag{3.3}$$

Aggiornando la stima del clock $\triangle_{1,2}$ usando il classico controllore PI con parametro α della parte integrativa, si ottiene:

$$\triangle_{1,2}(t_2) = \triangle_{1,2}(t_1) + \frac{\alpha}{2}\left(-x_{1,2}(t_2^-) + x_{2,1}(t_2^-)\right) \qquad \text{e sostituendo con l'eq (3.3) si ottiene} \tag{3.4}$$

$$= \triangle_{1,2}(t_1) + \frac{\alpha}{2}\left(-c - \triangle_{1,2}(t_1) f_{1,2}(t_2 - t_1) + c + \triangle_{2,1}(t_1) f_{2,1}(t_2 - t_1)\right) \tag{3.5}$$

$$= \triangle_{1,2}(t_1) + \frac{\alpha}{2}(t_2 - t_1)\left(-\triangle_{1,2}(t_1) f_{1,2} + \triangle_{2,1}(t_1) f_{2,1}\right) \tag{3.6}$$

dove si ricorda che $\triangle_i(t) f_i$ rappresenta la pendenza della retta (skew) che dovrebbe valere 1 se la stima del clock interna fosse perfetta.

Come si nota da questi risultati, ottenuti in questo semplice esempio, la variazione di Δ non dipende unicamente dalla differenza di "pendenza" della retta $-\triangle_{1,2}(t_1) f_{1,2} + \triangle_{2,1}(t_1) f_{2,1}$ ma anche dalla differenza $t_2 - t_1$, intervallo di tempo trascorso dall'attuale comunicazione alla precedente.

Nel caso in cui

$$t_2 - t_1 \ggg -\triangle_{1,2}(t_1) f_{1,2} + \triangle_{2,1}(t_1) f_{2,1} \tag{3.7}$$

il controllo sarebbe influenzato più dal tempo trascorso che dall'effettiva differenza di pendenza delle 2 rette e quindi si avrebbe un elevato rischio di ottenere instabilita.
Bisognerebbe quindi cercare in qualche modo di far dipendere il parametro α dall'intervallo temporale $t_2 - t_1$.

Per semplicità si pone $\alpha = \frac{\overline{\alpha}}{(t_2 - t_1)}$ ottenendo quindi:

$$\triangle_{1,2}(t_2) = \triangle_{1,2}(t_1) + \frac{1}{2}\frac{\overline{\alpha}}{(t_2 - t_1)}(t_2 - t_1)\left(-\triangle_{1,2}(t_1) f_{1,2} + \triangle_{2,1}(t_1) f_{2,1}\right) \tag{3.8}$$

$$= \triangle_{1,2}(t_1) + \frac{1}{2}\overline{\alpha}\left(-\triangle_{1,2}(t_1) f_{1,2} + \triangle_{2,1}(t_1) f_{2,1}\right) \tag{3.9}$$

un sistema tempo invariante dove la variazione di $\triangle$ dipende unicamente dalle "pendenze" delle rette e non dal tempo trascorso tra una comunicazione e la precedente. Così facendo si risolve semplicemente il problema dell'influenza di $t_2 - t_1$.

Nella realtà però questo valore non risulta mai disponibile.

Si nota però che

$$dist(t_2^-) = |x_1(t_2^-) - x_2(t_2^-)| = |c + \triangle_1(t_1) f_1(t_2 - t_1) - c - \triangle_2(t_1) f_2(t_2 - t_1)| \tag{3.10}$$

$$= |(t_2 - t_1) \cdot (\triangle_1(t_1) f_1 - \triangle_2(t_1) f_2)| \tag{3.11}$$

Si vede quindi come la distanza sia proporzionale all'intervallo di tempo $t_2 - t_1$.
Supponendo che la differenza delle pendenze sia contenuta, si può ipotizzare che la distanza assuma un valore grande se è passato molto tempo da una comunicazione alla precedente, cioè se $t_2 - t_1$ è grande. Usando un certo parametro α che diminuisca all'aumentare della distanza si riuscirebbe ad ottenere, come visto prima, un sistema tempo invariante e quindi più facilmente stabilizzabile. Questo ragionamento si può anche rapportare al voler ottenere gli stessi pregi del controllore PI con azione integrale applicata in ritardo. Come visto in precedenza si vorrebbe usufruire di un controllo che non tenda a rendere instabile il sistema applicando un parametro α troppo elevato nella parte iniziale del processo di sincronizzazione.

Bisogna inoltre considerare che nelle simulazioni effettuate, dove non si utilizza un protocollo di comunicazione simmetrico, ci sarà anche da considerare l'effetto degli offset. Di conseguenza una distanza elevata non è da interpretare solamente come alta differenza di pendenza, ma è colpa anche di possibili offset oltre che da $t_2 - t_1$ elevati.
Ridurre α in funzione della distanza pare allora essere una soluzione sensata, anche per le considerazioni fatte precedentemente. È quindi necessario disporre di un valore del parametro α_i che diminuisca all'aumentare della distanza temporale tra 2 nodi e che sia elevato quando le distanze sono contenute in modo tale da assicurare convergenza al sincronismo.
Viene illustrato in Fig. 3.7 l'andamento di una funzione (non lineare) che può rispondere alle richieste sopra evidenziate.

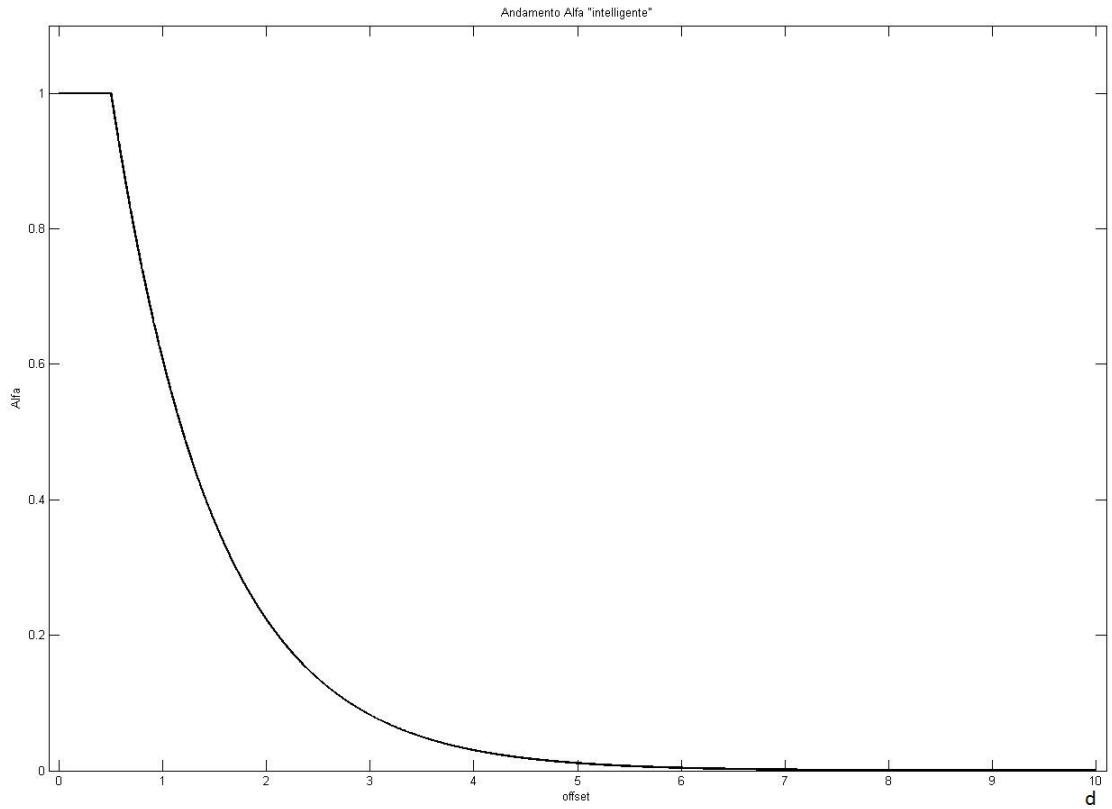


Figura 3.7: Andamento $\frac{\alpha_j}{\overline{\alpha}}$ con $\tau = 0.5$.

Chiamato $\overline{\alpha}$ il valore massimo che raggiungerà il guadagno della parte integrativa, τ il punto di discontinuità (della derivata) e d la "distanza temporale" tra i due nodi

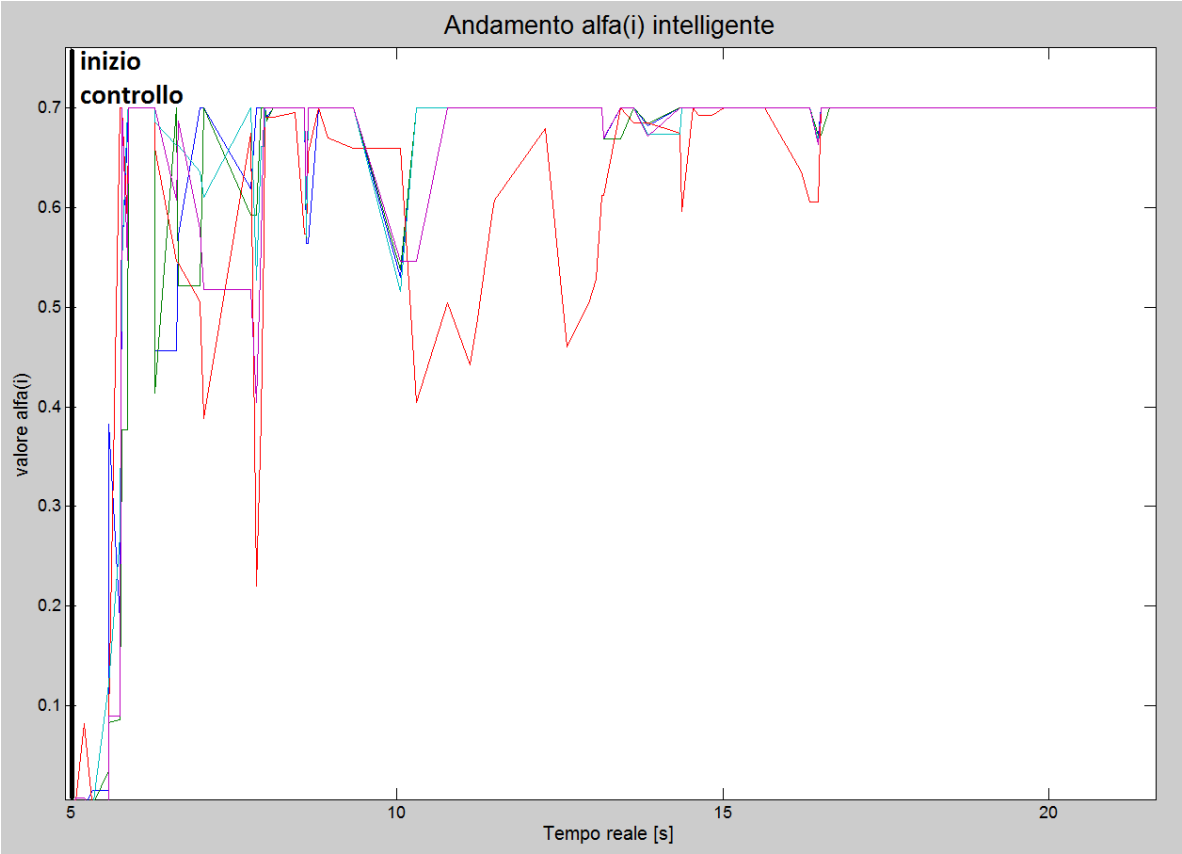
$$d_{ij}(t) = |x_i(t) - x_j(t)| \tag{3.12}$$

si ha:

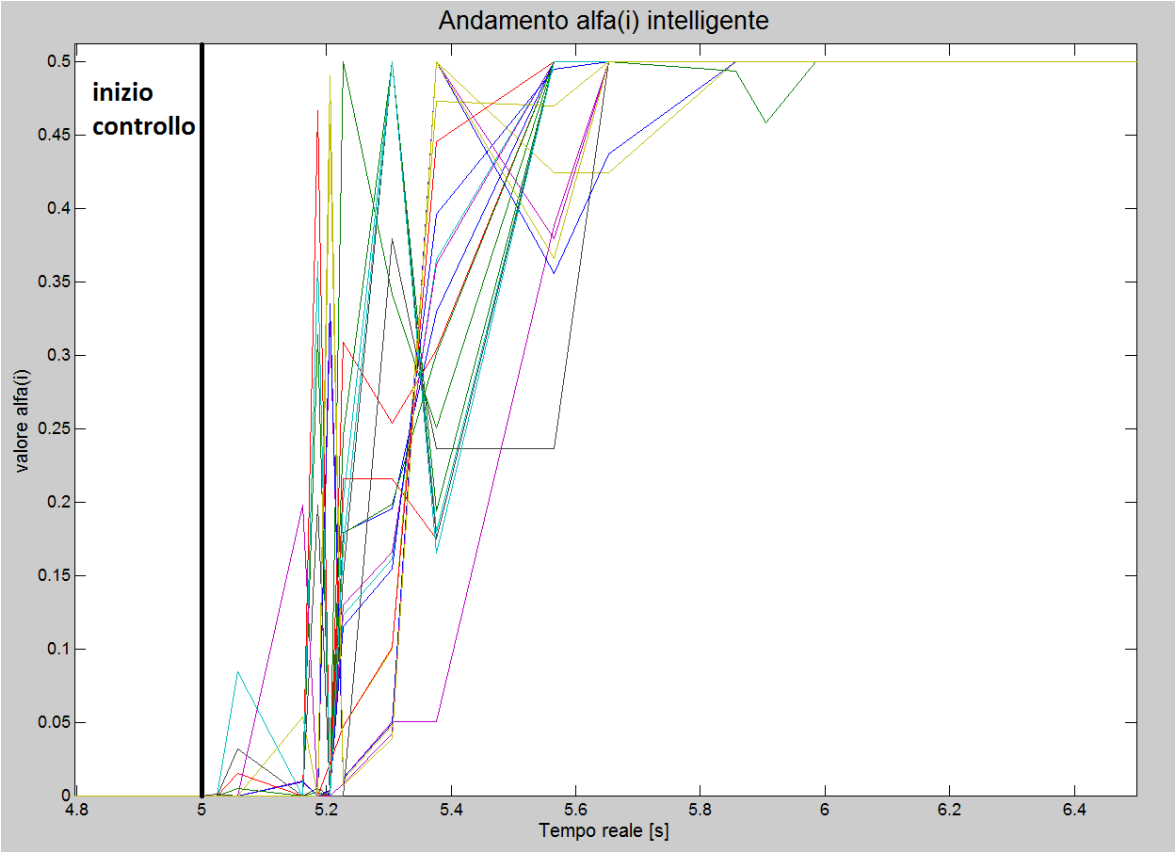
$$\alpha_j = \overline{\alpha} * \begin{cases} 1 & d_{ij} \in [0, \tau] \\ e^{-d_{ij}+\tau} & d_{ij} \in (\tau, +\infty) \end{cases} \tag{3.13}$$

3.3.2 Simulazioni

Per applicare il risultato appena ottenuto all'algoritmo che governa la simulazione, risulta necessario modificare il codice come segue: ogniqualvolta che il nodo j riceve il valore di x_i' dal nodo i, prima viene calcolata la distanza temporale d_{ij} secondo (3.12) e successivamente viene calcolato il valore di α_j con (3.13), parametro che viene poi usato nell'abituale controllore PI.
I valori di τ e $\overline{\alpha}$ sono preimpostati per tutti i nodi e si possono modificare (per ogni simulazione) in modo da ottimizzare le prestazioni ma avendo sempre un occhio di riguardo per il compromesso tra robustezza, stabilità e prestazioni.



(a)

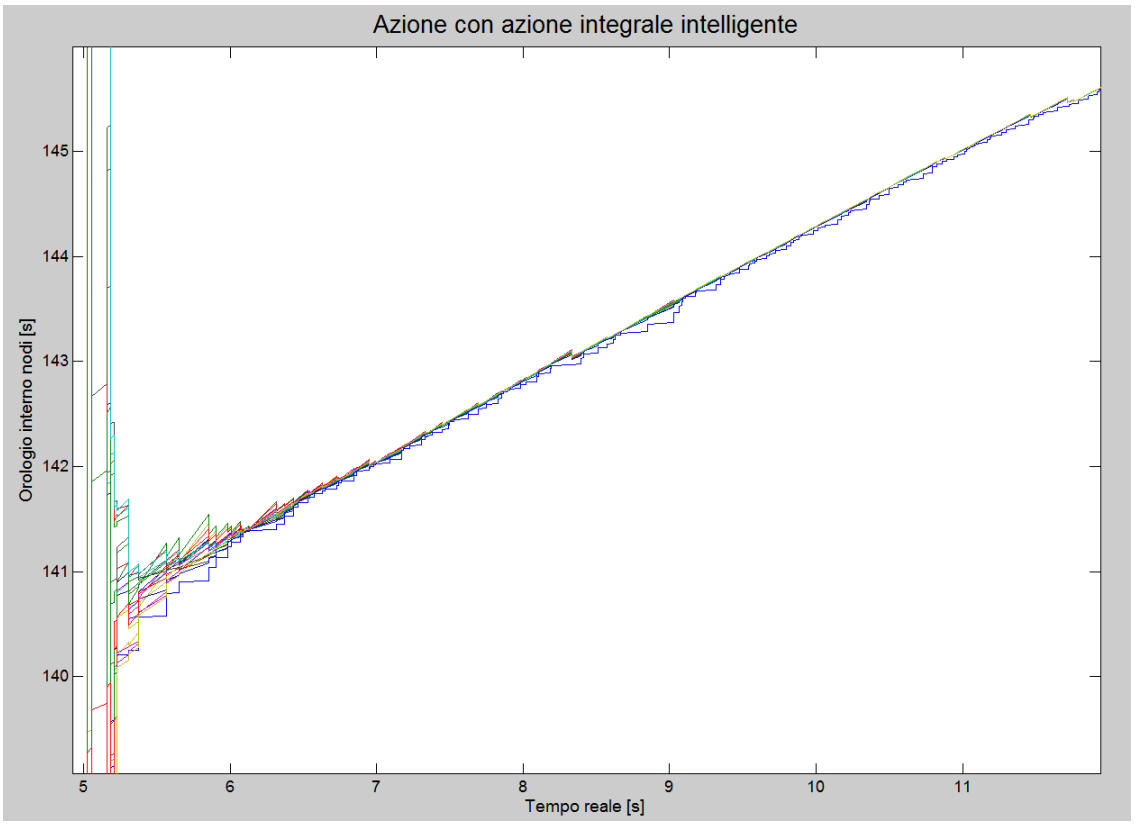


(b)

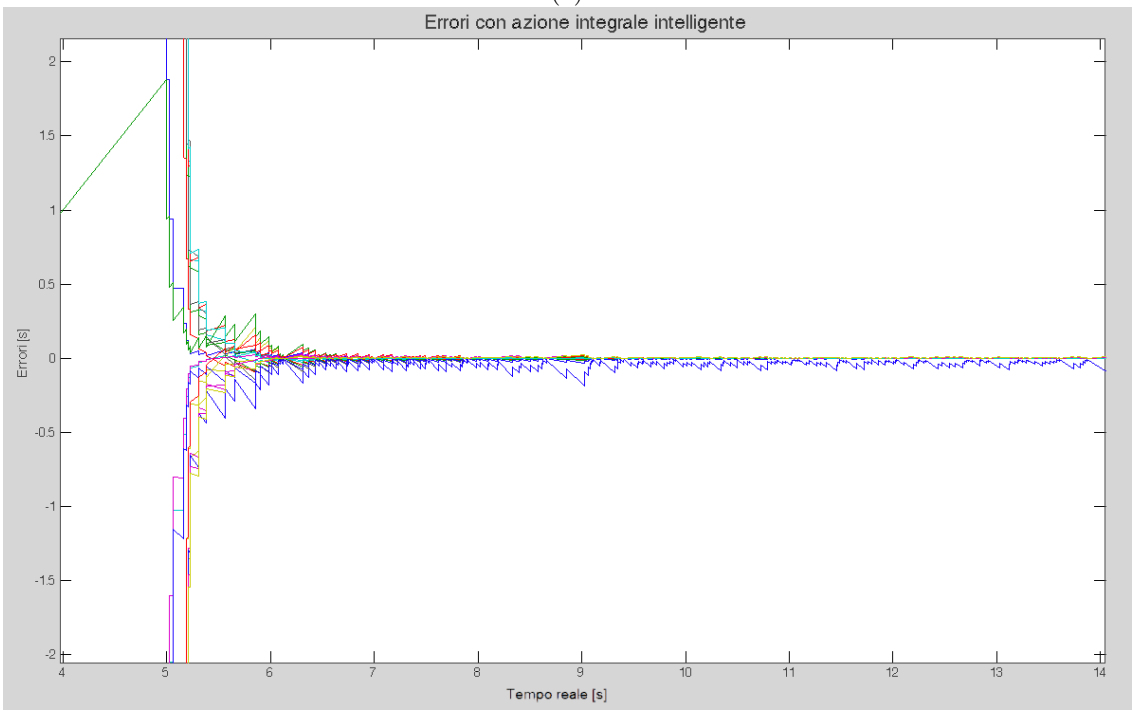
Figura 3.8: Esempio dell'andamento di α_j in una rete (a) di 5 nodi con $\tau = 0.5$ e $\overline{\alpha} = 0.7$ ed in una rete (b) di 20 nodi con $\tau = 0.35$ e $\overline{\alpha} = 0.5$.

Facendo una simulazione su una rete di soli 5 nodi si ottengono i risultati illustrati in Fig. 3.8(a). Gli andamenti di α_j per ogni singolo nodo presentano le caratteristiche già incontrate nel metodo in cui l'azione integrale viene applicata in ritardo e nel metodo dove il parametro α cresce a rampa. Si presentano quindi gli stessi benefici elencati precedentemente ma incorporando ora anche il vantaggio della robustezza. Infatti i parametri di "ritardo" ed "incremento" risultano qui dinamici e cioè dipendono ad ogni istante da come sta evolvendo la sincronizzazione dei nodi in modo auto-adattivo. Si evidenzia (Fig. 3.8(b)) come i valori di α_j non siano monotoni crescenti ma riescono appunto ad essere i valori ottimali per ogni istante temporale. Si ottiene robustezza potendo anche impostare un valore "elevato" di $\overline{\alpha}$: il controllo assicurerà stabilità a prescindere dalle condizioni iniziali complessive della rete.
Dalle varie simulazioni svolte si nota infatti come il comportamento dinamico degli errori e degli orologi interni risulta ora simile tra una simulazione e l'altra, a prescindere dalle condizioni iniziali della rete.

Prendendo per esempio una rete di 20 nodi e inizializzando l'aggiornamento di stato con condizioni iniziali pessimistiche (in cui ci si è posti abitualmente) si delinea (Fig. 3.9) come i nodi convergano velocemente e come la convergenza al sincronismo sia particolarmente "pulita", nel senso che non si presentano picchi accentuati.
Si vede infatti anche nel grafico degli errori (Fig. 3.9(b)) come questi ultimi convergano più velocemente a zero senza accennare a divergere.



(a)



(b)

Figura 3.9: Simulazione con $\overline{\alpha} = 0.5$, $\epsilon = 1\,Hz$, $c = 200\,secondi$, $N = 20$ e $\tau = 0.3$. Andamento del tempo (a) e degli errori (b)

Unica considerazione negativa di questo metodo è la sua complessita implementativa causata dalla funzione esponenziale su cui è basato il calcolo dell'α ottimale.
Solitamente i microprocessori di una rete di sensori wireless, per esempio, non dispongono di una potenza di calcolo sufficiente a garantire calcoli complessi in tempi ragionevoli. Per implementare questo metodo su di un microprocessore si dovrebbe quindi utilizzare una funzione di più facile realizzazione, come per esempio approssimare la funzione (3.13) mediante una spezzata composta da segmenti di retta.
Trascurando questo elemento negativo, il metodo presentato risulta prestazionale ed offre stabilità e robustezza come si desiderava.

3.4 Termine integrativo con memoria

Un ulteriore possibile modifica, per cercare di trovare un controllore PI robusto e prestazionale, è l'aggiunta di un cosiddetto "effetto memoria".
L'idea che viene qui presentata si pone alla base della teoria riguardante l'inserimento del termine derivativo nel controllore PI, metodo approfondito nel prossimo paragrafo.
Per regolare il proprio skew, ogni nodo tiene conto sia dell'offset temporale all'istante di ricezione di informazione da un altro nodo, sia di un secondo offset temporale relativo ad un precedente istante di aggiornamento e mantenuto nella memoria interna del nodo.

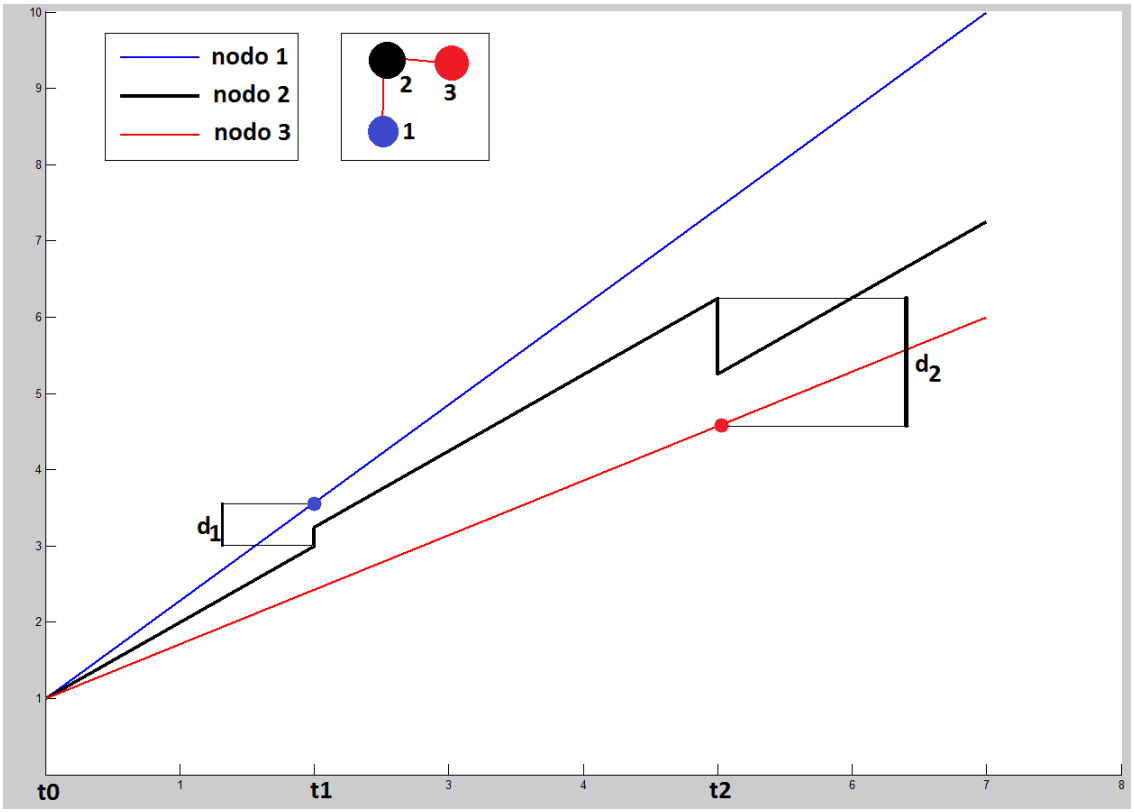


Figura 3.10: Evoluzione dei 3 nodi usati per le motivazioni con grafo di comunicazione

Per dare una motivazione di ciò, si può analizzare il semplice esempio di Fig. 3.10 che rappresenta una rete di 3 nodi aventi skew[1] tali per cui $\gamma_1 > \gamma_2 > \gamma_3$, all'istante t_1 si accende e comunica il nodo 1 mentre all'istante t_2 si accende e comunica il nodo 3. In entrambi i casi, come si vede dal grafo di comunicazione, è solamente il nodo 2 ad aggiornare i propri offset e skew secondo il normale controllore PI.
Si avrà quindi:

$$\gamma_2(t_1^+) > \gamma_2(t_0^+) \text{ e } \gamma_2(t_2^+) < \gamma_2(t_1^+) \tag{3.14}$$

[1] γ_i è il valore di skew nel nodo i.

cioè si ottiene un primo aumento e una successiva diminuzione dello skew. In determinate condizioni si potrebbe verificare anche il caso limite $\gamma_2(t_2^+) = \gamma_2(t_0^+)$.
Intuitivamente si vuole quindi cercare il modo per modificare $\gamma_2(t_2^+)$ facendo riferimento ad entrambi gli offset temporali (d_1 e d_2 in Fig. 3.10) in t_1 e t_2, in modo tale da ridurre la dinamica "non necessaria" degli skew e rendere il sistema generale più stabile, anche se si potrebbe ottenere un leggero calo delle prestazioni complessive dovuto ai ritardi introdotti dall'effetto memoria.

Sia ora

$$d_{ij,m}(t) = \lambda \cdot d_{ij}(t) + (1 - \lambda) \cdot m_j \tag{3.15}$$

il nuovo offset temporale del nodo j dove:

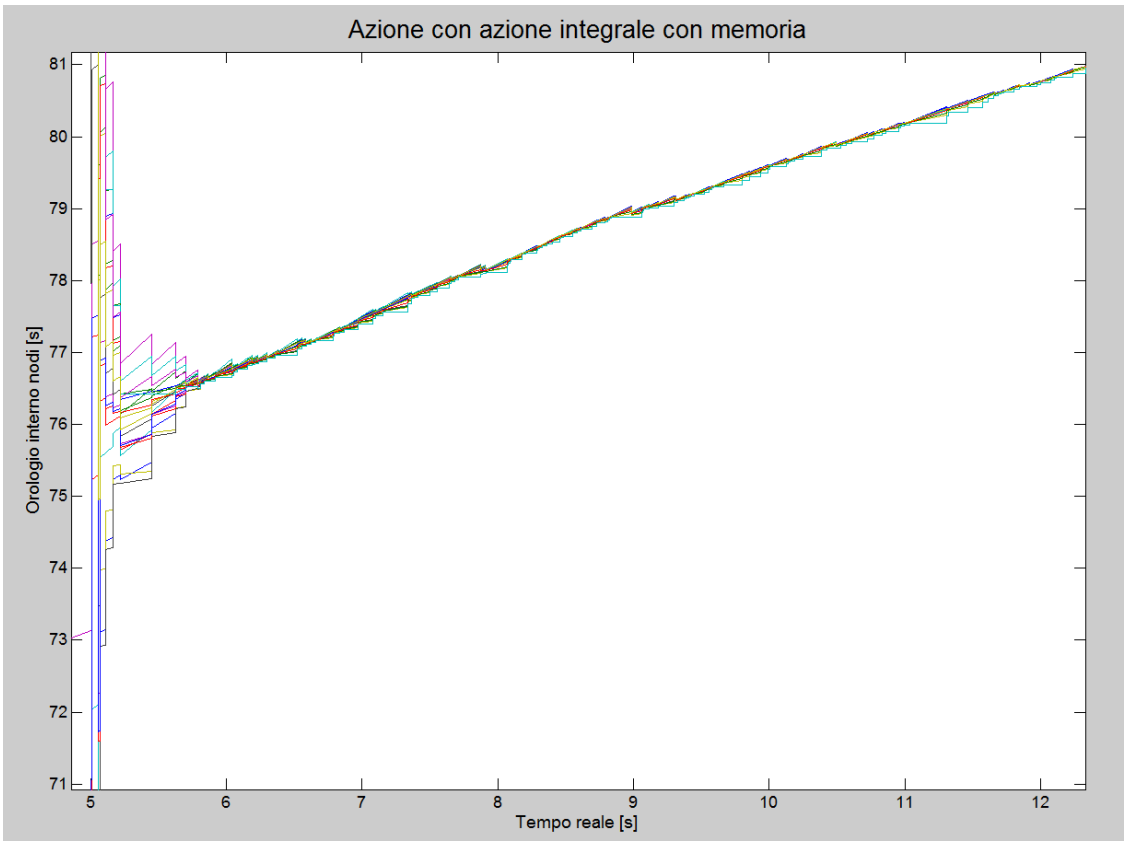
- $\lambda \in [0, 1]$ è un parametro che pesa l'effetto memoria. Se $\lambda = 0.5$ si ha la media esatta tra d_{ij} e m_j mentre se si vuole tener conto più dell'istante presente che di quello passato mantenuto in memoria, si pone $\lambda > 0.5$. Per il viceversa $\lambda < 0.5$;
- m_j è l'offset mantenuto in memoria. Per capire come si aggiorna la variabile m_j consideriamo 2 istanti temporali consecutivi t_a e t_b in cui il nodo j riceve l'informazione, con $t_a < t_b$: si ha che $m_j(t_b) = d_{ij}(t_a)$ e questa variabile si aggiorna ad ogni successivo istante di ricezione dell'informazione da parte di un nodo vicino.

Una possibile via per implementare questo metodo è la modifica della parte integrativa del controllore PI in modo tale che l'aggiornamento di α e di $\triangle(t)$ siano in funzione di $d_{ij,m}$ al posto di d_{ij}, mantenendo invece uguale il controllo proporzionale. Per fare questo è sufficente modificare opportunamente il parametro α_j come $\alpha_j = \frac{d_{ij,m}}{d_{ij}}\bar{\alpha_j}$, riuscendo così ad ottenere, alla generica h-esima iterazione:

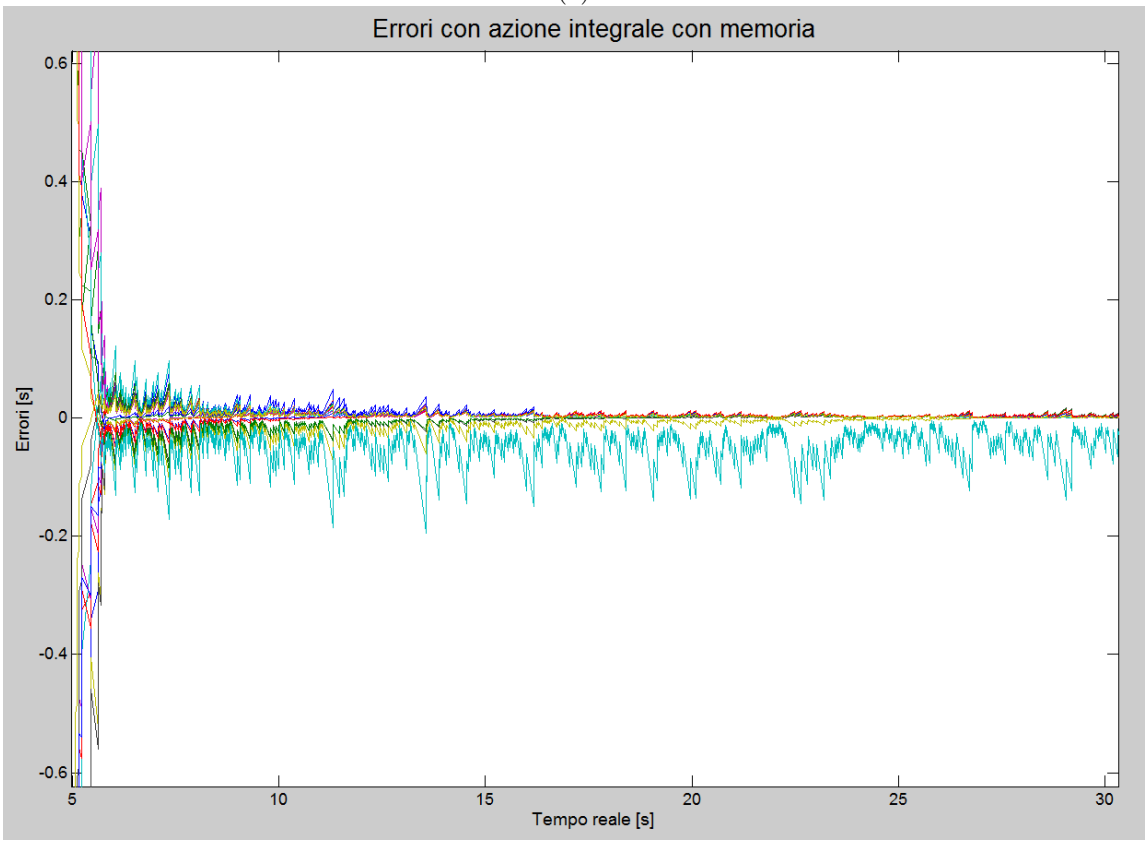
$$\Delta_j(h+1) = \Delta_j(h) + \alpha_j \cdot d_{ij} = \Delta_j(h) + \frac{d_{ij,m}}{d_{ij}}\bar{\alpha_j} \cdot d_{ij} = \Delta_j(h) + \bar{\alpha_j} \cdot d_{ij,m} \tag{3.16}$$

cioè è la formula di aggiornamento del periodo stimato ricavata precedentemente, ora però in funzione di $d_{ij,m}$.
Per calcolare $\bar{\alpha}_j$ si possono usare i metodi visti precedentemente, in particolare α "intelligente" (3.13) in funzione di $|d_{ij,m}|$ resta anche qui la migliore soluzione.

In Fig. 3.11, dove è stata eseguita una simulazione con i parametri in dadascalia, è evidente come questo metodo sia efficace in quanto riesce a stabilizzare il sistema ed è migliore dei metodi "crescita a rampa di α" o "termine integrale in ritardo" in quanto, usando l'α intelligente, riesce ad essere intrinsecamente robusto.
Resta comunque da dire che nelle condizioni usate non si riesce ad avere nessun tipo di vantaggio rispetto al metodo α intelligente e comporta un'ulteriore complicazione computazionale che non è detto sia supportata dai microprocessori delle WSN. Si può anzi vedere che i tempi di convergenza a volte sono addirittura superiori agli altri metodi esposti. Si può ritenere però che usando grafi non completi questa metodologia potrebbe risultare più efficace.



(a)



(b)

Figura 3.11: Simulazione con $\overline{\alpha} = 0.5$, $\epsilon = 1\,Hz$, $c = 200\,secondi$, $N = 20$ e $\tau = 0.05$. Andamento del tempo (a) e degli errori (b)

Capitolo 4

Controllore PID

In questo capitolo si introduce un controllo teoricamente più completo dei precedenti cercando di dimostrare fattibilità e convenienza nel suo campo applicativo.
Un controllo del tipo P consente un'azione *proporzionale* al segnale d'errore, non permettendo però sempre di raggiungere un valore finale assolutamente prossimo al riferimento (errore a regime). Per questo motivo nell'algoritmo studiato finora è stato inserita un'azione proporzionale all'*integrale* dell'errore permettendo di correggerne il valore a regime e giungere asintoticamente ad annullarlo.
Il controllo *Proporzionale-Integrale-Derivativo* è molto comune nelle applicazioni industriali. A differenza del controllore PI, l'aggiunta di una parte *derivativa* permette di regolare l'uscita in base alla velocità di variazione del segnale di errore in modo da migliorare la prestazione nella fase iniziale del controllo.
La taratura dei parametri relativi ad ogni parte del controllore PID può essere eseguita sperimentalmente attraverso delle semplici regole empiriche, come i metodi di Ziegler-Nichols.

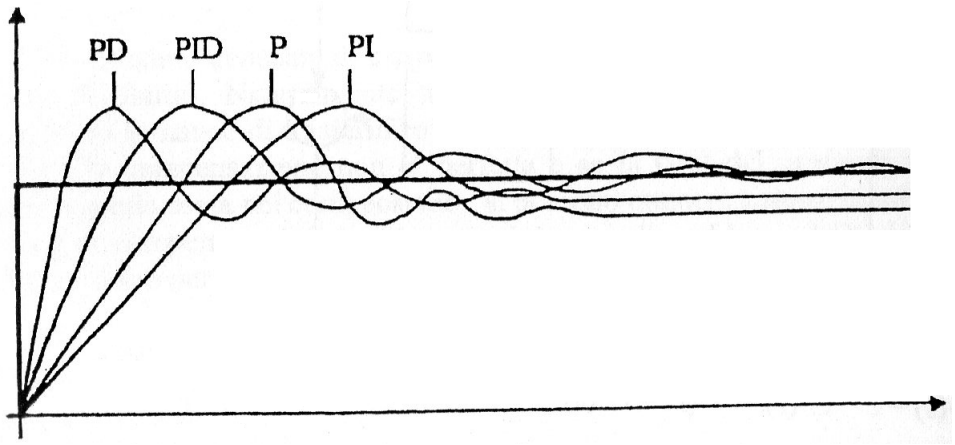


Figura 4.1: Esempio comparativo delle prestazioni ottenibili con le varie configurazioni di controllo.

Si vuole provare ad applicare questa tipologia di controllo al problema si sincronizzazione temporale, considerando nuovamente di eseguire il controllo singolarmente su ogni nodo.
Riferendosi quindi alla notazione introdotta nel secondo capitolo, alle ipotesi effettuate e considerate per tutto il progetto e infine ponendo $e(t_k) = x'_j(t_k) - x'_i(t_k)$ l'errore di sincronizzazione (differenza temporale) tra il nodo mittente j e il nodo destinatario i al passo k-esimo, si ha che la legge di controllo lineare, che descrive l'effetto di un controllore PID, risulta:

$$u_{PID}(t_k) = \begin{bmatrix} u'(t_k) \\ u''(t_k) \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 1 \\ \alpha + \frac{\beta}{\delta_k} \end{bmatrix} e(t_k) - \frac{1}{2} \begin{bmatrix} 0 \\ \frac{\beta}{\delta_k} \end{bmatrix} e(t_{k-1}) \tag{4.1}$$

ricordando che $\delta_k = t_k - t_{k-1}$.
Mediante la classica legge di aggiornamento $x(t_{k+1}) = x(t_k) + u(t_k)$, dove $x(t_k) = [x'(t_k)\ x''(t_k)]^T$, si

ottengono quindi:

$$x_i'(t_{k+1}) = x_i'(t_k) + \frac{1}{2}e(t_k) \tag{4.2}$$

$$= \frac{1}{2}(x_j'(t_k) + x_i'(t_k)) \tag{4.3}$$

relativa alla correzione dell'offset e

$$x_i''(t_{k+1}) = x_i''(t_k) + \frac{\alpha}{2}e(t_k) + \frac{\beta}{2}\frac{(e(t_k) - e(t_{k-1}))}{t_k - t_{k-1}} \tag{4.4}$$

$$= x_i''(t_k) + \frac{\alpha}{2}(x_j'(t_k) - x_i'(t_k)) + \frac{\beta}{2}\frac{((x_j'(t_k) - x_i'(t_k)) - (x_j'(t_{k-1}) - x_i'(t_{k-1}))}{\delta_k} \tag{4.5}$$

per la correzione dello skew.

Dalle precedenti equazioni si nota chiaramente che $x''(t_{k+1})$ dipende dagli stati $x'(t_k)$, $x''(t_k)$ e $x'(t_{k-1})$ e di conseguenza, introducendo il nuovo stato in forma semplificata[1] $x(t) = [x'(t)\ x''(t)\ x'(t-1)]^T$, si ottiene la forma matriciale

$$x(t+1) = \begin{bmatrix} x'(t+1) \\ x''(t+1) \\ x'(t) \end{bmatrix} = \begin{bmatrix} I_N - K(t) & \delta(t)D & 0 \\ -(\alpha + \frac{\beta}{\delta(t)})K(t) & I_N & K(t)\frac{\beta}{\delta(t)} \\ I_N & 0 & 0 \end{bmatrix} x(t) \tag{4.6}$$

dove

$$K(t) = \frac{1}{2}\sum_{j \in \mathcal{N}_i} e_j(e_j - e_i)^T \tag{4.7}$$

è la matrice che governa la corretta applicazione del controllo ai soli nodi interessati, già esposta nel primo capitolo, $D = diag(f_1, \cdots, f_N)$ con $f_i, i = 1, \cdots, N$ le frequenze interne dei clocks mentre $\delta(t) = costante = 1$.
Si può facilmente intuire, come già visto nella sezione 3.4 (Termine integrativo con memoria), che per il funzionamento dell'algoritmo qui proposto è necessario che ogni nodo memorizzi il valore dell'informazione ricevuta all'istante precedente, cioè in ogni istante di sincronizzazione dev'essere possibile utilizzare il precedente valore dell'errore di sincronizzazione $e(t_{k-1}) = x_j'(t_{k-1}) - x_i'(t_{k-1})$. Questo fatto potrebbe notevolmente complicare l'implementazione di questo metodo in WSN dalle ridotte capacità computazionali oppure senza memoria.

4.1 Simulazioni

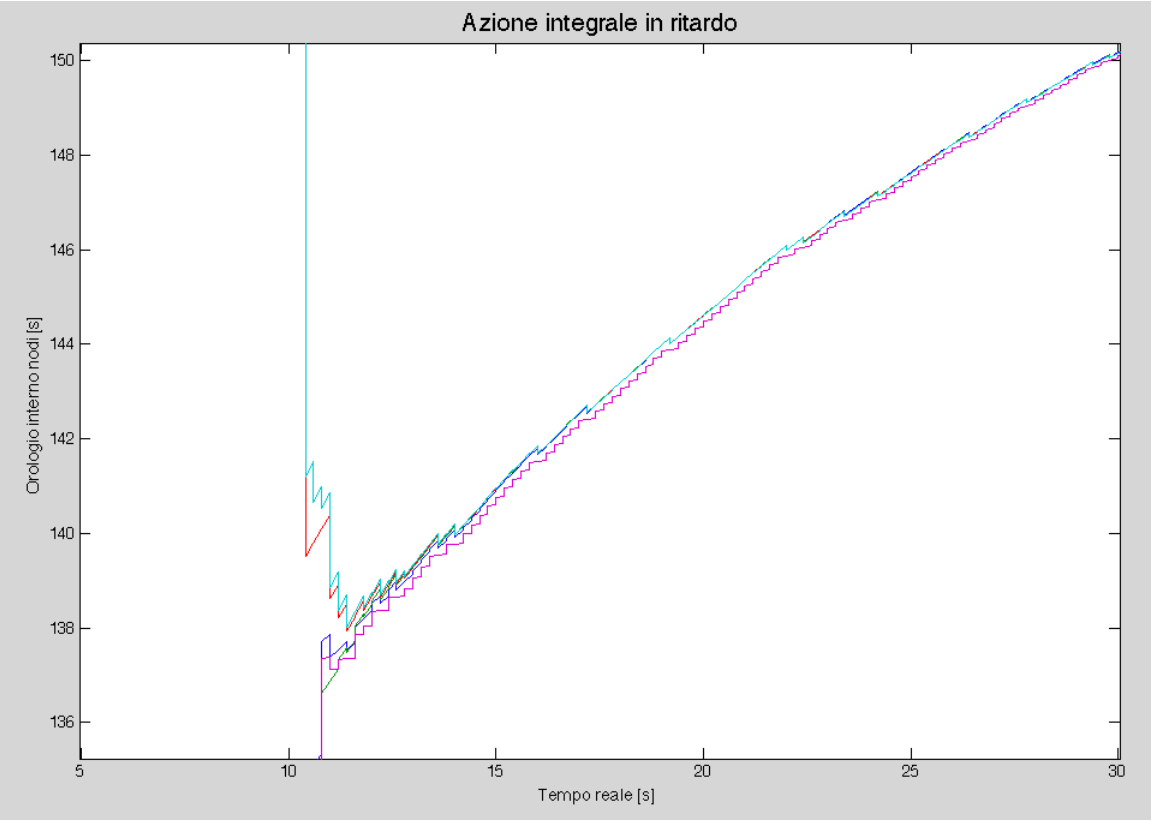
Per effettuare le prove simulative si vogliono preliminarmente introdurre delle semplificazioni rispetto alle ipotesi poste precedentemente riguardo il protocollo di comunicazione e la tipologia di WSN utilizzata.
Rispetto alla precedente scelta di utilizzare una distribuzione di Poisson, atta a descrivere la legge di accensione e trasmissione delle informazioni da parte di un nodo sulla rete, si è deciso di sfruttare una più semplice distribuzione di tipo uniforme in modo tale che l'intervallo temporale tra l'istante attuale di invio di dati e il precedente sia costante e per semplicità considerato pari a $\frac{1}{N}$. Inoltre, sempre per semplicità di trattazione, i test simulativi sono stati effettuati su di una WSN conforme ad un grafo fortemente connesso.
Questi accorgimenti sono stati volutamente imposti in modo tale da poter confrontare questo metodo di controllo nelle condizioni più semplici da poter implementare. Nel caso i confronti con i metodi descritti nel capitolo 3 dovessero offrire risultati rilevanti ai fini applicativi, si vorrebbe complicare l'algoritmo in modo da introdurre le ipotesi più complesse citate all'inizio di questa sezione.

Per questa fase simulativa è stato pensato di attivare la parte derivativa del controllore durante i primi passi dell'algoritmo di sincronizzazione e successivamente concludere il processo solamente utilizzando la parte integrale (per la correzione dello skew), mentre la parte Proporzionale ovviamente

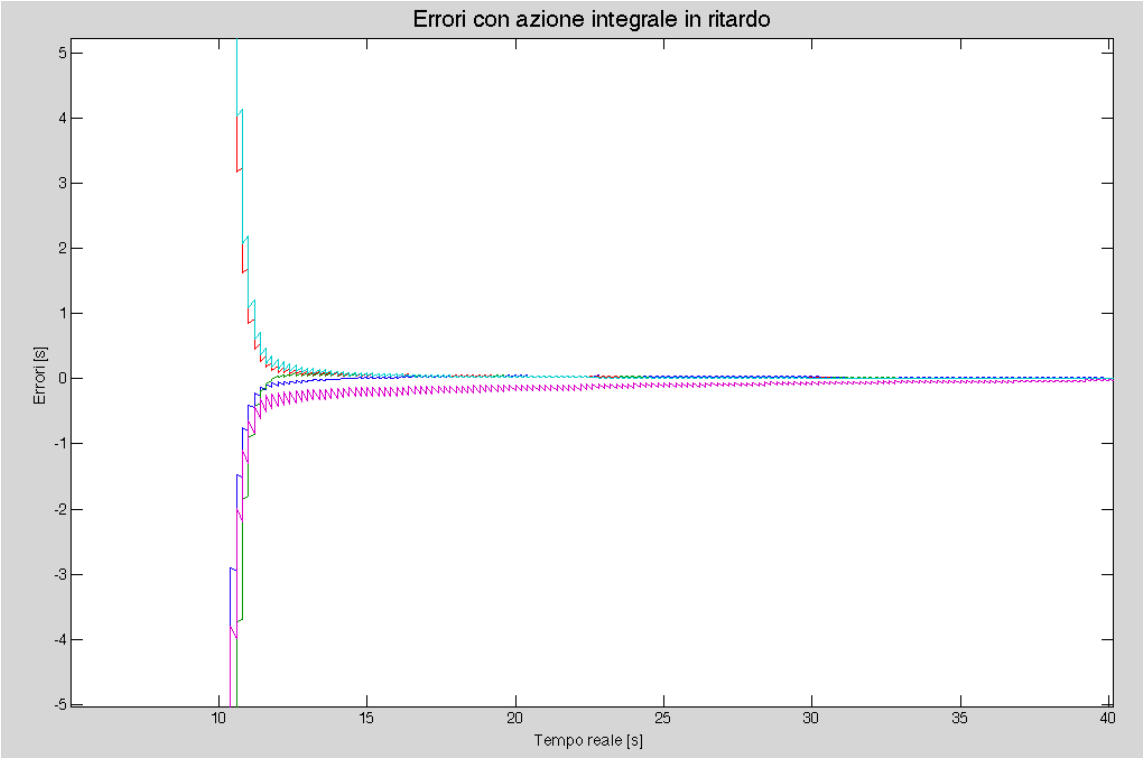
[1] Per effettuare in maniera più intuitiva la successiva parte simulativa.

rimane sempre presente (per la correzione dell'offset). In questo modo, negli istanti iniziali dell'algoritmo, si vorrebbe raggiungere un intorno molto prossimo alla condizione di convergenza in tempo minore rispetto al solo controllo proporzionale (idea deducibile dalla Fig. 4.1).
La simulazione viene eseguita utilizzando 5 nodi rappresentabili in un grafo completamente connesso. Come per le prove effettuate nei capitoli precedenti, le frequenze reali dei clock sono scelte casualmente tra 0 e 1 Hz mentre il vettore dei tempi iniziali è anch'esso scelto casualmente tra 0 e 200 secondi, in modo da evidenziare la differenza temporale iniziale.
Per testare le funzionalità del PID è stato deciso di confrontarlo con l'algoritmo basato su di un controllore PI con parametro α costante ma applicato dopo un certo tempo ed il tempo totale di evoluzione è stato scelto pari a 100 secondi.
Le prestazioni migliori con il controllore di tipo PI, partendo dalle condizioni iniziali sopra definite, si ottengono ritardando l'applicazione della parte integrativa di 10 passi dall'inizio del controllo. In questo modo viene inoltre assicurata la stabilità del sistema. Bisogna ricordare che essendo questo tipo di controllore poco robusto alla variazione delle condizioni iniziali, per assicurare sempre la stabilità ci si accontenterà di prestazioni peggiori. Per questa specifica simulazione è stata quindi ottimizzata la scelta dei parametri senza considerare la robustezza alla variazione delle condizioni iniziali.
Per il PID invece è stato scelto di limitare l'azione derivativa ai primi 10 passi dall'inizio del controllo. Come si nota nelle figure seguenti, l'azione derivativa permette una buona sincronizzazione temporale soprattutto nei casi in cui la differenza temporale iniziale risulti evidente. In generale, applicare il termine derivativo per ulteriori istanti di sincronizzazione non porta a miglioramenti in quanto le variazioni relativamente basse degli errori di misura non necessitano di una notevole velocità alla convergenza verso il valore finale, bensì di un'errore il più basso possibile, ottenibile solamente mediante il termine integrativo. Per le simulazioni qui effettuate invece, applicare la parte derivativa solo per i primi istanti offre la stessa prestazione dell'applicazione per tutta la durata dell'algoritmo, purchè il termine β sia sufficientemente limitato (nell'ordine del 10^{-2}).

I risultati seguenti sono stati ottenuti a seguito di un opportuno tuning dei parametri del controllore che ha portato a porre $\alpha = 0.3$, sia per il controllore PI che per quello PID, e $\beta = 0.01$.
Nelle figure che seguono sono riportati i risultati delle simulazioni effettuate con i due controllori sopra citati. Interessante il confronto di Fig.4.4 dove effettivamente possono essere comparate le prestazioni, soprattutto a regime, dei due controllori.

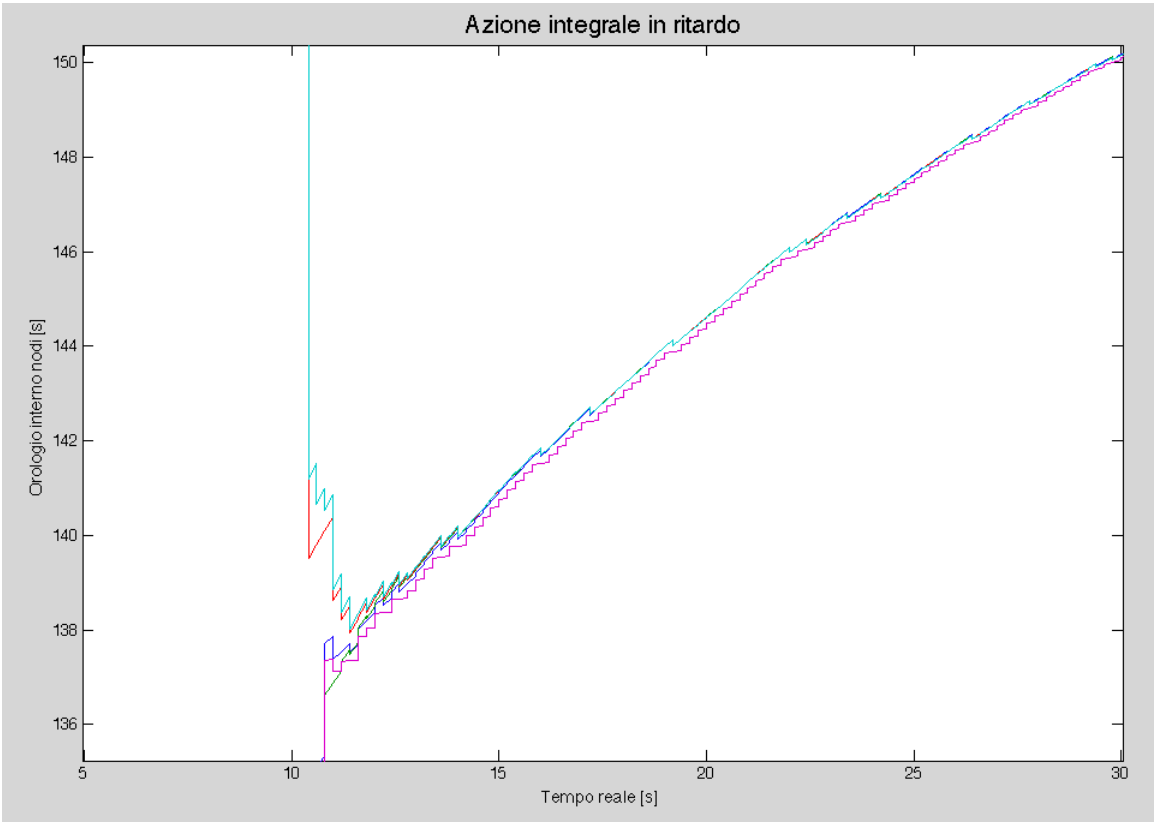


(a)

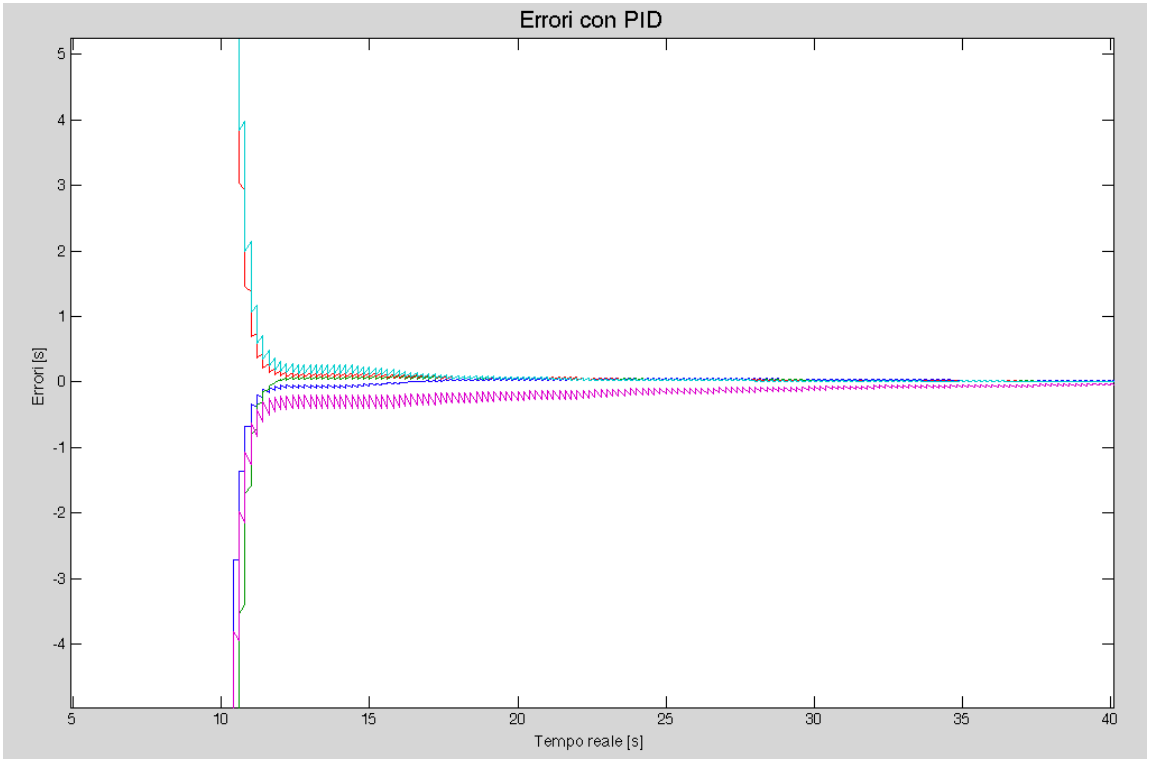


(b)

Figura 4.2: azione del controllore PI e $\epsilon = 1\,Hz$, $c = 200\,secondi$, $N = 5$ dove $\alpha = 0.3$. Primi 10 passi di evoluzione libera, seguono 10 passi di azione P e successivamente azione PI.



(a)



(b)

Figura 4.3: azione del controllore PID con $\epsilon = 1\,Hz$, $c = 200\,secondi$, $N = 5$ dove $\alpha = 0.3$ e $\beta = 0.01$. Primi 10 passi di evoluzione libera, seguono 10 passi di azione PD e successivamente azione PI.

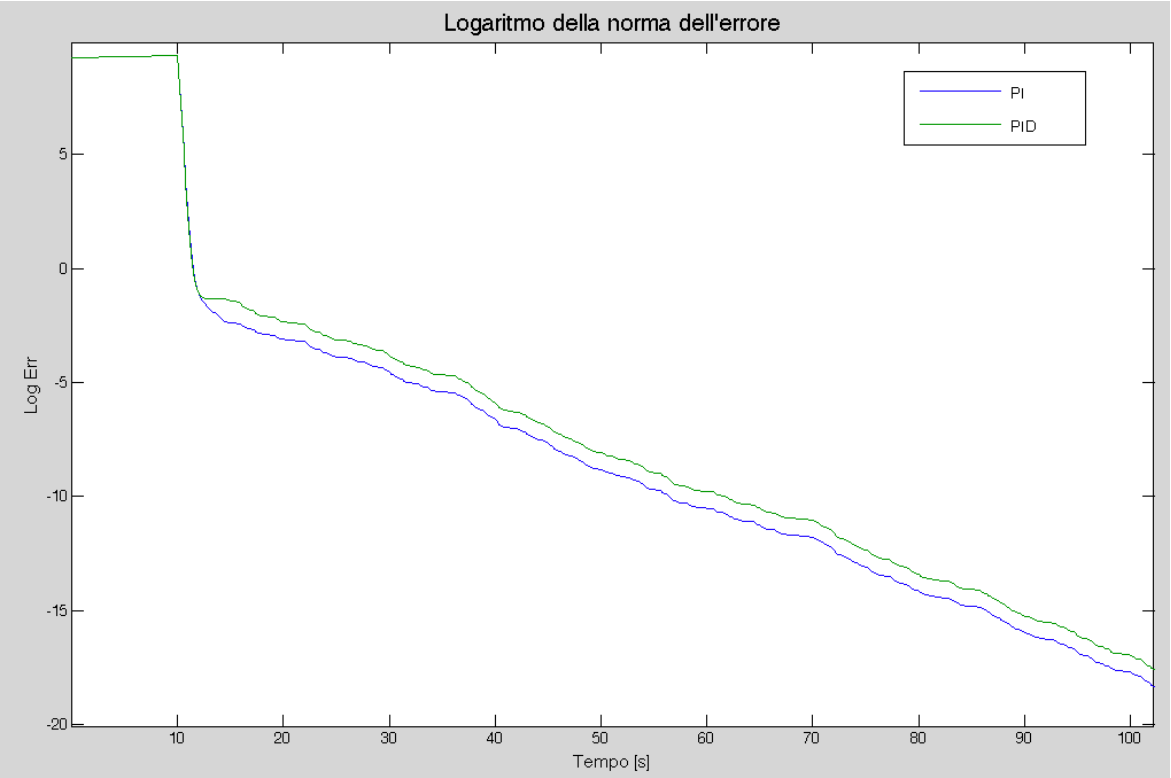


Figura 4.4: confronto tra i logaritmi della norma dell'errore di sincronizzazione per i controllore PI e PID dove $\epsilon = 1\,Hz$, $c = 200\,secondi$, $N = 5$ dove $\alpha = 0.3$ e $\beta = 0.01$.

Da queste prime simulazioni si può subito capire come l'applicazione del termine derivativo non porta ad avere sostanziali miglioramenti rispetto al semplice controllo PI con azione Integrale ritardata. Si nota in ogni caso che mantenendo il parametro β sufficientemente limitato (nell'ordine del 10^{-2}) il controllore PID in generale offre buone prestazioni ma sempre equivalenti o leggermente peggiori rispetto al controllore PI, soprattutto per quanto riguarda l'errore di sincronizzazione a regime (come si può visionare in Fig.4.4).
Evidentemente già quest'ultima soluzione risulta quella ottima per lo scopo proposto e quindi risulta inutile complicare la dinamica del controllore incorporando un termine derivativo parzialmente presente o addirittura applicato per tutta l'evoluzione dell'algoritmo. Non si procede quindi all'incorporamento dell'ipotesi più complessa citata precedentemente, quale descrizione dei tempi di trasmissione mediante distribuzione di Poisson di intensità λ, dato che già utilizzando una condizione più semplice non si ottengono risultati soddisfacenti.

Capitolo 5

Conclusioni e sviluppi futuri

Dopo aver analizzato nel Capitolo 2 la struttura del controllo PI per la sincronizzazione temporale di WSN ed aver proposto nuove tecniche di miglioramento dell'algoritmo nel Capitolo 3, si vuole ora concludere il lavoro mostrando il resoconto globale di ciò che ne è emerso. Per far questo sono state effettuate delle simulazioni atte a confrontare le peculiarità dei 5 metodi proposti analizzando il logaritmo della norma dell'errore di sincronizzazione.
Per una prima simulazione effettuata su di una rete composta da $N = 20$ nodi e della durata di $T = 100$ secondi, ci si è posti nelle stesse condizioni iniziali per tutti i metodi, scelte in maniera randomizzata con parametri $c = 200\,secondi$, $\epsilon = 1\,Hz$ ed in particolare fissando:

- termine integrale costante: $\alpha = 0.3$;
- crescita a rampa di α: $\bar{\alpha} = 0.5$ e $\rho = 0.06$ (incremento);
- termine integrale costante e ritardato: $\alpha = 0.5$ e ritardo di 2 secondi;
- α intelligente: $\bar{\alpha} = 0.8$ e $\tau = 0.01$;
- termine integrativo con memoria: $\bar{\alpha} = 0.7$, $\lambda = 0.35$ (peso della memoria) e $\tau = 0.01$.

Questi valori sono giustamente stati tarati per offrire le migliori performance di ogni singolo metodo e garantendo stabilità e robustezza, in modo da poter svolgere il confronto tra tutti i controllori nelle condizioni migliori in funzione degli stessi stati iniziali.
Il grafico del logaritmo della norma degli errori, in Fig. 5.1, mostra effettivamente come il metodo classico, con controllore PI ad azione integrale costante, offra a regime il più elevato valore di errore di sincronizzazione viceversa il metodo denominato "α intelligente" possa raggiungere valori estremamente piccoli della stessa grandezza.
La motivazione di ciò è che usando algoritmi che sfruttano una sorta di auto-tuning dinamico dei guadagni (α "intelligente" e termine integrativo con memoria), si riesce ad avere più robustezza (quindi sempre stabilità) e si possono riuscire ad avere controllori nettamente più prestazionali potendo aumentare i valori massimi di α senza vincoli legati alla stabilità.
Se viceversa le condizioni iniziali fossero deterministiche (non solo i valori c ed ϵ delle simulazioni ma i veri e propri valori di offset iniziali per esempio), si riuscirebbero ad aumentare notevolmente le prestazioni dei metodi con azione integrale in ritardo e "α crescente". Infatti, potendo alzare i valori massimi di α e tarando rispettivamente il ritardo e l'incremento per quelle determinate condizioni iniziali senza eccessive preoccupazioni sulla robustezza del controllo (in quanto queste non varierebbero), si riuscirebbe ad avere velocità di convergenza prossime a quelle dei metodi adattivi robusti.
Purtroppo però lo stato iniziale della rete (skew e offset iniziali nel momento di inizio del controllo) non è noto a priori e quindi bisogna diminuire i guadagni dei metodi non intrinsecamente robusti per garantire sempre stabilità, oltre a lasciare un certo margine sul ritardo e sull'incremento per garantire robustezza.
Se, per esempio, le condizioni iniziali fossero più favorevoli di quelle proposte nella simulazione (c.i. usate per tarare i controllori per il caso peggiore che si possa presentare), i controllori avrebbero prestazioni nettamente peggiori di un metodo robusto che si adatta alle condizioni della rete (metodo α "intelligente").

Un altro tipo di simulazione effettuata per confrontare i metodi proposti è stato l'esecuzione di un ciclo di 50 diverse simulazioni, ognuna delle quali inizializzata da condizioni casuali scelte negli intervalli sopra citati ($c = 200\,secondi$ e $\epsilon = 1\,Hz$).
Per ognuna delle 50 simulazioni e per ognuno dei 5 metodi presentati è stato calcolato il tempo di convergenza, il tempo cioè che impiega il controllo sulla rete affinché tutti gli errori restino definitivamente entro un intervallo di confidenza, posto pari a 0.05 secondi.
Nel grafico di Fig. 5.2 si possono visionare questi tempi di convergenza per ognuna delle 50 diverse simulazioni e per ognuno dei 5 metodi esposti, grazie ai quali si ottengono (Tabella 5.1) i valori dei tempi medi di convergenza.
In linea di massima si hanno tempi di convergenza coerenti con l'andamento del logaritmo degli errori illustrato in Fig 5.1; il metodo dell'azione integrale costante è quello che presenta i tempi di convergenza peggiori, a causa prevalentemente del basso guadagno α, mentre i metodi adattivi riescono sempre a sincronizzare il sistema nel tempo minore.
Nella Tab. 5.1 sono anche riportate le medie (fra tutte le simulazioni) delle deviazioni standard del vettore contenente gli skew finali per i diversi metodi. Più questo numero è piccolo, più il metodo assicura uguaglianza tra gli skew a regime dei singoli nodi.

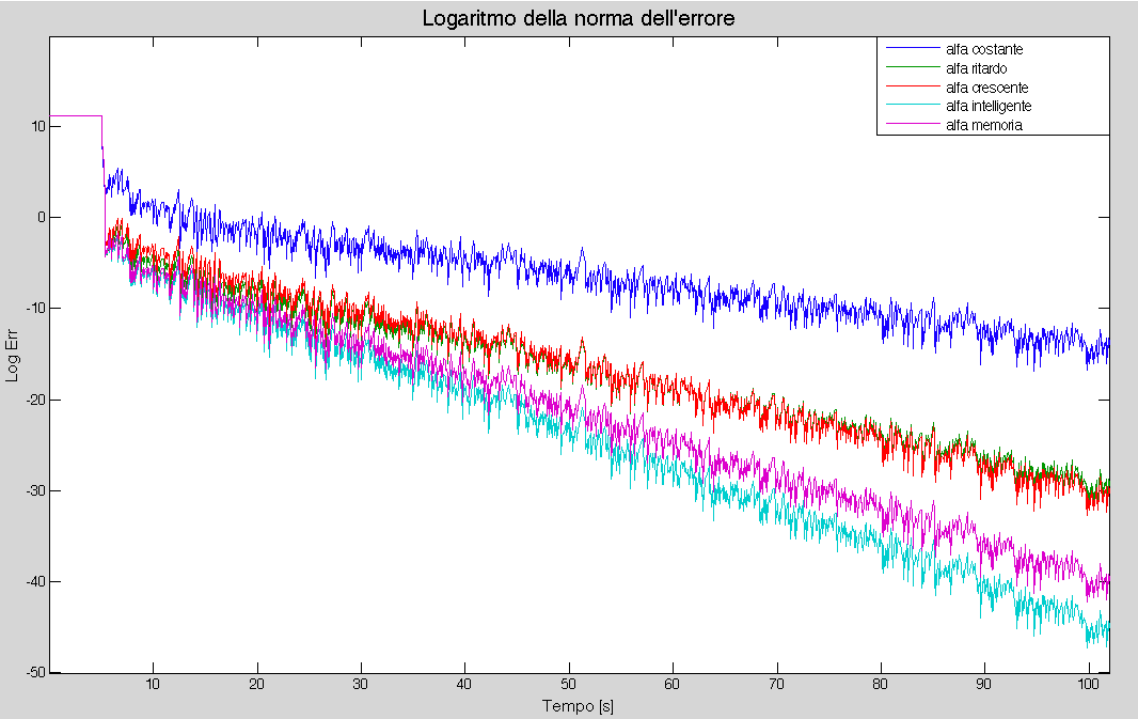


Figura 5.1: Sim. 1: Confronto tra i logaritmi della norma dell'errore di sincronizzazione per i 5 metodi proposti nel Cap.3 in un ciclo di 50 simulazioni eseguite con le c.i. e i parametri elencati ad inizio capitolo.

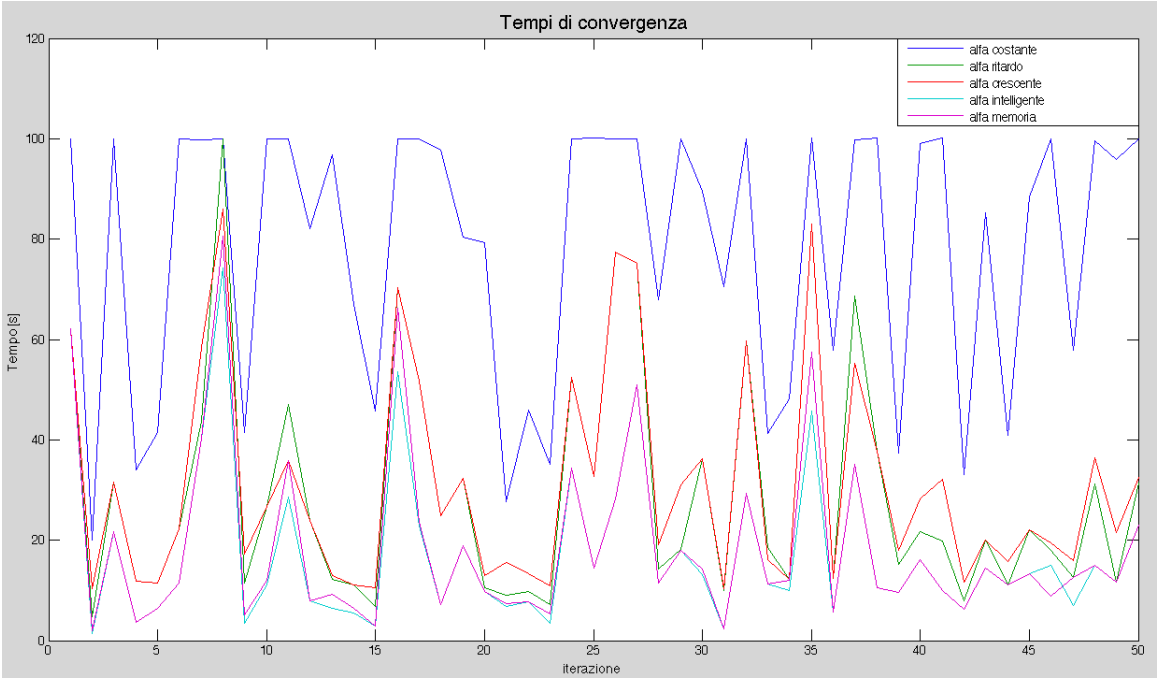


Figura 5.2: Sim. 1: Tempi di convergenza per i 5 metodi proposti nel Cap.3 in un ciclo di 50 simulazioni eseguite con le c.i. e i parametri elencati ad inizio capitolo.

termine integrale	costante	in ritardo	a rampa	intelligente	con memoria
Media	78.1443	29.4961	30.9844	17.8297	18.8320
Dev. Std.	0.1007	0.0065	0.0063	0.0028	0.0035

Tabella 5.1: Sim. 1: Medie e Deviazioni Standard per i 5 metodi proposti nel Cap.3 in un ciclo di 50 simulazioni eseguite con le c.i. e i parametri elencati ad inizio capitolo.

Mediante una seconda simulazione di durata $T = 100$ secondi, composta sempre dal ciclo di 50 simulazioni diverse, si è voluto testare l'efficienza dei 5 metodi esposti nel Capitolo 3 in una particolare situazione in cui anche gli intervalli dai quali vengono scelte le 50 condizioni iniziali fossero scelti in maniera casuale:

- $N = [0, 25]$ il numero dei nodi presenti sulla rete;
- $c = [0, 200]$ secondi, il valore massimo del loro offset;
- $\epsilon = [0, 1]$ Hz.

In questo modo, utilizzando gli stessi valori per i guadagni e i parametri elencati nella precedente simulazione, si vuole giudicare quale tipo di controllore si adatta meglio a condizioni iniziali completamente casuali, verificando se è vero quanto detto precedentemente.
In Fig. 5.3 si ha il grafico relativo ai tempi di convergenza, sempre entro una fascia di 0.05 secondi, dei 5 metodi relativi ad ogni iterazione del ciclo. Le medie ottenute in questo caso sono riportate nella Tabella 5.2, nella quale sono visionabili anche la media delle deviazioni standard del vettore degli skew finali per i diversi metodi.
Nuovamente, come è lecito aspettarsi, il controllore PI con parametro integrale "intelligente" offre le migliori caratteristiche in termini di velocità di convergenza e robustezza nonostante le condizioni iniziali variabili.
Il controllore che sfrutta il termine integrativo con memoria offre prestazioni comparabili poichè sfrutta lo stesso basilare concetto su cui è centrato il funzionamento del primo. Nota dolente però il fatto che l'implementazione di un metodo che ha bisogno di memorizzare dati, si pone ad un livello di complessità superiore rispetto al metodo con parametro integrale "intelligente".

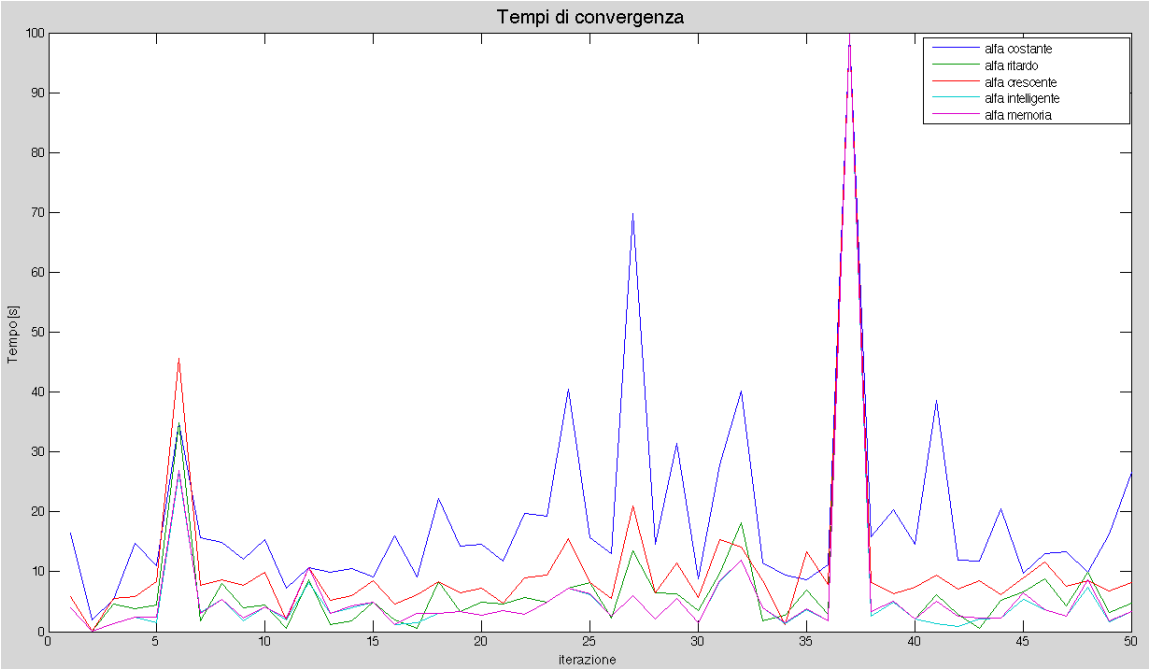
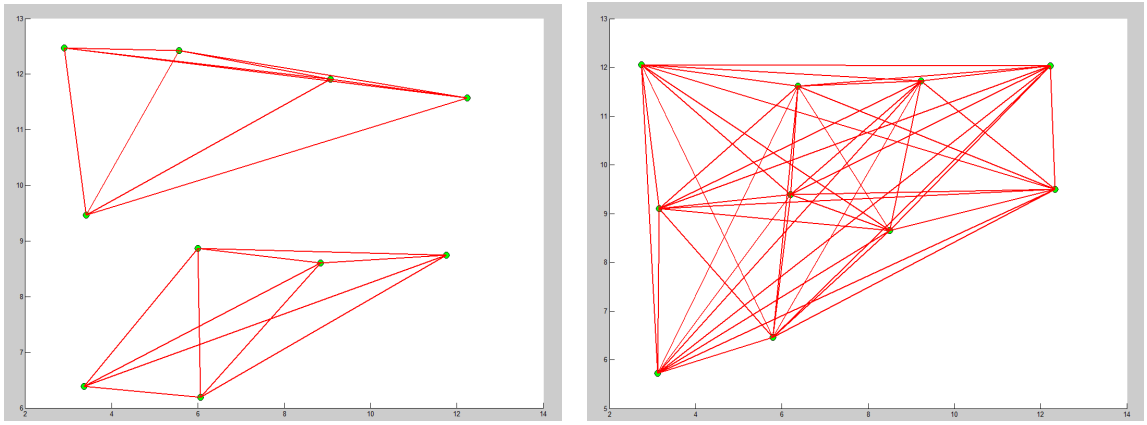


Figura 5.3: Sim. 2: Tempi di convergenza per i 5 metodi del Cap.3 in un ciclo di 50 simulazioni.

termine integrale	costante	in ritardo	a rampa	intelligente	con memoria
Media	18.8564	7.4757	10.6470	5.9431	6.2817
Dev. Std.	0.3723e-03	0.0558e-03	0.0573e-03	0.0030e-03	0.0067e-03

Tabella 5.2: Sim. 2: Medie e Deviazioni Standard per i 5 metodi del Cap.3 in un ciclo di 50 simulazioni.

Per dare un'ulteriore conferma della non robustezza dei metodi "termine integrativo applicato in ritardo" e "crescita a rampa di α", prendiamo in esame una situazione in cui il grafo di comunicazione tra i nodi viene modificato mentre l'algoritmo di controllo è già in azione: in particolare, in un certo istante di tempo, emergono nuovi rami del grafo.
In questa simulazione per i primi 5 secondi i nodi evolvono liberamente; all'istante $t = 5$ secondi si aziona l'algoritmo di sincronizzazione per i nodi disposti secondo il grafo di comunicazione di Fig. 5.4(a) (come se fossero 2 reti da 5 nodi indipendenti l'una dall'altra) e a $t = 12$ secondi la rete si modifica, adottando il grafo di comunicazione completamente connesso con 10 nodi di Fig. 5.4(b).



(a) (b)

Figura 5.4: (a) Grafo iniziale e (b) Grafo finale.

In questo modo (Fig. 5.5), i metodi adattivi riescono a variare il proprio guadagno nonostante il dinamismo della rete, assicurando una buona stabilità e buone prestazioni. Al contrario, i metodi non robusti assicureranno buone prestazioni all'attivazione del controllo ($t = 5$ secondi) ma non saranno in grado di gestire la variazione della rete di comunicazione, avendo un controllo (e quindi una dinamica) pari a quello della semplice α costante.

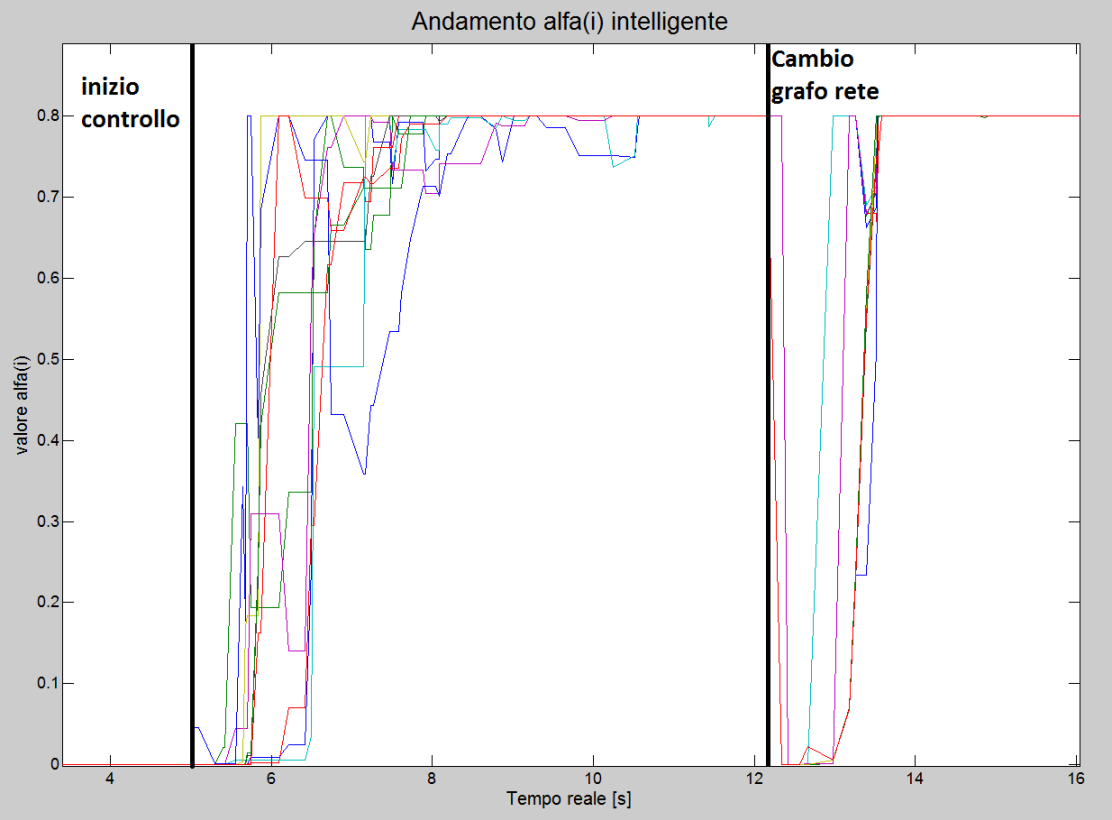


Figura 5.5: Andamento α_i prima e dopo l'aggiunta di archi di comunicazione.

Dalla Fig. 5.6 si può infatti notare, grazie al logaritmo della norma dell'errore, come l'applicazione di α in ritardo e l'incremento di α non abbiano ormai più nessun effetto mentre i metodi robusti risultano più prestazionali. Ugualmente, dal confronto tra le Fig. 5.7(a) e 5.7(b), si nota come il metodo dell'applicazione di α in ritardo sia ormai inefficace, in quanto da $t = 7$ secondi in avanti si ha un semplice controllo PI con guadagno α costante, che non assicura buone prestazioni a regime causa offset elevati.

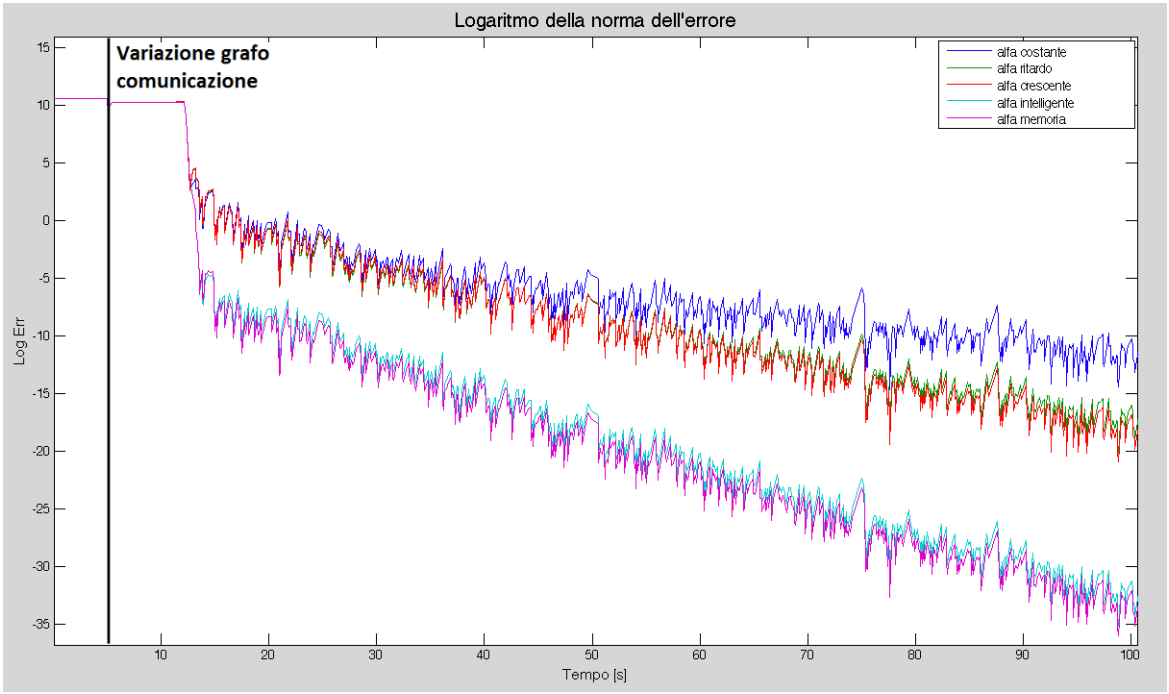
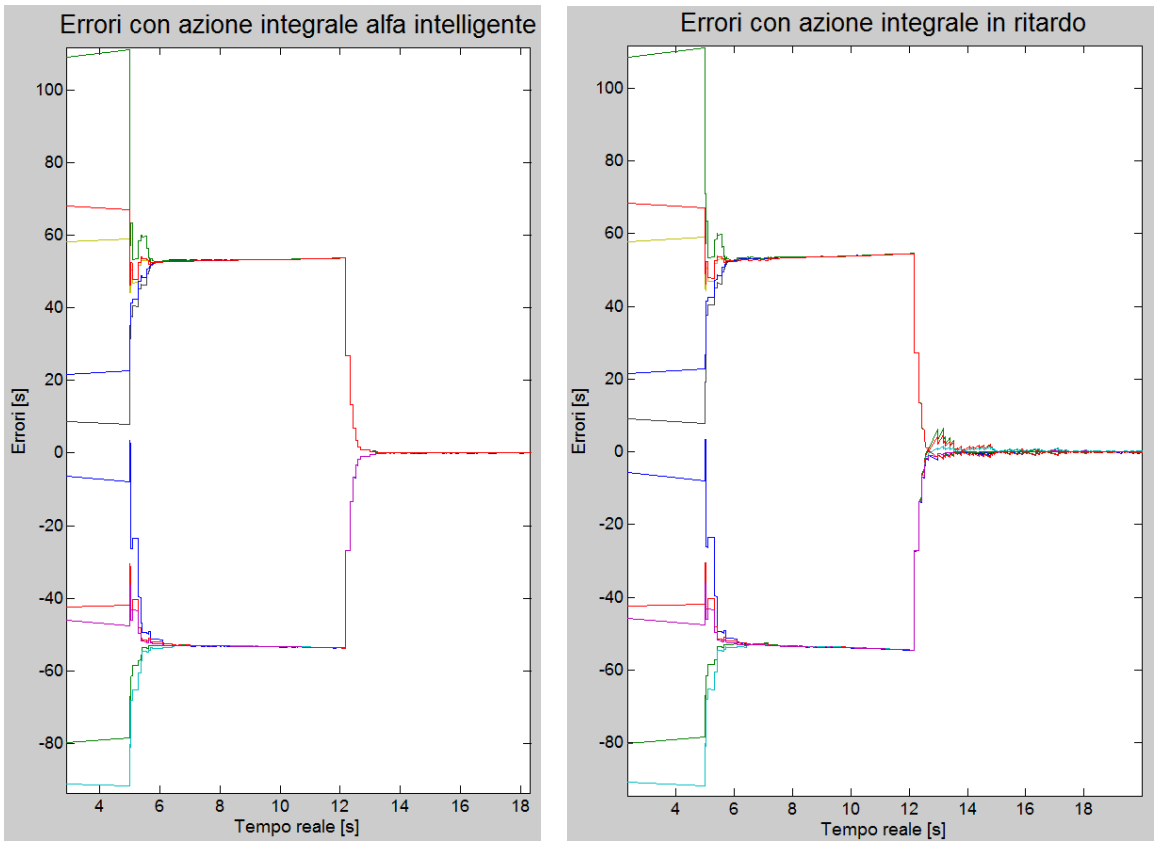


Figura 5.6: Logaritmo della norma dell'errore di sincronizzazione prima e dopo l'aggiunta di archi di comunicazione, confrontando per i metodi riportati in legenda.



(a) (b)

Figura 5.7: Andamento dell'errore di sincronizzazione, prima e dopo l'aggiunta di archi di comunicazione, usando (a) metodo con termine α intelligente e (b) metodo con termine integrativo applicato in ritardo.

Per quanto riguarda i metodi che si sono rivelati più soddisfacenti sotto il punto di vista della velocità di convergenza al consensus, robustezza e stabilità, è interessante poter definire alcune classiche applicazioni nelle quali questi controllori potrebbero offrire un buon metodo risolutivo al problema di sincronizzazione.
Come già evidenziato a tempo debito nella sezione 3.3, il metodo ad α "intelligente" può essere integrato esclusivamente in quelle applicazioni elettroniche dotate di hardware in grado di svolgere calcoli ad un sufficiente livello di complessità. Infatti questo metodo, proponendo di utilizzare una relazione non lineare (in particolare esponenziale) in funzione alla distanza temporale tra nodi, non risulta di facile implementazione su microprocessori di bassa capacità computazionale, come ad esempio sulla rete *SensLab* introdotta nella sezione 2.3 dove urge un algoritmo semplice. In queste circostanze l'implementazione in linguaggio macchina di un'integrazione o addirittura di una più semplice divisione richiede artifizi al fine di ridurre al minimo la complessità di calcolo; per esempio anche l'apparente semplice regolazione dello skew può rappresentare un limite nelle applicazioni reali. Viceversa, nel caso in cui i dispositivi in esame abbiano maggiori capacità computazionali e spazi di memoria sarebbe possibile addirittura migliorare l'algoritmo in vari modi.

Per il metodo con termine integrativo dotato di memoria si può giungere alle stesse conclusioni: elevata difficoltà di implementazione sia per l'utilizzo intrinseco di funzioni non lineari, sia per la necessità di dover disporre ogni elemento della rete con un'unità atta alla memorizzazione di dati.

Una nota di merito bisogna comunque rivolgerla ai più semplici metodi che propongono rispettivamente il termine integrale applicato in ritardo e il termine integrale crescente. Questi algoritmi, seppur non capaci di offrire le stesse caratteristiche dei due metodi sopra considerati, sono in grado di fornire un buon compromesso tra semplicità di implementazione e prestazioni, che potrebbero essere sufficienti per alcune applicazioni che non richiedono alte velocità di sincronizzazione. Bisogna ricordare però la scarsa robustezza di tali metodi: nella loro applicazione pratica risulta fondamentale la conoscenza delle condizioni iniziali al fine di assicurarne la stabilità. Avendo infatti a disposizione per esempio i valori (approssimativi) di offset iniziali, i tempi di aggiornamento, la numerosità dei nodi, lo scostamento dalla frequenza nominale, la rete di comunicazione ecc, si potrebbero tarare in maniera migliore anche i metodi non intrinsecamente robusti, riuscendo quindi ad avere robustezza per il particolare campo di applicazione in uso.

Nei casi in cui il metodo α "intelligente" non fosse utilizzabile (ad esempio per limitate capacità di calcolo del microprocessore) ma volendo però sfruttare gli stessi pregi del metodo anche in situazioni in cui è necessario, si potrebbe provare a linearizzare la funzione esponenziale, ottima per il caso in questione, mediante per esempio una successione di segmenti.
In alternativa risulterebbe necessario dover rendere più robusti i metodi a parametri fissati. Un metodo che viene proposto consiste nel far variare il ritardo dell'azione integrale in base all'offset tra il nodo ricevente e quello comunicante. Si potrebbe per esempio far usare l'azione integrale, in maniera indipendente per ogni singolo nodo, solo se l'offset è minore di una certa soglia. Non si avranno di certo le prestazioni offerte dal metodo α "intelligente", ma si riuscirebbe ad avere sicuramente più robustezza, come illustrato in Fig. 5.8; è stata rifatta quest'ultima simulazione usando (per ogni nodo) la parte integrale del controllo solo se l'offset è inferiore ad una soglia di 0.4 secondi.

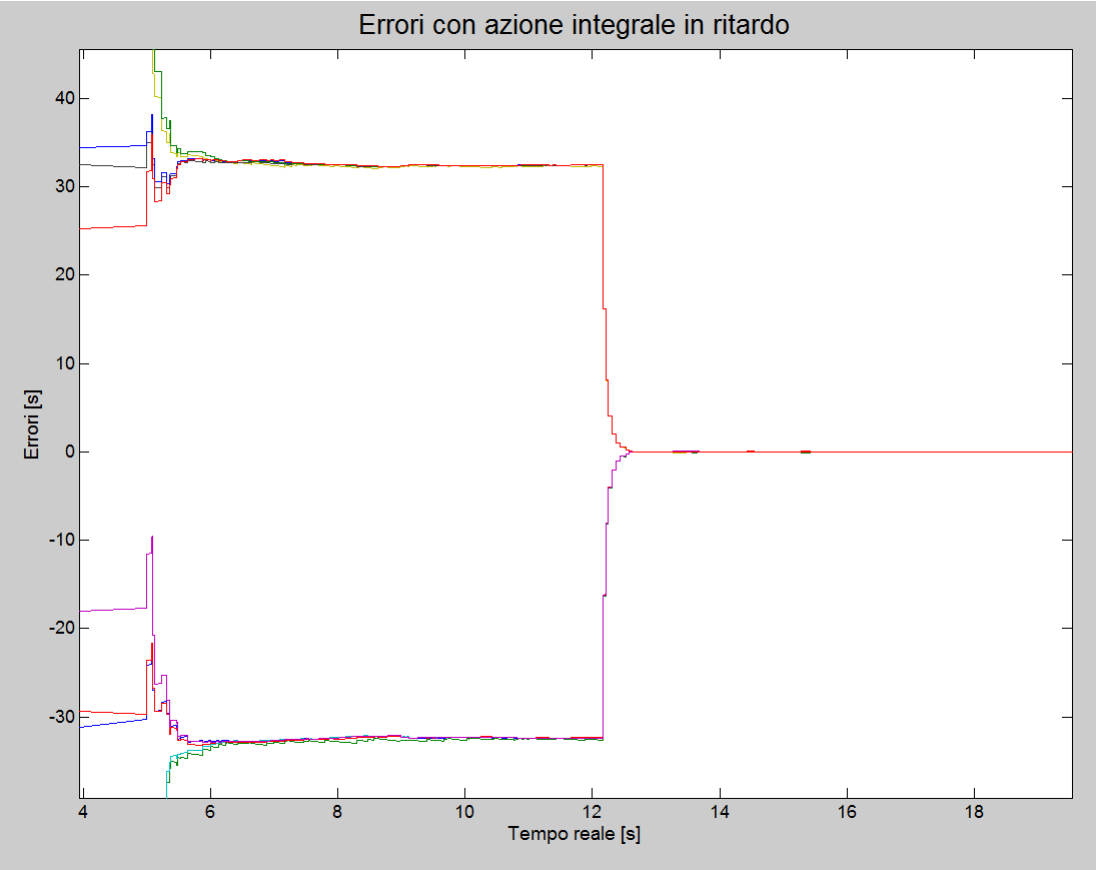


Figura 5.8: Proposta di ottimizzazione del metodo termine integrale applicato in ritardo.

Questa idea si potrebbe facilmente applicare anche al metodo α crescente, facendo aumentare il guadagno solo se l'offset è un valore "piccolo" o altrimenti azzerando il termine integrale se l'offset è "grande". Si potrebbe altrimenti regolare l'incremento del guadagno integrale con una semplice funzione dell'offset (magari mediante una tabella di dati reimpostati).
Queste alternative non sono state teoricamente approfondite in questo elaborato (sono state simulate in maniera quantitativa) però risultano metodologie algoritmicamente semplici e quindi di vasta applicazione. Si porpongono quindi questi argomenti come base per eventuali sviluppi futuri.

Bibliografia

[1] L. Schenato, Appunti del corso di Progettazione di Sistemi di Controllo, *a.a. 2012/13.*

[2] R. Carli, E. D'Elia, S. Zampieri, *A PI controller based on asymmetric gossip communication for clocks synchronization in wireless sensor networks, University of Padua, Department of Information Engineering, March 2011.*

[3] F. Fiorentin, "Implementazione di sincronizzazione temporale distribuita in reti di sensori wireless (italian)," *Master's thesis, University of Padova, Department of Information Engineering, Dicembre 2008.*

[4] Edoardo D'Elia, Analysis of Clocks Synchronization Algorithms in Wireless Sensor Networks, *Master's thesis, University of Padova, Department of Information Engineering, March 2011.*

[5] R. Carli, A. Chiuso, L. Schenato, S. Zampieri, A PI Consensus Controller for Networked Clocks Synchronization, *in IFAC, Seul, Corea, Jul. 2008.*

[6] S. Bolognani, R. Carli, S. Zampieri, A PI consensus controller with gossip communication for clock synchronization in wireless sensors networks, *2009.*

[7] L. Schenato, F. Fiorentin, Average timesync: A consensus-based protocol for time synchronization in wireless sensor networks, *in Proceedings of 1st IFAC Workshop on Estimation and Control of Networked Systems (NecSys09), September 2009.*

[8] http://www.senslab.info/.

[9] http://wiki.senslab.info/.

[10] D. Ciscato, Appunti di Controllo Digitale, *a.a. 2009/10.*

Ringraziamenti

Si ringrazia espressamente il Prof. Ruggero Carli per la disponibilità, la gentilezza e l'aiuto offertoci in merito al quale è stato possibile concludere, in modo per noi estremamente soddisfacentemente, il lavoro qui proposto.
Si vuole ringraziare anche il Prof. Saverio Bolognani per la chiara introduzione e per il materiale concessoci relativo all'utilizzo della piattaforma *SensLab*.
Si ringrazia inoltre il Prof. Luca Schenato per averci offerto la possibilità di intraprendere un progetto formalmente interessante sia dal punto di vista personale sia da quello applicativo, oltre alla possibilità di svolgere un elaborato lavoro di squadra, elemento importante per l'imminente futuro in ambito lavorativo.