UNIVERSITÀ DEGLI STUDI DI PADOVA

Facoltà di Ingegneria
Corso di Laurea in Ingegneria dell'Automazione

Corso di Progettazione di Sistemi di Controllo
a.a. 2010/11

Identificazione termodinamica di un edificio

Andrea Barazzuol 601399 barazzu1@dei.unipd.it

Markus Ausserer 621889 ausserer@dei.unipd.it

Riccardo Alberton 621396 alberton2@dei.unipd.it

Padova, 25 Febbraio 2011

Indice

1 Introduzione

La sempre maggior attenzione all'ambiente e all'utilizzo sostenibile delle risorse hanno portato una forte attenzione sulle problematiche di riduzione dei consumi di energia primaria. Più di un terzo dei consumi europei è dovuto al settore di climatizzazione degli edifici, per cui si rende dunque necessaria da una parte una costruzione attenta alla coibentazione degli edifici e dall'altra un controllo termico che riesca ad introdurre maggior efficienza.

Essendo che l'aggiornamento del parco immobiliare ha un ciclo temporale lungo i risultati a breve termine posso essere garantiti da sistemi di regolazione che utilizzano modelli dettagliati dell'edificio. Questo comporta la necessità di dover ricostruire il modello termico senza conoscere alcun dato di progetto (fondamentale nell'ambito di edifici esistenti il cui recupero di informazioni tecniche operazione ardua se non impossibile a causa di operazioni invasive).

Il metodo di identificazione degli edifici tramite una rete di sensori, è da considerarsi valido non solo per gli edifici vecchi di cui non si possiedo informazioni a sufficienza per usare i modelli teorici, ma anche per i nuovi edifici come sistema di aggiornamento del modello, aggiornamento che potrebbe essere reso necessario per un infinità di motivi che vanno dalle modifiche edilizie, al cambio di prestazione dei materiali, al degrado della struttura ecc.

Risulta naturale che vaste reti di sensori richiedano sia un forte dispendio economico che di tempo per l'installazione, quindi risulterebbe di forte interesse poter avere a disposizione più dati di quelli che realmente si posso raccogliere con la rete di sensori, il cui scopo sarebbe quella di minimizzarla. Per questo l'idea sarebbe quella di avere una rete fissa minima da cui poi generare ulteriori stati per una migliore controllo.

La relazione si propone di valutare i nuovi algoritmi di SSEH, GLAR e SS-GLAR la cui forza è anche l'auto selezione dei sensori da utilizzare; usando dei principi di proiezioni geometriche vengono scelti gli ingressi maggiormente correlati con il segnale. I dati utilizzati sono stati i dati del progetto SIMEA (Sistema Integrato/distribuito di Monitoraggio Energetico ed Ambientale).

La relazione si articola in una prima parte in cui si spiegano gli algoritmi utilizzati, mentre nella seconda si analizzano le prestazioni degli stessi confrontandoli anche con gli algoritmi più classici come la PEM.

La forza di questi algoritmi è la capacità di identificare modelli buoni anche quando il rapporto dati/parametri scende a valori in cui l'identificazione classica non riscuote grossi risultati.

2 Stato dell'arte

L'identificazione black-box è ampiamente utilizzata per ricostruire la dinamica di modelli da un insieme finito di dati ingresso-uscita. In questo scenario una situazione particolare che si verifica in svariati ambiti (ad esempio in ingegneria chimica, in econometria, nella visione computazionale, e così via) è quella in cui si richiede l'identificazione di sistemi di grandi dimensioni in cui entrano in gioco un gran numero di variabili. In un contesto simile un punto cardine delle procedure di identificazione dovrebbe essere quello di favorire la sparsità dei risultati, ovvero di estrarre dal gran numero di sottosistemi presenti nel sistema principale solo il sottoinsieme che influenza l'uscita del sistema in maniera più significativa.
Il fatto che ogni variabile possa dipendere solo da un sottoinsieme di altre variabili si esprime nel caso Gaussiano statico con delle condizioni di indipendenza condizionata fra sottoinsiemi di variabili. Nel caso dinamico, cioè quando i dati osservati sono traiettorie di processi stocastici (possibilmente stazionari) le condizioni di indipendenza condizionale si esprimono nel fatto che la predizione dell'andamento di una variabile (che verrà chiamata uscita) richieda solo il passato di qualche altra variabile (che verranno chiamate ingressi) più eventualmente il proprio. Più avanti ciò verrà rappresentato con un grafo in cui i nodi sono le variabili, gli archi diretti sono le funzioni di trasferimento non nulle e i cappi (archi che si chiudono sullo stesso nodo da cui partono) codificano la dipendenza della variabile dal proprio passato. In generale sia il sistema dinamico sia la struttura che rappresenta il legame fra le variabili dovranno essere estratti dai dati. In questo contesto di identificazione gli usuali metodi di minimizzazione dell'errore di predizione (PEM) possono diventare computazionalmente problematici poichè la teoria sottostante ai metodi di stima dell'ordine del modello (ad esempio AICC e BIC) inizia a vacillare. Questo problema insieme al problema legato alla presenza di non-convessità è stato risolto con l'introduzione degli algoritmi ai sottospazi che invece di fornire la stima utilizzando metodi iterativi di ottimizzazione non lineare (come in PEM), si basano su degli strumenti efficienti e robusti derivati dall'algebra lineare come la decomposizione QR e SVD. Tuttavia anche per questi metodi la presenza di un gran numero di parametri da stimare rispetto al numero di dati disponibili resta un problema. In questa situazione dunque è possibile che sia gli algoritmi parametrici tradizionali che quelli ai sottospazi possano fornire risultati non soddisfacenti.
Oggigiorno sono stati introdotti dei nuovi approcci parametrici (come LAR e LASSO) che inducono sparsità nel risultato sfruttando la norma l_1. Un siffatto termine peso porta con sè la cosiddetta proprietà di bi-separazione, cioè la preferenza di soluzioni con molti termini nulli a spesa di pochi termini ma di modulo maggiore. Sono inoltre stati ampliati con delle versioni a gruppi (GLars e GLasso) in cui interi gruppi di termini sono posti a zeri ottenendo così una sparsità in termini di funzioni di trasferimento piuttosto che di singoli termini. Tuttavia la maggior parte del lavoro in questo ambito è stato svolto in uno scenario statico a discapito del caso

dinamico.
Gli algoritmi che verranno testati maggiormente in questo lavoro adottano un punto di vista Bayesiano e prendono come punto di partenza un nuovo paradigma sviluppato in [9] che si basa sulla stima non parametrica delle risposte impulsive. In questo modo invece di dover postulare una struttura finito dimensionale per le funzioni di trasferimento del sistema (con ordine quindi da stimare) si cerca la risposta impulsiva all'interno di uno spazio infinito dimensionale. Il mal condizionamento intrinseco a un problema così formulato viene risolto utilizzando dei metodi di regolarizzazione Bayesiana. Più nel dettaglio si può dire che le risposte impulsive sono modellate come dei processi Gaussiani la cui autocovarianza viene chiamata *stable spline kernel* ed include il vincolo di BIBO stabilità del risultato.

3 Pem: Prediction Error Methods

In questo capitolo viene spiegato brevemente il principio di funzionamento dei metodi a minimizzazione dell'errore di predizione.
Dato un modello $M(\theta)$ appartenente a una certa classe parametrica $\mathcal{M} \equiv \{M(\theta); \theta \in \Theta\}$ e dato una sequenza di N misure dei segnali d'ingresso e d'uscita

$$u^N := \{u(t); t = 1, \ldots, N\} \qquad y^N := \{y(t); t = 1, \ldots, N\}$$

si procede nel modo seguente:

1. Per un qualche valore di θ fissato si costruisce il miglior (secondo qualche criterio) predittore all'istante $t-1$ dell'uscita $y(t)$. Questo predittore è una funzione dei dati passati che viene denotata col simbolo $\hat{M}(\theta)$:

   $$\hat{M}(\theta) : (y^{t-1}, u^{t-1}) \to \hat{y}_\theta(t|t-1).$$

   La predizione $\hat{y}_\theta(t|t-1)$ si può pensare come funzione dei dati passati e quindi come una funzione aleatoria che viene indicata usando il simbolo in grassetto $\hat{\mathbf{y}}_\theta(t|t-1)$.

2. Si formano gli errori di predizione

   $$\varepsilon_\theta(t) := y(t) - \hat{y}_\theta(t) \qquad \text{per } t = 1, \ldots, N.$$

   Analogamente a quanto detto nel caso del predittore anche gli errori di predizione possono essere visti come quantità aleatorie e vengono quindi indicate col simbolo in grassetto $\boldsymbol{\varepsilon}_\theta(t)$.

3. Si minimizza rispetto a θ una cifra di merito che descrive quanto bene il modello predice il dato successivo. Ad esempio si minimizza l'errore quadratico medio di predizione

   $$V_N(\theta) := \frac{1}{N} \sum_{t=1}^{N} \varepsilon_\theta(t)^2$$

   o più in generale una media degli errori quadratici di predizione pesati con una cifra di merito non negativa $\beta(N,t) > 0$,

   $$V_N(\theta) := \frac{1}{N} \sum_{t=1}^{N} \beta(N,t)\varepsilon_\theta(t)^2$$

   che per N piccoli dà peso minore agli errori di predizione compiuti nella fase iniziale dell'algoritmo quando l'influenza di condizioni iniziali stimate in modo approssimativo è più deleteria. Per N grande $\beta(N,t)$ tende a essere inutile e

quindi deve tendere a uno.
In ogni caso si ricava la stima si θ dalla minimizzazione della cifra di merito

$$\hat{\theta}_N := Arg \min_{\theta} V_N(\theta)$$

che è la stima PEM del parametro del modello. Lo stimatore $\hat{\theta}_N$ viene chiamato stimatore PEM del parametro θ.

4. Infine si prende come stima della varianza dell'innovazione $\lambda^2 = var\{\mathbf{e}(t)\}$ l'errore quadratico residuo

$$\hat{\lambda}^2 := V_N(\hat{\theta}_N).$$

Per quanto questa procedura sembra intuitivamente sensata, l'unica giustificazione per la sua adozione nei procedimenti di identificazione sta nelle sue proprietà statistiche. Per avere una trattazione completa delle proprietà statistiche dello stimatore PEM si consulti [1].

Finora si è visto come ricavare lo stimatore PEM nel caso si ha un parametro θ fissato. Poiché in generale però non si ha alcuna informazioni sulla struttura del sistema vero serve uno strumento per scegliere la classe parametrica $\mathcal{M}$ alla quale appartiene il modello identificato $M(\theta)$. Dato un set di misure e diverse classi di modelli con ordini p plausibili si calcolano degli indici che danno la possibilità di scegliere la classe che promette di ottenere un modello identificato con buone prestazioni. Questi indici sono sostanzialmente una generalizzazione del criterio FPE (Final Prediction Error, vedere [1] per la trattazione completa). Nelle simulazioni svolte per questo elaborato sono stati usati l'AICC (Akaike Information Criterion with a second-order Correction) e il BIC (Bayesian Information Criterion) definito dalle seguenti formule:

$$AICC = N \cdot \ln \frac{rss}{N} + N \cdot \frac{N+p}{N-p-2} \tag{3.1}$$

$$BIC = N \cdot \ln \frac{rss}{N} + p \cdot \ln(N) \tag{3.2}$$

dove rss (Residual Sum of Squares) è definito da $rss = \sum_{t=1}^{N} \varepsilon_\theta(t)^2$, mentre N è il numero di misure e p è l'ordine del parametro θ del modello.
Nel progetto in esame è stata implementata una routine nella quale viene considerato la classe dei modelli ARMAX. In un ciclo ad ogni passo viene aumentato l'ordine del parametro θ e vengono calcolati i valori dei due indici. Terminato il ciclo si controlla in relazione a quale ordine gli indici hanno raggiunto i valori minimi. Viene quindi rifatta l'identificazione PEM usando i rispettivi ordini ritenuti ottimi ottenendo due modelli che spesso si verificano aver ordine differente. Si noti però che per i due criteri considerati all'aumentare di N la probabilità che la scelta dell'ordine sia sbagliato tende a zero (e quindi si dice che la selezione dell'ordine è consistente).

4 Metodi di identificazione ai sottospazi

I metodi di identificazione a sottospazi sono particolarmente adatti per l'identificazione di sistemi MISO (o MIMO) di grandi dimensioni per la loro efficienza computazionale. Vengono implementati utilizzando strumenti derivati dall'algebra lineare (per effettuare opportune proiezioni ortogonali e oblique) come la decomposizione QR e SVD di cui esistono implementazioni molto efficienti. Si noti che in questo modo si evita l'utilizzo di minimizzazioni ricorsive e quindi si evitano i problemi di convergenza che esse comportano (e che possono capitare con i metodi PEM). Nella prossima sezione verrà analizzato il metodo ai sottospazi

4.1 Il metodo N4SID

Date le osservazioni $\{y_1, \ldots, y_T, u_1, \ldots, u_T\}$, $y_t \in \mathbb{R}^p$, $u_t \in \mathbb{R}^m$ il problema è quello di costruire il modello:

$$\begin{cases} x(t+1) &= Ax(t) + Bu(t) + Ke(t) \\ y(t) &= Cx(t) + e(t) \end{cases} \tag{4.1}$$

che descriva i dati (con $e(t)$ rumore bianco).

A partire dal processo stostico $\mathbf{y}(t)$ si scrivono i dati nella forma $\{y_t, \ldots, y_{t+N-1}\} = \mathbf{Y}_t$ per creare una corrispondenza biunivoca (al crescere del numero N di dati) tra lo spazio astratto di variabili aleatorie $span\{y(t), \ldots, y(s)\}$ e lo spazio generato dalle righe $\mathbf{Y}_t$ della matrice di Henkel diagonale a blocchi:

$$Y_{[t,s]} = \begin{bmatrix} y_t & y_{t+1} & \cdots & y_{t+N-1} \\ y_{t+1} & y_{t+2} & \cdots & y_{t+N} \\ \vdots & \vdots & \ddots & \vdots \\ y_s & y_{s+1} & \cdots & y_{s+N-1} \end{bmatrix} \tag{4.2}$$

di dimensione $Y_{[t,s]} \in \mathbb{R}^{p(s-t)xN}$ e da questa si definiscono le matrici delle uscite passate e future

$$Y_{[0,2t-1]} = \begin{bmatrix} Y_p \\ Y_f \end{bmatrix} = \begin{bmatrix} Y_{[0,t-1]} \\ Y_{[t,2t-1]} \end{bmatrix} \tag{4.3}$$

e si svolgono delle operazioni analoghe con i dati d'ingresso $\mathbf{u}(t)$. A questo punto si può costruire una base per lo spazio di stato X_t attraverso una proiezione adatta delle sequenze dei dati e successivamente da questa base si risolve nel senso dei minimi quadrati il sistema:

$$\begin{cases} \hat{X}_{t+1} &\simeq A\hat{X}_t + B\hat{U}_t + KE_t \\ Y_t &\simeq C\hat{X}_t + E_t \end{cases} \tag{4.4}$$

dove $\hat{E}_t := Y_t - \hat{C}X_t$.
Si definisce poi la matrice di osservabilità del sistema come:

$$\Gamma := \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^f \end{bmatrix} \tag{4.5}$$

e matrici triangolari inferiori a blocchi di Toeplitz con i parametri di Markov del sistema:

$$H_d = \begin{bmatrix} 0 & 0 & \dots & 0 \\ CB & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{f-1}B & CA^{f-2}B & \dots & 0 \end{bmatrix} \quad , \tag{4.6}$$

$$H_s = \begin{bmatrix} I & 0 & \dots & 0 \\ CK & I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{f-1} & CA^{f-2}K & \dots & I \end{bmatrix} \quad . \tag{4.7}$$

Utilizzando ora la relazione

$$\mathbf{Y}_{[p,p+f]} = \Gamma \mathbf{X}_p + H_d \mathbf{U}_{[p,p+f]} + H_s \mathbf{E}_{[p,p+f]} \tag{4.8}$$

si può effettuare la proiezione obliqua [1] (mediante decomposizione QR)

$$\Gamma \mathbf{X}_p \simeq \hat{\mathbf{Y}}_{[p,p+f]} := E_{||\mathbf{U}_{[p,p+f]}} \left[\mathbf{Y}_{[p,p+f]} | \mathbf{Z}_{[0,p)} \right] . \tag{4.10}$$

[1] La proiezione obliqua é la scomposizione in due sottospazi tra loro non ortogonali. Si denota come $\hat{E}_{||B}[\cdot|C_t]$, ossia la proiezione lungo lo spazio generato delle righe di B sullo spazio generato dalle righe di C_t. La formula estesa é:

$$\hat{E}_{\| B_t}[A_t|C_t] = \hat{\Sigma}_{ac|b} \hat{\Sigma}^{-1}_{cc|b} C_t \tag{4.9}$$

Dove sotto le ipotesi di ergodicitá le matrici di covarianza campionaria (condizionata e non) sono: $\hat{\Sigma}_{ab} := \frac{1}{N} \sum_{i=0}^{N-1} a_{t+i} b_{t+i}^\top$ e $\hat{\Sigma}_{ab|c} := \hat{\Sigma}_{ab} - \hat{\Sigma}_{ac} \hat{\Sigma}_{cc}^{-1} \hat{\Sigma}_{cb}$

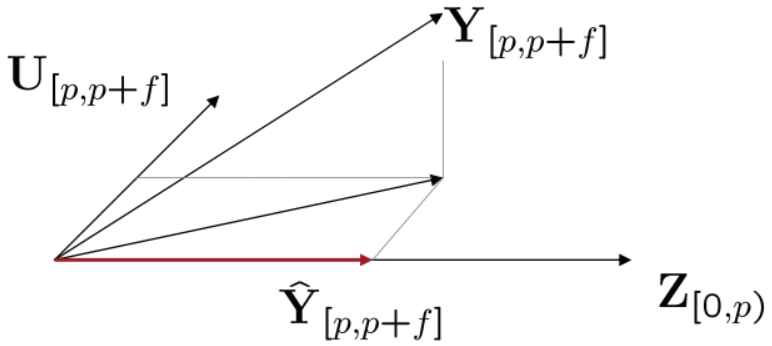


Figura 4.1: Rappresentazione geometrica della proiezione obliqua effettutata in (4.10)

Per stimare Γ (e se necessario $\mathbf{X}_p$) si utilizza una decomposizione 'SVD pesata'

$$U_n S_n V_n^T \simeq W_r \hat{\mathbf{Y}}_{[p,p+f]} W_c \tag{4.11}$$

e si ottiene

$$\hat{\Gamma} := W_r^{-1} U_n S_n^{1/2} \qquad \hat{\mathbf{X}}_p := \hat{\Gamma}_r^{-L} \hat{\mathbf{Y}}_{[p,p+f]} \tag{4.12}$$

5 LARS

5.1 L'algoritmo

Il LARS (Least Angle Regression) è una versione semplificata della procedura Stagewise la quale usa una semplice formula matematica per accelerare il calcolo computazionale. In soli m passi si giunge all'insieme delle soluzioni, dove m è il numero delle covarianze.

Come nel più classico algoritmo Forward Selection si parte con tutti i coefficienti della stima β uguali a zero e si trova il predittore x_{j_1} maggiormente correlato con la risposta. Una volta trovato la direzione scelta è quella di x_{j_1} e viene mantenuta finchè non subentra un altro predittore x_{j_2} che ha maggior correlazione con il residuo. Da qui in poi il LARS procede lungo una nuova direzione equiangolare rispetto a x_{j_1} e x_{j_2}. Si percorre tale direzione finché un nuovo predittore x_{j3} diventa il più correlato, a questo punto la direzione che viene presa è equiangolare rispetto a x_{j_1}, x_{j_2} e x_{j_3}. Ovviamente si procede con questa iterazione di k passi di volta in volta avanzando lungo una direzione equiangolare ai predittori precedentemente individuati e aggiornando la direzione nel momento in cui si trova un predittore maggiormente correlato con il residuo.

Si dà ora una veloce descrizione geometrica di questo processo. Con il LARS si cerca di dare una stima di $\widehat{\mu} = X\widehat{\beta}$ in una successione di passi, aggiungendo ad ogni passaggio una covarianza del modello, in maniera che dopo k passi il vettore $\widehat{\beta}_j$ abbia k componenti non nulle.

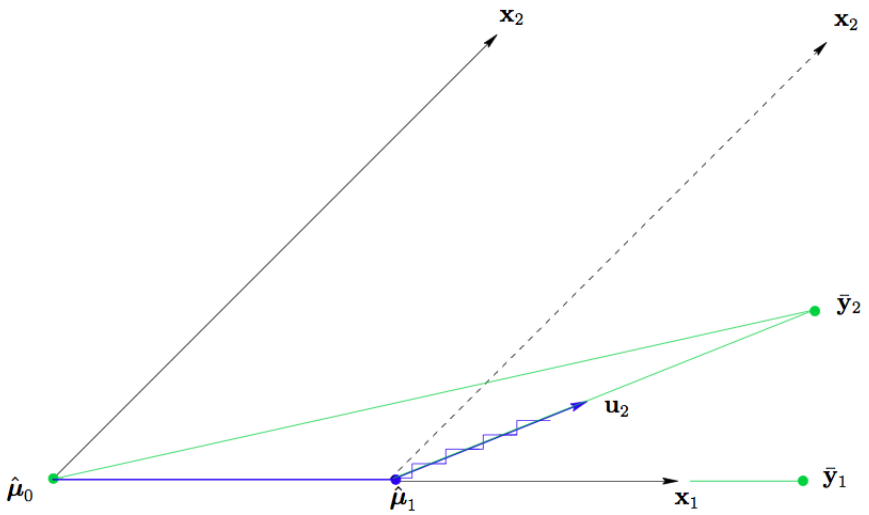


Figura 5.1

In figura 5.1 viene illustrato il caso con due covarianze $\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2)$ cioè con m=2. In questo caso la correlazione corrente diventa:

$$\mathbf{c}(\hat{\mu}) = X'(\mathbf{y} - \hat{\mu}) = X'(\bar{\mathbf{y}}_\mathbf{2} - \hat{\mu}) \tag{5.1}$$

dimostrando una dipendenza solo da $\bar{\mathbf{y}}_2$ nello spazio lineare $\mathcal{L}(X)$ generato da $\mathbf{x}_1$ e $\mathbf{x}_2$. Come precedentemente spiegato si parte con $\hat{\mu}_0 = \mathbf{0}$. In figura 5.1 si ha che $\bar{\mathbf{y}}_2 - \hat{\mu}_0$ produce un angolo più piccolo con $\mathbf{x}_1$ che con $\mathbf{x}_2$ ovvero si ha $c_1(\hat{\mu}_0) > c_2(\hat{\mu}_0)$; per cui il LARS aumenta $\widehat{\mu}_0$ nella direzione di $\mathbf{x}_1$ ottenendo:

$$\hat{\mu}_\mathbf{1} = \hat{\mu}_\mathbf{0} + \widehat{\gamma}_1 \mathbf{x}_1. \tag{5.2}$$

A questo punto l'algoritmo Stagewise sceglierebbe $\widehat{\gamma}_1$ uguale ad un qualche valore ϵ (molto piccolo) per poi ripetere il processo molte volte. Il classico Forward Selection prenderebbe invece $\widehat{\gamma}_1$ sufficientemente grande per rendere $\hat{\mu}_1$ equivalente a $\bar{\mathbf{y}}_1$, la proiezione di $\mathbf{y}$ su $\mathcal{L}(\mathbf{x}_1)$. Invece LARS usa un valore intermedio di $\widehat{\gamma}_1$, il valore che rende $\bar{\mathbf{y}}_2 - \hat{\mu}$ ugualmente correlato con $\mathbf{x}_1$ e $\mathbf{x}_2$, in maniera che $\bar{\mathbf{y}}_2 - \hat{\mu}_1$ sia la retta secante l'angolo compreso fra $\mathbf{x}_1$ e $\mathbf{x}_2$, ottenendo $c_1(\hat{\mu}_1) = c_2(\hat{\mu}_1)$.

Sia $\mathbf{u}_2$ il versore della bisettrice, il nuovo passo del LARS sarà:

$$\hat{\mu}_\mathbf{2} = \hat{\mu}_\mathbf{1} + \widehat{\gamma}_2 \mathbf{u}_2, \tag{5.3}$$

con $\widehat{\gamma}_2$ scelta per rendere $\widehat{\mu}_2 = \bar{\mathbf{y}}_2$ nel caso di m=2. Si noti che con più di 2 covarianze $\widehat{\gamma}_2$ sarebbe più piccola in conformità con un altro cambio di direzione come illustrato in figura 5.2.

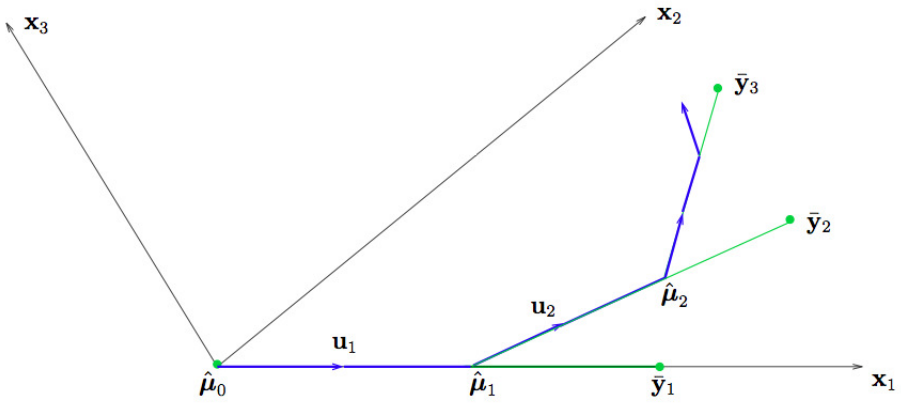


Figura 5.2

Da questo processo si evince che il calcolo teorico di $\widehat{\gamma}_1$, $\widehat{\gamma}_2$, ..., $\widehat{\gamma}_m$ sia molto più veloce per il LARS rispetto al calcolo con l'algoritmo Stagewise che avanza solo per piccoli passi.

Si presume ora di assumere i vettori covarianze $\mathbf{x}_1$, $\mathbf{x}_2$, ... $\mathbf{x}_m$ linearmente indipendenti. Per $\mathcal{A}$, un sottoinsieme degli indici $\{1, 2, \ldots, m\}$, si definisce la matrice:

$$\mathbf{X}_{\mathcal{A}} = (\dots s_j \mathbf{x}_j \dots)_{j \in \mathcal{A}} \tag{5.4}$$

dove la variabile segno s_j può assumere i valori ± 1. Si prende

$$\mathcal{G}_{\mathcal{A}} = X'_{\mathcal{A}} X_{\mathcal{A}} \quad \text{con} \quad \mathcal{A}_{\mathcal{A}} = (1'_{\mathcal{A}} \mathcal{G}_{\mathcal{A}} 1_{\mathcal{A}})^{-\frac{1}{2}} \tag{5.5}$$

dove $1_{\mathcal{A}}$ è un vettore con elementi unitari con lunghezza pari $|\mathcal{A}|$, la dimensione di $\mathcal{A}$.

Sia

$$\mathbf{u}_{\mathcal{A}} = X_{\mathcal{A}} w_{\mathcal{A}} \quad \text{con} \quad w_{\mathcal{A}} = \mathcal{A}_{\mathcal{A}} \mathcal{G}_{\mathcal{A}}{}^{-1} 1_{\mathcal{A}} \tag{5.6}$$

il vettore equiangolare ovvero il vettore unitario che rende gli angoli (minori di 90°) con le colonne della matrice $\mathcal{A}$ tutti uguali,

$$X'_{\mathcal{A}} \mathbf{u}_{\mathcal{A}} = \mathcal{A}_{\mathcal{A}} 1_{\mathcal{A}} \quad \text{e} \quad ||\mathbf{u}_{\mathcal{A}}||^2 = 1. \tag{5.7}$$

Si descrivono ora i passi seguiti dall'algoritmo LARS. Come in Stagewise si parte con $\widehat{\mu}_0 = \mathbf{0}$ e si costruisce $\widehat{\mu}$ iterativamente, cercando però di percorrere passi più grandi. Supponendo che $\widehat{\mu}_{\mathcal{A}}$ sia la stima corrente con

$$\hat{\mathbf{c}} = X'(\mathbf{y} - \hat{\mu}_{\mathcal{A}}) \tag{5.8}$$

il vettore delle correlazioni correnti come definito dalla generica $\widehat{\mathbf{c}} = \mathbf{c}(\widehat{\mu}) = X'(\mathbf{y} - \widehat{\mu})$. L'insieme attivo $\mathcal{A}$ è composto dagli indici delle covarianze che hanno la più grande correlazione corrente in valore assoluto, ovvero

$$\widehat{C} = \max_j |\hat{c}_j| \quad \text{e} \quad \mathcal{A} = \{j : |\widehat{c}_j| = \widehat{C}\}. \tag{5.9}$$

Prendendo

$$s_j = sign\{\widehat{c}_j\} \quad \text{con} \quad j \in \mathcal{A} \tag{5.10}$$

si calcola $X_{\mathcal{A}}$, $A_{\mathcal{A}}$ ed $\mathbf{u}_{\mathcal{A}}$, e si calcola il prodotto interno

$$\mathbf{a} \equiv X' \mathbf{u}_{\mathcal{A}}. \tag{5.11}$$

Al passo successivo l'algoritmo LARS aggiorna $\widehat{\mu}_{\mathcal{A}}$, definita come

$$\widehat{\mu}_{\mathcal{A}_+} = \widehat{\mu}_{\mathcal{A}} + \widehat{\gamma} \mathbf{u}_{\mathcal{A}} \tag{5.12}$$

dove

$$\widehat{\gamma} = \min_{j \in \mathcal{A}^c}{}^{+} \{\frac{\widehat{C} - \widehat{c}_j}{A_{\mathcal{A}} - a_j}, \frac{\widehat{C} + \widehat{c}_j}{A_{\mathcal{A}} + a_j}\} \tag{5.13}$$

con $= \min^+$ che indica che il minimo viene preso solo tra i componenti positivi delle scelte j in 5.13. Usando le formule 5.12 e 5.13 si definisce

$$\mu(\gamma) = \widehat{\mu}_{\mathcal{A}} + \gamma \mathbf{u}_{\mathcal{A}} \tag{5.14}$$

con $\gamma > 0$, in maniera che la correlazione corrente sia

$$c_j(\gamma) = \mathbf{x}_j'(\mathbf{y} - \mu(\gamma)) = \widehat{c}_j - \gamma a_j. \tag{5.15}$$

Per $j \in \mathcal{A}$ usando 5.7 5.9

$$|c_j(\gamma)| = \widehat{C} - \gamma A_{\mathcal{A}} \tag{5.16}$$

il che mostra come tutte le correlazioni correnti declinino nella stessa maniera. Per $j \in \mathcal{A}^c$ uguagliando l'equazione 5.15 con la 5.16, si ottiene che $c_j(\gamma)$ è uguale al massimo valore di $\gamma = (\widehat{C}_j - \widehat{c}_j)/(A_{\mathcal{A}} - a_j)$; alla stessa maniera per $-c_j(\gamma)$ la correlazione corrente per la covarianza inversa $-\mathbf{x}_j$ raggiunge il massimo per $(\widehat{C}_j + \widehat{c}_j)/(A_{\mathcal{A}} + a_j)$.

Quindi la $\widehat{\gamma}$ in 5.13 è il più piccolo valore positivo di γ tale che il nuovo indice $\widehat{j}$ venga aggiunto all'insieme attivo degli indici; $\widehat{j}$ è indice che minimizza la 5.13,

5.2 L'estensione a gruppi

Quando tutti i fattori ($Y = \sum_{j=1}^{J} X_j \beta_j + \varepsilon$) hanno lo stesso numero di variabili di input ($p_1 = \ldots = p_j$) una naturale estensione del LARS per la selezione dei coefficienti è la sua estensione a gruppi che mantiene la proprietà di linearità a tratti del percorso di soluzione. Si parte definendo l'angolo $\theta(r, X)$, definito come l'angolo compreso tra r (vettore di n elementi) ed il fattore rappresentato da X_j, che acquista il senso dell'angolo tra il vettore r e lo spazio generato dalle colonne di X_j, ovviamente è di più facile comprensione interpretare l'angolo $\theta(r, X)$ come l'angolo tra r e la sua proiezione r_{X_j} sullo spazio generato da X_j come illustrato in Figura 5.3.

Dunque il $\cos^2\{\theta(r, X)\}$ è proporzionale alla somma dei quadrati delle variazioni in r (è proporzionale alla proiezione r_{X_j}), che è motivato dalla regressione di X_j. Finché X_j è ortonormale si ha che

$$\cos^2(\theta(r, X)) = ||X_j' r||^2 / ||r||^2. \tag{5.17}$$

Iniziando con tutti i coefficienti vettoriali posti a zero, group LARS trova il primo fattore (che chiameremo X_{j1}) che è l'angolo più piccolo con Y e si procede nella direzione della proiezione di Y finché un nuovo fattore (che chiameremo X_{j2}) ha un angolo più piccolo con il residuo, cioè ci si ferma quando

$$||X_{j1}' r||^2 = ||X_{j2}' r||^2 \tag{5.18}$$

dove r è il residuo corrente. A questo punto la proiezione del residuo corrente nello spazio generato dalle colonne di X_{j1} e di X_{j2} ha lo stesso angolo con i due

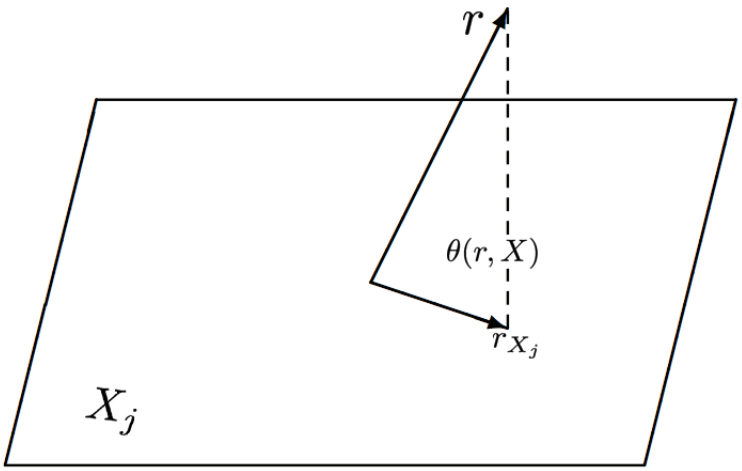


Figura 5.3

fattori ed il group LARS procede in questa direzione. Groups LARS continua in questa direzione finché il residuo con X_{j3} è il più correlato. Group LARS continua come per il LARS ad avanzare finché non trova uno spazio maggiormente correlato e a questo punto aggiorna la sua direzione. Il lettore avrà capito che invece di cercare il vettore maggiormente correlato nel caso dell'algoritmo a gruppi si cerca lo spazio maggiormente correlato, in questa maniera la sparsità non sarà sui singoli coefficienti ma sugli stessi ingressi, i cui coefficienti sono stati raggruppati in un unico spazio.

Per riassumere, la versione implementata a gruppi dell'algoritmo LARS si procede nella seguente maniera:

- Passo 1: si inizia con $\beta^{[0]} = 0$, $k = 1$ e $r^{[0]} = Y$.
- Passo 2: si calcola il vettore maggiormente correlato

$$\mathcal{A}_1 = \arg\max_j ||X_j' r[k-1]||^2/p_j. \tag{5.19}$$

- Passo 3: si calcola la direzione corrente γ come $p = \sum p_j$ vettore di dimensione con $\gamma\mathcal{A}_k^c = 0$

$$\gamma_{\mathcal{A}_k} = (X'_{\mathcal{A}_k} X_{\mathcal{A}_k}) X'_{\mathcal{A}_k} r^{[k-1]} \tag{5.20}$$

  dove $X_{\mathcal{A}_k}$ denota la matrice che comprende le colonne di X corrispondenti a $\mathcal{A}_k$.

- Passo 4:

$$||X_j'(r^{[k-1]} - \alpha_j X\gamma)||^2/p_j = ||X_{j'}'(r^{[k-1]} - \alpha_{j'} X\gamma)||^2/p_{j'} \tag{5.21}$$

  dove j' è arbitrariamente scelto da $\mathcal{A}_k$.

- Passo 5: se $\mathcal{A}_k \neq \{1, \ldots, J\}$, si prende $\alpha = \min_{j \notin \mathcal{A}_k}(\alpha_j) \equiv \alpha_{j^*}$ e si aggiorna $\mathcal{A}_{k+1} = \mathcal{A} \cup \{j^*\}$; altrimenti si imposta $\alpha = 1$.
- Passo 6: si aggiorna $\beta^{[k]} = \beta^{[k-1]} + \alpha\gamma$, $r^{[k]} = Y - X\beta^{[k]}$ e $k = k + 1$. Si torna al passo 3 finché non si ottiene $\alpha = 1$.

L'equazione 5.21 è una equazione quadratica di α_j che può essere risolta facilmente. Nel caso in cui $\alpha_j = 0$, j' è il vettore maggiormente correlato e la parte sinistra dell'equazione è minore rispetto alla parte destra . Comunque in accordo con la definizione di γ , il lato destro dell'equazione 5.21 è zero quando $\alpha_j = 1$. Per cui all'ultima iterazione la soluzione di 5.21 deve essere compresa tra 0 e 1. In altre parole α_j al passo 4 è sempre ben definita. L'algoritmo si ferma solo al raggiungimento di $\alpha_j = 1$, a quel punto il residuo sarà ortogonale con tutte le colonne di X perciò la soluzione dopo l'ultimo passo è la stima ai minimi quadrati, con probabilità 1 che viene raggiunta in J passi.

6 Nuclei per l'identificazione

6.1 Definizione del problema

Sia $\{z_t\}_{t\in\mathbb{Z}}$, $z_t \in \mathbb{R}^m$ un processo stocastico, stazionario che descrive l'evoluzione temporale di qualche variabile di interesse. Con un piccolo abuso di notazione in questo contesto z_t indica sia la variabile aleatoria che la sua realizzazione. Come mostrato in figure 6.1 ogni componente del processo $\{z_t\}_{t\in\mathbb{Z}}$ può essere pensato come un nodo di una grafo. L'obiettivo di questi algoritmi di identificazione è di trovare modelli lineari dinamici che descrivono tutti i componenti di $\{z_t\}$ in funzione degli altri componenti. A questo scopo si definisce $y_t := z_t^{[i]}$ (la componente i-esima di z_t) come l'uscita e tutte le altre componenti $u_t := z_t^{[-i]} \in \mathbb{R}^{m-1}$ come gli ingressi. Ovviamente il procedimento può essere ripetuto per ogni $i = 1, \ldots, m$ ottenendo in questo modo una descrizione completa di tutte le variabili di z_t in funzione delle rimanenti. In questa sezione però per questioni di chiarezza si sceglie un valore specifico per i, per esempio $i = 1$. In questo caso z_t può essere rappresentato come

$$z_t = \begin{bmatrix} y_t \\ u_t \end{bmatrix}$$

Si definiscono gli insiemi delle variabili passate all'istante t

$$Y^t = [y_{t-1}, y_{t-2}, \ldots], \qquad U^t = [u_{t-1}, u_{t-2}, \ldots]$$

Dalla stazionarietà del processo $\{z_t\}_{t\in\mathbb{Z}}$ segue che anche i processi $\{y_t\}_{t\in\mathbb{Z}}$ e $\{u_t\}_{t\in\mathbb{Z}}$ sono processi stocastici, congiuntamente stazionari che possono quindi essere pensati rispettivamente come ingressi e uscita di un sistema dinamico tempo-invariante. Possono essere quindi descritti dalla seguente relazione:

$$y_t = \sum_{k=1}^{\infty} f_k u_{t-k} + \sum_{k=0}^{\infty} g_k e_{t-k} \tag{6.1}$$

dove $f_k \in \mathbb{R}^{1\times m}$ (avendo assunto assenza di retroazione diretta, cioè $f_0 = 0$) e $g_k \in \mathbb{R}$ sono coefficienti incogniti della risposta impulsiva e e_t rappresenta un processo di innovazione. Esso è equivalente all'errore di predizione di un passo

$$e_t := y_t - \hat{y}_{t|t-1} = y_t - E[y_t|Y^t, U^t] \tag{6.2}$$

$$E[y_t|Y^t, U^t] := \sum_{j=1}^{m-1} \left[\sum_{k=1}^{\infty} h_k^j u_{t-k}^{[j]} \right] + \sum_{k=1}^{\infty} h_k^{[m]} y_{t-k}. \tag{6.3}$$

Le sequenze $h_k := [h_k^{[1]}, h_k^{[2]}, \ldots, h_k^{[m]}] \in \mathbb{R}^{1\times m}, k \in \mathbb{Z}^+$ rappresentano i coefficienti della risposta impulsiva del predittore e servono a descrivere sistemi (BIBO) stabili e si ha quindi $h^{[m]} \in l_1(\mathbb{Z}^+)$.

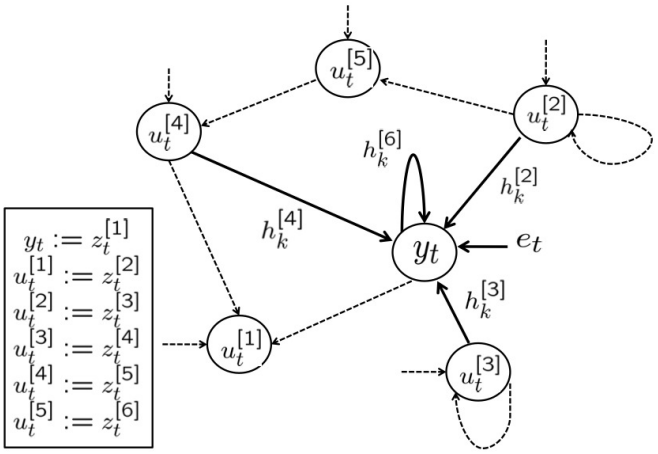


Figura 6.1: Grafo che rappresenta l'interconnessione di $m = 6$ variabili. Gli archi solidi rappresentano le connessioni nel modello dinamico con il nodo $y_t := z_t^{[1]}$ da tutti gli altri nodi. L'assenza degli archi da $u_t^{[i]},\ i = 1, 5$ a y_t indica che $h_k^{[i]} = 0,\ i = 1, 5,\ \forall k \in \mathbb{Z}^+$. Il nodo y_t ha l'arco entrante indicando la dipendenza dall'errore di predizione a un passo e_t. Gli archi punteggiati indicano dipendenze tra gli ingressi u_t.

Nell'ambito della PEM la predizione di sistemi dinamici può essere vista come predizione dei coefficienti della risposta impulsiva h_k del predittore. Si vuole prestare particolare attenzione a situazioni in cui non tutte le variabili sono necessarie per predire y_t, il che in termini matematici equivale ad avere $h_k^{[i]} = 0$ per ogni $k \in \mathbb{Z}^+$. In una rappresentazione grafica c'è una arco diretto $u_k^{[i]}$ al nodo y_k se e solo se esiste $k \in \mathbb{Z}^+$ tale che $h_k^{[i]} \neq 0$ per $i = 1, \ldots, m-1$; inoltre non c'è un capo se e solo se esiste $k \in \mathbb{Z}^+$ tale che $h_k^{[m]} \neq 0$. Per esempio il grafo rappresentato in Figura 6.1 ha $h_k^{[5]} = h_k^{[1]} = 0$, $\forall k \in \mathbb{Z}+$, mentre $h_k^{[2]} = h_k^{[3]} = h_k^{[4]}$ e $h_k^{[6]}$ non sono identicamente nulle. Questo significa che per predire y_t serve il passato di $u^{[2]},\ u^{[3]},\ u^{[4]}$ e di y stesso.

In pratica non si sa a priori quali variabili saranno significative per predire l'uscita y_t. I metodi PEM e N4SID non effettuano alcuna selezione di ingressi e stimano un modello "completo" che tiene conto di tutte le misure. Questo è un motivo per cui questi due algoritmi a volte producono stime scarse.

Qui di seguito si presentano degli algoritmi che favoriscono la sparsità e che sono quindi capaci di individuare una struttura del sistema dinamico simile a quella riportata in Figura 6.1 ed allo stesso momento riescono a stimare le risposte impulsive $h_k^{[i]}$ nulle.

6.2 La regolarizzazione

Un approccio ampiamente utilizzato per ricostruire una funzione da un insieme di misure $\{y_t\}$ consiste nel minimizzare un funzionale di regolarizzazione appartenente ad un RKHS (Reproducing Kernel Hilbert Space) $\mathcal{H}$ associato ad un nucleo K simmetrico e definito positivo.
Date N misure, la regolarizzazione ai minimi quadrati in $\mathcal{H}$ stima la funzione sconosciuta come

$$\hat{h} = arg\min_{h} \sum_{t=1}^{N} (y_t - \Gamma_t[h])^2 + \eta \|h\|_{\mathcal{H}}^2 \tag{6.4}$$

dove $\{\Gamma_t\}$ sono dei funzionali lineari e limitati in $\mathcal{H}$ legati al modello di misura mentre il parametro η è uno scalare positivo che permette di settare il trade-off fra l'errore di predizione e la regolarità della soluzione. La funzione che minimizza (6.4) può essere interpretata da un punto di vista Bayesiano come la stima a minima varianza di h quando h è visto come un processo Gaussiano a media nulla con autocovarianza K e $\{y_t - \Gamma_t[h]\}$ un rumore bianco indipendente da h. Spesso la conoscenza a priori è limitata al fatto che i segnali in gioco siano continui e a energia limitata. In questo caso la h viene modellata con un integrale p-dimensionale di un rumore bianco di intensità unitaria. L'autocorrelazione W_p di h si può così scrivere come:

$$W_p(s,t) = \int_0^1 G_p(s,u) G_p(t,u) du \tag{6.5}$$

$$G_p(r,u) = \frac{(r-u)_+^{p-1}}{(p-1)!}, \quad (u)_+ = \begin{cases} u & \text{if } u \geq 0 \\ 0 & \text{if } u < 0 \end{cases} \tag{6.6}$$

dove G_p è l'autocovarianza associata all'interpretazione Bayesiana di una spline di regolarizzazione di ordine p. Se $p = 2$ si ottiene un nucleo a spline cubica.
Il problema principale nell'utilizzo di un nucleo come (6.6) è che non tiene conto della stabilità della risposta impulsiva. Infatti se l'autocovarianza di h è proporzionale a W_p (come nel caso cubico) la varianza di $h(t)$ è nulla nell'istante iniziale e poi diverge all'infinito. Per questo motivo è stato definito un nucleo apposta per l'identificazione di sistemi lineari in cui h rappresenti la risposta impulsiva di un sistema stabile (con covarianza non nulla nell'instante iniziale che converge esponenzialmente a zero). Questo viene garantito prendendo la covarianza proporzionale alla classe di nuclei

$$K_p(s,t) = W_p(e^{-\beta s}, e^{-\beta t}) \quad s,t \in \mathbb{R}^+ \tag{6.7}$$

dove β è uno scalare positivo che regola il decadimento della varianza; essendo in pratica un parametro sconosciuto sarà trattato come un iperparametro da stimare a partire dai dati. Tenendo presente le formule precedenti si vede che con $p = 2$ l'autocovarianza diventa ciò che in letteratura viene chiamato *Stable Spline kernel*:

$$K_2(t,\tau) = \frac{e^{-\beta(t+\tau)} e^{-\beta \max(t,\tau)}}{2} - \frac{e^{-3\beta \max(t,\tau)}}{6} \tag{6.8}$$

E' stato dimostrato (vedi[??]) che se h è un processo Gaussiano a media nulla con autocovarianza K_2 allora con probabilità uno le realizzazioni di h sono risposte impulsive di sistemi BIBO stabili.
E' chiaro che lavorando in pratica con segnali a tempo discreto si parlerà di versioni campionate h_k di $h(t)$ così che $\{h_k\}_{k\in\mathbb{Z}^+}$ verranno viste come realizzazioni campionate del nucleo $K(i,j)$.

Si considera ora il problema di stimare il parametro $\theta \in \mathbb{R}^m$ dal modello lineare

$$Y = X\theta + W \tag{6.9}$$

dove si ha $Y \in \mathbb{R}^N$ è il vettore dei dati di uscita, $X \in \mathbb{R}^{N\times m}$ è il vettore di regressione e $W \in \mathbb{R}^N$ è un rumore bianco vettoriale a media nulla con $\mathbb{E}[WW^T] = \sigma^2 I$. Ciò che verrà fatto è considerare ogni componente di θ appartenente ad uno spazio infinito dimensionale RKHS con nucleo $K(s,t)$ come in (6.7). In altre parole θ_i viene visto come un processo gaussiano con covarianza $K(s,t)$. La matrice di regressione X sarà un operatore lineare contente la storia passata di u e y.

Il problema di selezionare le variabili più significative può essere affrontato come un problema di ricerca sparsità nella soluzione con alcune importanti differenze. In particolare si tratta ora di un problema a gruppi, poichè per escludere dalla procedura di identificazione delle variabili è necessario che vengano poste a zero delle funzioni di trasferimento. Ogni gruppo di parametri descrive dunque una risposta impulsiva $h^{[i]}$. Per evitare il problema relativo alla selezione del modello e della parametrizzazione non lineare si preferisce non utilizzare algoritmi a gruppi parametrici scegliendo di lavorare in un contesto non parametrico. I parametri vivono dunque in un RKHS che come già detto possono essere trattati mediante probabilità a priori e regolarizzazione.
I due approcci che verranno utilizzati sono:

1. SS-GLAR: Versione a gruppi del problema posto dal Lasso [2] portato in un contesto non-parametrico dove i gruppi $h^{[i]}$ appartengono ad uno spazio infinito dimensionale. Per includere il vincolo del problema è necessario risolvere un problema di regolarizzazione misto $l_1 - l_2$.

2. SSEH: un modello gerarchico dove $h^{[i]}$ è un processo Gaussiano con covarianza $\lambda_i K(s,t)$ e il fattore di scala λ_i ha una distribuzione esponenziale per favorire la sparsità sullo spazio dei fattori di scala.

[2] Nel Lasso la scelta del regressore avviene risolvendo il problema nella forma:

$$\hat{\theta} := arg\min_{\theta} \|Y - X\theta\|_2^2 + \gamma_1\|\theta\|_1 \tag{6.10}$$

con γ_1 parametro di regolarizzazione compreso fra 0 e 1.

A questo punto si rende necessario introdurre della notazione aggiuntiva per poter spiegare gli algoritmi che verranno utilizzati. Si definisce dunque

$$y_t^+ := \begin{bmatrix} y_t \\ \vdots \\ y_{t+N-1} \end{bmatrix} \quad e_t^+ := \begin{bmatrix} e_t \\ \vdots \\ e_{t+N-1} \end{bmatrix} \quad h := \begin{bmatrix} h^{[1]} \\ \vdots \\ h^{[m]} \end{bmatrix}, \tag{6.11}$$

dove $h^{[i]} \in l_1(\mathbb{Z}^+)$, $i = 1, \ldots, m$. Il predittore (6.3) può essere così riscritto [3] come:

$$y_t^+ = \underbrace{[A_{t1} \ \ldots \ A_{tm}]}_{:=A_t} h + e_t^+ \tag{6.12}$$

con $A_{ti} \in \mathbb{R}^{N \times \infty}$, $i = 1, \ldots, m$ definiti da:

$$\begin{array}{rcll} A_{ti}^{[jk]} & := & u_{t-j-k}^{[i]}, & i = 1, \ldots, m-1 \ , \quad j, k \in \mathbb{Z}^+ \\ A_{tm}^{[jk]} & := & y_{t-j-k}, & j \, , k \in \mathbb{Z}^+ \end{array} \tag{6.13}$$

e il problema di identificazione può essere dunque visto come la stima di h in (6.11),(6.12) soggetto al vincolo che $h^{[i]} \in l_1(\mathbb{Z}^+)$, $i = 1, \ldots, m$. Come già detto lo stimatore dovrà automaticamente selezionare tra $\{u^{[1]}, \ldots, u^{[m]}, y\}$ le variabili necessarie alla predizione di y e quali no e questo farà in modo che alcune risposte impulsive $\hat{h}^{[i]}$ siano poste esattamente a zero.

[3] Il prodotto tra matrici semi-infinite va inteso come il limite della sequenza finita, che sotto l'ipotesi $h^{[i]} \in l_1\mathbb{Z}^+$, è ben definito.

7 Predizione in RKHS

Si assume che le risposte impulsive $h^{[i]}$ siano (le versioni campionate) di funzioni in $\mathcal{H}_K$, lo spazio di Hilbert riproduttivo associato al nucleo campionato K. Questo corrisponde al fatto che le risposte impulsive $h^{[i]}$ sono processi Gaussiani con funzione di covarianza K.
Il problema di stimare le risposte impulsive $h^{[i]}$ dai dati misurati può essere formulato come il seguente problema di tipo Tikhonov di regolarizzazione:

$$\{\hat{h}^{[i]}\} = \arg\min_{h^{[i]}\in\mathcal{H}_K} \sum_{t=t_0}^{N}(y_t - \hat{y}_{t|t-1})^2 + \gamma_2^2 \left(\sum_{i=1}^{m} \|h^{[i]}\|^2_{\mathcal{H}_K}\right) \tag{7.1}$$

soggetto a

$$\hat{y}_{t|t-1} = \sum_{i=1}^{m-1}\left[\sum_{k=1}^{\infty} h_k^{[i]} u_{t-k}^{[i]}\right] + \sum_{k=1}^{\infty} h_k^{[m]} y_{t-k}. \tag{7.2}$$

Il parametro γ_2 è il cosiddetto parametro di regolarizzazione che deve regolare il trade-off tra fit e regolarità di $h^{[i]}$.

Per portare questo problema in dimensione finita si modifica il nucleo K in modo che la realizzazione del nucleo modificato soddisfi la risposta impulsiva alla condizione $h_k^{[i]} = 0,\ \forall k > J$. È facile vedere che il nucleo (la covarianza, indicata con K_J) che soddisfa la condizione è $K_J(h,k) = 0,\ \forall(h,k) :\ k > J$ oppure $h > J$.
Raccogliendo i coefficienti della risposta impulsiva in un vettore colonna si ha

$$\underline{h}^{[i]} := [h_1^{[i]}, h_2^{[i]}, \ldots, h_{t_0}^{[i]}]^T,$$

dove $t_0 = J$. Il numero J non ha influenza sul trade-off tra bias e varianza, ma serve solo per ragioni di complessità computazionale. Con un piccolo abuso di notazione si indica con $K_J \in \mathbb{R}^{J\times J}$ la matrice simmetrica, definita positiva formato dai nuclei campionati fino al ritardo J. Sotto queste condizione la norma $\|h^{[i]}\|^2_{\mathcal{H}_{K_J}}$ può essere scritta come

$$\|h^{[i]}\|^2_{\mathcal{H}_{K_J}} = (\underline{h}^{[i]})^T K_J^{-1} \underline{h}^{[i]}.$$

Il predittore a un passo $\hat{y}_{t|t-1}$ diventa

$$\hat{y}_{t|t-1} = \sum_{i=1}^{m-1}\left[\sum_{k=1}^{J} h_k^{[i]} u_{t-k}^{[i]}\right] + \sum_{k=1}^{J} h_k^{[m]} y_{t-k}. \tag{7.3}$$

Risulta che sotto la restrizione $h^{[i]} \in \mathcal{H}_{K_J}$ la soluzione del problema 7.1 è

$$\arg\min_{\underline{h}^{[i]}\in\mathbb{R}^{t_0}} \sum_{t=t_0}^{N}(y_t - \hat{y}_{t|t-1})^2 + \gamma_2^2 \sum_{i=1}^{m}\left[(\underline{h}^{[i]})^T K_J^{-1} \underline{h}^{[i]}\right] \tag{7.4}$$

soggetto a 7.3. Si vede quindi che il problema può essere riformulato come quello di stimare un modello ARX con un peso di regolarizzazione incluso nel problema della stima.

8 L'algoritmo SSGLAR

Viene ora discusso come si può risolvere il problema 7.4 invocando sparsità nei gruppi $h_{t_0}^{[i]}$ usando l'algoritmo GLAR. Per fare questo si assume che i parametri θ siano stati fissati; senza alcuna informazione a priori questo parametro viene posto uguale a zero.
Il problema 7.4 è un problema di regressione lineare con una funzione peso in l_2 che può essere riscritto nella forma

$$\overline{y}_{t_0}^{+} = \sum_{i=1}^{m} \overline{A}_{Ji} h_{t_0}^{[i]} + W \tag{8.1}$$

avendo definito

$$\begin{aligned} \overline{y}_{t_0}^{+} &:= \begin{bmatrix} y_{t_0}^{+} \\ 0_{1\times(t_0\cdot m)} \end{bmatrix} \\ \overline{A}_{Ji} &:= [A_{Ji} \quad v_i \otimes \Lambda] \quad \Lambda := \gamma_2 K_J^{-1/2} \\ v_i &:= [\underbrace{0\ldots0}_{i-1} 1 \underbrace{0\ldots0}_{m-i}]^T \end{aligned}$$

La selezione degli ingressi come descritto prima può essere eseguito dall'algoritmo GLAR applicato al problema di regressione. Nella prossima sezione si definisce nel dettaglio questo algoritmo che viene chiamato SS-GLAR.

I passi: I passi che compie l'algoritmo SS-GLAR sono:

1. stabilire il parametro β del nucleo;
2. stabilire il parametro γ_2 di 7.4 e formare $\overline{A}_{Ji}$ in 8.1;
3. stima di $h^{[i]}$ applicando l'algoritmo GLAR al problema 8.1;

8.1 Stima degli iperparametri

Prima di poter eseguire l'algoritmo appena visto bisogna fissare i seguenti parametri:

1. la funzione peso γ_2 in l_2 nell'equazione 8.1 (regolarità di $[h_1^{[i]}, h_2^{[i]}, \ldots, h_{t_0}^{[i]}]^T$ nello spazio $\mathcal{H}_{K_{t_0}}$);
2. il parametro β del nucleo (velocità di decrescita del nucleo);
3. Il numero di blocchi diversi da zero stimati dall'algoritmo GLAR;

Questi parametri possono essere stimati usando il seguente approccio di validazione: disponendo delle misure $\{y_t, u_t\}_{t=1,\dots,N}$ si suddivide l'insieme in due parti. Il primo $\{y_t, u_t\}_{t=1,\dots,2N/3}$ viene chiamato l'insieme di identificazione, il secondo con i dati rimanenti viene chiamato l'insieme di validazione. Si esegue l'algoritmo per iperparametri fissati usando l'insieme di identificazione e successivamente si valuta il modello ottenuto sull'insieme di validazione. Dall'insieme degli iperparametri ($\beta \in \mathbb{R}^+, \gamma \in \mathbb{R}^+$) si sceglie un insieme con un numero finito (possibilmente piccolo) di alternative, sul quale poi si deve testare l'algoritmo. Vengono inoltre testati diversi livelli di sparsità. Alla fine viene scelto il modello identificato che ha dato le miglior prestazioni sul set di validazione in termini di RMS_1 (Root Mean Square) dell'errore di predizione a un passo, dove il generico RMS_k (k numero di passi) è definito da:

$$RMS_k := \sqrt{\frac{1}{N}\sum_{t=1}^{N}(y_t - \hat{y}_{t|t-k})^2}.$$

Vengono quindi fissati gli iperparametri e il livello di sparsità con i quali viene rifatta la stima del modello usando l'intero insieme di dati $\{y_t, u_t\}_{t=1,\dots,N}$.

9 L'algoritmo SSEH

In questa sezione il predittore di equazione 7.2 viene formulato in un contesto Bayesiano assumendo che le risposte impulsive $h^{[i]}$ siano processi Gaussiani con covarianza $\lambda_i K(t,s)$. Serve specificare gli iperparametri β, θ e λ_i e la varianza di rumore σ^2 che descrivono interamente le densità di probabilità delle misure di y e le risposte impulsive dei predittori $h^{[i]}$.

9.1 Iperprobabilità a priori per gli iperparametri

La varianza di rumore σ^2 viene stimata in un passo preliminare usando un modello ARX per poi supporla nota nella stima Bayesiana. Gli iperparametri β, θ e λ_i vengono invece modellati come variabili aleatorie indipendenti. β ha una densità di probabilità non informativa (cioè inizialmente non si hanno informazioni a priori sul parametro) su $\mathbb{R}^+$, mentre θ è distribuita uniformemente. Ogni λ_i è una variabile aleatoria esponenziale con media $1/\xi \in \mathbb{R}^+$, quindi vale

$$p(\lambda_i) = \xi exp(-\xi\lambda_i)\chi(\lambda_i \geq 0), \qquad i = 1, \ldots, m$$

con χ la funzione indicatrice. ξ può essere interpretata come una variabile aleatoria non informativa su $\mathbb{R}^+$. Tutti gli iperparametri possono essere raccolti in un vettore aleatorio indicato con $\zeta := [\lambda_1, \ldots, \lambda_m, \theta_1, \theta_2, \beta, \xi]$.
Si noti che i parametri λ_i, che hanno lo stesso ruolo di σ^2/γ^2 con γ^2 definito in 7.1, possono ora essere tutti differenti aumentando così la flessibilità del modello.

9.2 Il modello Bayesiano intero

Sia $A_{ti} \in \mathbb{R}^{N\times\infty}$ come definito in 6.13 in modo che per le uscite future $y^+ := y_1^+$ risulta

$$y^+ = \left[\sum_{i=1}^{m} A_{1i} h^{[i]}\right] + e^+,$$

dove $e^+ := e_1^+$.
In pratica $y^- := [y_0, y_{-1}, y_{-2}, \ldots]$ non è mai interamente noto e una soluzione consiste nel porre a zero le componenti non note. Viene quindi imposta la seguente approssimazione:

$$\begin{aligned} p(y^+, \{h^{[i]}\}, y^-, u|\zeta) &\propto \left[\prod_{t=1}^{N} p(y_t|\{h^{[i]}\}, y_t^-, \zeta, u_t^-)\right] p(y^-, \{h^{[i]}\}, u^-|\zeta) \\ &\approx \left[\prod_{t=1}^{N} p(y_t|\{h^{[i]}\}, y_t^-, \zeta, u_t^-)\right] p(\{h^{[i]}\}|\zeta) p(y^-, u^-) \end{aligned}$$

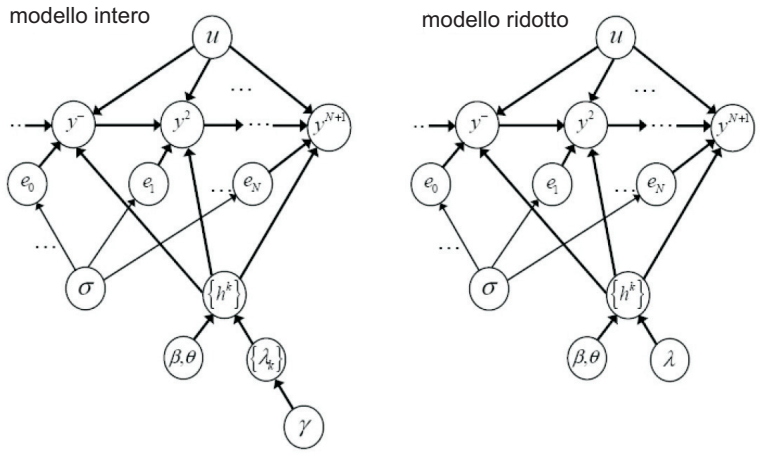


Figura 9.1: Rete Bayesiana che descrive il nuovo modello non parametrico per l'identificazione di sistemi dinamici sparsi, dove $y^- := [y_{l-1}, y_{l-2}, \ldots]$ e nel modello ridotto $\lambda := \lambda_1, \ldots, \lambda_m$.

La proporzionalità deriva dall'assunzione che il predittore di u_t dato il passato u_t^- e y_{t+1}^- non dipenda da ζ mentre l'approssimazione deriva dall'assunzione che il passato u_t^- e y_t^- porti informazione sulla risposta impulsiva del predittore e sugli iperparametri. Questo modello statistico è descritto dalla rete Bayesiana riportata in Figura 9.1 (sinistra).
Come visto sopra, per questioni di complessità computazionale bisogna troncare la risposta impulsiva $h^{[i]}$ a una certa lunghezza t_0.

9.3 Stima degli iperparametri

Il vettore degli iperparametri ζ viene stimato minimizzando la probabilità marginale, cioè la densità di y^+, ζ e $\{h^{[i]}\}$. Può essere dimostrato (si guardi [2]) che la stima si ottiene da

$$\hat{\zeta} = arg \min_{\zeta} J(y^+; \zeta) \tag{9.1}$$

con $\xi, \beta > 0$, $\lambda_i > 0$, per $i = 1, \ldots, m$ e J dato da

$$J(y^+; \zeta) = \frac{1}{2} log(det[2\pi V[y^+]]) + \frac{1}{2}(y^+)^T (V[y^+])^{-1} y^+ + \xi \sum_{i=1}^{m} \lambda_i - log(\xi) \tag{9.2}$$

$$V[y^+] = \sigma^2 I_N + \sum_{k=1}^{m} \lambda_i A_{1i} K A_{1i}^T . \tag{9.3}$$

Una questione molto importante per l'utilizzo pratico di questo procedimento matematico è l'utilizzo di un buon valore iniziale. Nel seguito viene descritto un

schema con il quale è possibile ottenere una soluzione sub-ottimale risolvendo il problema di ottimizzazione in $\mathbb{R}^4$ in relazione al modello Bayesiano di Figura 9.1 (destra).

(i) Si ottengono $\{\hat{\lambda}_i\}$, $\hat{\theta}$ e $\hat{\beta}$ risolvendo il problema 9.1 modificato nella seguente versione:

$$\arg\min_{\zeta}[J(y^+;\zeta) - \xi\sum_{i=1}^{m}\lambda_i + log(\xi)]$$

con $\beta > 0,\ \lambda_1 = \ldots = \lambda_m \geq 0$

(ii) Si fissano $\hat{\xi} = 1/\hat{\lambda}_1$ e $\hat{\zeta} := [\hat{\lambda}_1, \ldots, \hat{\lambda}_m, \hat{\theta}_1, \hat{\theta}_2, \hat{\beta}, \hat{\xi}]$. Poi, per $i = 1, \ldots, m$ si fissa $\bar{\zeta} = \hat{\zeta}$ tranne la i-esima componente di $\bar{\zeta}$ che viene messa a zero. Se invece $J(y^+;\bar{\zeta}) \leq J(y^+;\hat{\zeta})$ si fissa $\hat{\zeta} = \bar{\zeta}$.

La procedura descritta al passo (ii) potrebbe non essere adatta nel caso in cui gli ingressi sono altamente correlati, perché potrebbe risultare sensibile all'ordine con cui le componenti di $\hat{\zeta}$ convergono a zero. Per aggirare questo problema si propone di usare l'algoritmo SS-GLAR per effettuare questa selezione. In particolare si può sostituire il passo (ii) con il passo seguente:

(ii') Si fissa $\gamma_2^2 = \sigma^2/\hat{\lambda}_1$ e si aggiusta il nucleo visto in sezione 8 con $K(:,:,\hat{\theta},\hat{\beta})$. Si esegue poi l'algoritmo SS-GLAR selezionando il numero di componenti non nulli usando l'approccio di validazione. A questo punto i componenti $\{j_1, \ldots, j_k\} \subseteq \{1, \ldots, m\}$ di $\hat{\zeta}$ vengono annullati, il che equivale a annullare $\hat{h}^{[j]} = 0,\ j \in \{j_1, \ldots, j_k\}$.

Il valore $\hat{\zeta}$ finale può essere usato come punto iniziale in problema 9.3.

9.4 Stima delle risposte impulsive con ζ noto

Sotto certi condizioni (vedi [2]) si può dimostrare:

$$\{\hat{h^{[i]}}\}_{i=1}^{m} = arg\min_{\{f^i \in \mathcal{H}\}_{i=1}^{m}} \|y^+ - \sum_{i=1}^{m} A_{1i} f^i\|^2 + \sigma^2 \sum_{i=1}^{m} \frac{\|f^i\|_{\mathcal{H}_K}^2}{\lambda_i^2}$$

Inoltre per $k = 1, \ldots, m+1$ si ha

$$\hat{h^{[i]}} = \lambda_i^2 K A_{1i}^T c, \quad c = \left(\sigma^2 I_n + \sum_{i=1}^{m+1} \lambda_i A_{1i} K A_{1i}^T\right)^{-1} y^+$$

Dopo aver ottenute le stime di $\{h^{[i]}\}$ semplice formule possono essere usate per ottenere le risposte impulsive f e g in 6.1 e quindi anche il predittore a k passi (per dettagli vedere [3]).

10 Indici di validazione

Una parte molto importante dell'identificazione è la fase di validazione del modello ottenuto in cui si cerca di capire la bontà del modello identificato nel descrivere il processo. Per fare questa analisi delle prestazioni è necessario un insieme di dati diversi da quelli usati per l'identificazione. I dati disponibili vengono quindi suddivisi in due insiemi:

- dati per l'identificazione,
- dati per la validazione.

La validazione consiste nel confrontare l'uscita reale con l'uscita del modello identificato che prende in ingresso i dati reali (il set per la validazione) e di calcolare successivamente diversi indici che danno informazione sulla bontà del modello.
Di seguito vengono spiegati i diversi indici significativi alla fine di fare validazione.

10.1 Covarianza e correlazione

La covarianza (campionaria) e la correlazione (campionaria) si possono facilmente calcolare in Matlab `cov`(x_1, x_2) e `corr`(x_1, x_2) rispettivamente.
La covarianza indica se esiste una dipendenza tra le due variabili. Valori elevati indicano una forte legame tra le due variabili, mentre per un legame debole la covarianza tende a zero. La covarianza dipende però della base scelta per rappresentare il modello (per esempio dipende dalla unità di misura utilizzata) e non è dunque un indice oggettivo per confrontare la bontà di diversi modelli Per migliori chiarimenti vedere [1]. La correlazione è invece un parametro adimensionale compreso tra -1 e 1 che indica la relazione lineare tra due variabili. Valori negativi della correlazione indicano che le due variabili sono inversamente correlate, cioè se una variabile cresce l'altra tende a decrescere. Due variabili indipendenti hanno correlazione nulla mentre non è vero il viceversa perché la correlazione può essere nulla anche se esiste una qualsiasi relazioni non lineari tra le due variabili.

10.2 Test di bianchezza

I due indici appena visti spiegano l'importanza del test di bianchezza. Serve per verificare se l'errore di predizione (residuo) tra i dati reali e l'uscita dal modello identificato è effettivamente un processo bianco. In base alla bianchezza di una realizzazione del processo vero si decide se il modello identificato è una buona descrizione del sistema dinamico reale.
Per effettuare il test si bianchezza è utile il commando `resid` di Matlab che restituisce un grafico con la auto-correlazione dell'errore di predizione e uno con la

cross-correlazione tra l'errore di predizione e ogni ingresso reale che viene poi utilizzato per effettuare il test di indipendenza. Inoltre nei grafici viene visualizzato l'intervallo di confidenza al 99%, che è un intervallo di valori plausibili ricavato dalla specifica distribuzione di probabilitá dei residui. Per avere bianchezza tutte le correlazioni (tranne la prima che in realtá è la varianza del residuo) dovrebbero trovarsi all'interno dell'intervallo di confidenza, o almeno non eccessivamente fuori. Nel caso contrario si ha che i campioni dell'errore di predizione sono correlati, il che si può avere per esempio se l'ordine del modello usato per l'identificazione non è adeguato e quindi il modello non è adeguato per descrivere il sistema dinamico vero.
Assumendo che l'errore di predizione $\varepsilon(t)$ sia bianco si ha che la covarianza $r_\varepsilon(\tau) = 0$ per ogni $\tau \neq 0$ e quindi per la covarianza campionaria

$$\hat{r}_\varepsilon(\tau) = \frac{1}{N} \sum_{t=1}^{N-\tau} \varepsilon(t+\tau)\varepsilon(t)$$

si ha per $N \to \infty$

$$\begin{aligned} &\hat{r}_\varepsilon(\tau) \to 0 \qquad \forall\ \tau \neq 0, \\ &\hat{r}_\varepsilon(0) \to E[\varepsilon^2(t)]. \end{aligned}$$

Per il test di bianchezza infine viene considerato la quantità normalizzata

$$x_\tau = \frac{\hat{r}_\varepsilon(\tau)}{\hat{r}_\varepsilon(0)}.$$

10.3 Test di indipendenza

Il test di indipendenza è relativo alla cross-correlazione tra l'errore di predizione e gli ingressi reali. Si verifica se i residui sono effettivamente indipendenti dalle osservazioni passate e quindi se è stata estratta tutta l'informazione possibile dai dati. Come accennato prima per concludere che il modello identificato è adeguato si controlla nei grafici della cross-correlazione tra il residuo e gli ingressi reali ottenuto mediante il commando `resid` che non ci siano picchi significativamente fuori dalla regione di confidenza. Se per esempio si ha un picco al ritardo k-esimo che sta fuori della regione di confidenza questo significa che l'uscita $y(t)$ non ha ancora estratto tutta l'informazione possibile dall'ingresso $u(t-k)$ e un altro modello potrebbe spiegare meglio l'uscita all'istante t.
Il commando `resid` assume che il residuo $\varepsilon(t)$ sia indipendente dagli ingressi passati e sia a media nulla e che si ha quindi $r_{\varepsilon u}(\tau) = E\varepsilon(t+\tau)u(t) = 0$.
Per τ si considerano sia valori positivi che negativi. Mentre se il modello non è adeguato per descrivere il sistema per valori di τ positivi si può aspettare che ci siano picchi di cross-correlazione che sono fuori dell'intervallo di confidenza questo non è detto per valori di τ negativi. Per esempio nel caso in cui $u(t)$ sia un processo

bianco questi picchi sono comunque nulli.
La correlazione campionaria si calcola come

$$\hat{r}_{\varepsilon u}(\tau) = \frac{1}{N}\sum_{t=m}^{M}\varepsilon(t+\tau)u(t)$$

con $m = 1-\min(0,\tau)$ e $M = N-\max(0,\tau)$. Per il test di indipendenza si considera la quantità normalizzata

$$x_\tau = \frac{\hat{r}_{\varepsilon u}(\tau)}{[\hat{r}_\varepsilon(0)\hat{r}_\varepsilon(0)]^{1/2}}.$$

10.4 Mappa zeri-poli

Tracciando la mappa zeri-poli di ogni funzione di trasferimento del modello identificato si può dare una prima indicazione sull'esattezza dell'ordine del modello. Se ci sono delle quasi-cancellazioni si può aspettare che è possibile ridurre l'ordine del modello identificato.

10.5 Fit

Il fit da la percentuale della varianza spiegata ed è calcolato mediante la seguente formula:

$$fit := 100 * \left[1 - \frac{||y_h - y||}{||y - E[y]||}\right]$$

dove y_h sono le uscite dal modello identificato, mentre y è l'uscita reale. Dalla formula si vede che più l'uscita del modello è vicina a quella reale, più vicino è il fit al 100% e piú sarrà quindi adeguato il modello identificato per spiegare il sistema reale.

10.6 Errore quadratico medio

Un'indice che valuta la prestazione del modello identificato che viene spesso usato è l'MSE (mean square error). Esso è definito da

$$\begin{aligned} MSE &:= \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \\ &= Var(\hat{y}) + (E[\hat{y} - y])^2 \end{aligned}$$

Questo indice fornisce quindi informazioni sulla bontà dello stimatore in termini di varianza e dispersione della stima. Spesso si calcola anche la radice di questo indice, che viene chiamata con il nome di radice dell'errore di predizione (in inglese root mean square error, RMS) che è quindi proporzionale alla deviazione standard della

stima.
Il valore RMS per l'errore di predizione a k passi viene così calcolato:

$$RMS_k := \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_{i|i-k})^2}$$

10.7 Coefficiente di determinazione, COD

In statistica il coefficiente di determinazione è usato principalmente quando l'obbiettivo dell'identificazione è la predizione delle uscite future. Il COD fornisce una misura sulla bontà della predizione dei dati futuri.

$$COD_k := 1 - \frac{RMS_k^2}{\frac{1}{N}\sum_{i=1}^{N}(y_i - \bar{y})^2} \tag{10.1}$$

dove $\bar{y}$ è la media campionaria di y_i. In generale dunque può essere visto in relazione alla varianza non spiegata, infatti il secondo termine compara la varianza dell'errore del modello con la varianza totale dei dati. Il COD è una statistica minore o uguale ad 1 che assume il valore massimo solo quando i dati predetti coincidono con quelli reali.

10.8 Sparsità

La sparsità dà la percentuale di funzioni di trasferimento nulle nel modello identificato. Avere una funzione di trasferimento nulla significa che la rispettiva misura (ingresso) non è stata utilizzata alla fine di calcolare l'uscita del modello identificato e quindi che il rispettivo sensore può essere eliminato nel calcolo della rispettiva uscita.

11 Dispositivi utilizzati per la raccolta dati

Per eseguire le misurazioni delle grandezze di interesse sono stati utilizzati i dispositivi *MoteIV Tmote Sky* equipaggiati con sensori per il rilevamento di temperatura, umidità e luminosità. Sebbene questi sensori siano predisposti per una comunicazione dei dati mediante rete wireless si è scelto di effettuare l'acquisizione off-line sfruttando la memoria flash per immagazzinare i dati raccolti più delle informazioni aggiuntive che sono stati letti alla fine mediante usb. La scelta di utilizzare questa modalità di lettura è dipesa principalmente dall'esigenza di un attento utilizzo del consumo energetico, infatti sebbene questi dispositivi siano progettati per garantire un basso consumo ed abbiano un processore in grado di funzionare con tensioni che arrivano fino a 1.9 V i componenti necessari all'esperimento (i sensori e la memoria flash) necessitano di una tensione costante di circa 3 V fornita da due batterie di tipo AA alcaline. Se la tensione in ingresso diminuisse eccessivamente potrebbero accadere dei malfunzionamenti al convertitore ADC con una perdita di bit significativi (da 14 bit a 12 bit upper) e alla memoria flash che per una scrittura sicura richiede una almeno 2.6 V. Si fa notare che la scelta di una alimentazione mediante porta usb per questa esperienza non è possibile poichè in questo caso l'alimentazione a 5 V provoca un surriscaldamento del processore posto in prossimità del sensore di temperatura, sfalsando così la misura di temperatura. In questo modo si è riusciti ad avere una alimentazione corretta durante tutta la raccolta dei dati, in Figura 14.2 è riportato l'andamento della tensione di ogni dispositivo raccolta proprio per monitorare il buon funzionamento di ogni dispositivo.
Viene riportata qui di seguito una breve descrizione dei sensori utilizzati dai *Tmote* e successivamente del software necessario:

1. *Umidità e temperatura*: Il sensore di temperatura e umidità utilizzato sul mote è il sensore Sensiron SHT11. Esso consta di un sensore di temperatura il cui principio di funzionamento si basa sulla dipendenza dalla temperatura della banda di non conduzione di un semiconduttore ed ha range di funzionamento: -40/+123.8 C. La sua risoluzione è a 14 bit (può essere ridotta a 12 bit per applicazioni che richiedono velocità elevate o un utilizzo con potenza in ingresso molto bassa). La calibrazione è gestita internamente in base a delle tabelle, la formula di conversione dalla parola a 14 bit T_{raw} presente nella memoria flash a gradi centigradi è

$$T = b_1 + b_2 T_{raw} - b_3 (T_{raw} - b_4)^2 \quad [^\circ C] \tag{11.1}$$

   con costanti $b_1 = -39.6$, $b_2 = 0.01$, $b_3 = -2 \cdot 10^{-8}$ e $b_4 = 7000$.
   Il sensore per rilevare l'umidità è basato sulla variazione di capacità di un polimero in funzione dell'umidità presente nell'ambiente circostante e ha range di funzionamento 0/100 % RH. La misura viene registrata in una parola a 12 bit H_{raw} che può scendere ad 8 bit se necessario. La formula per la conversione

è:

$$RH_{true} = (T - 25)(c_1 + c_2 H_{raw}) + c_3 + c_4 H_{raw} + c_5 H_{raw}^2 [\%] \quad (11.2)$$

con constanti $c_1 = 0.01$, $c_2 = 8 \cdot 10^{-6}$, $c_3 = -4$,$c_4 = 0.0405$ e $c_5 = -2.8 \cdot 10^{-6}$. Per maggiori informazioni si rimanda al datasheet del componente scaricabile facilmente dal sito inernet della Sensiron.

2. *Luminosità* I sensori presenti nel mote per misurare la quantità di luce sono due fotodiodi prodotti dalla Hamamatsu. In particolare è presente il modello S1087 sensibile alla sola luce visibile (320 - 730nm) e il modello S1087-01 che rivela anche lo spettro infrarosso (320 - 1100nm). Le misure sono memorizzate in parole da 16 bit unsigned e la conversione in lux é data da:

$$AR = 625 AR_{raw} \quad [lux] \quad (11.3)$$

con AR_{raw} il valore in bit fornito dal sensore S1087. Analogamente a partire dal dato in bit TR_{raw} del sensore S1087-01 si implementa

$$TR = 76.9 TR_{raw} \quad [lux]\,. \quad (11.4)$$

Per la fase di identificazione la misura che viene utilizzata è quella fornita dal secondo sensore, ovvero quella comprensiva della radiazione visibile e dell'infrarosso poichè entrambe influiscono sul comportamento dinamico dell'edificio.

3. *Memoria flash* Il mote è equipaggiato con una unità di memoria flash esterna di tipo seriale di capienza 1024kb. Essendo tuttavia configurata in lettura e scrittura solo nei primi quindici settori lo spazio effettivamente disponibile è di 983040 byte. La struttura dati presente consta di 6 campi da 16 bit ciascuno, il che significa una capacità totale di 81920 campioni. Questo valore è in realtà ridotto a 65535 campioni poichè parte dello spazio è utilizzato per controllare se dei dati non vengono scritti in memoria (di fatto l'ultimo dei sei campi è occupato da un numero che aumenta di un passo ad ogni istante di campionamento). Contando che il tempo di campionamento adatto per le dinamiche termiche di un edificio è dell'ordine dei 5-10 minuti si capisce che la memoria disponibile nel mote è più che sufficiente per lo svolgimento dell'esperienza.

Si fornisce qui di seguito una breve descrizione del codice che viene utilizzato per la gestione dei mote. La sessione di acquisizione viene suddivisa in tre fasi dal firmaware caricato su ogni mote. La prima parte è dedicata al setup che viene effettuato comunicando con il mote tramite usb e pc. In questa fase vengono gestite le operazioni principali di lettura della memoria flash e di cancellazione della stessa. La seconda fase è dedicata alla sincronizzazione e il passaggio avviene allo scadere di un timer opportunamente impostato. In questa fase le comunicazioni usb sono interrotte e gli unici eventi che il mote è può rilevare sono ricezione del pacchetto radio

di sincronizzazione o la pressione del tasto user (che causa la partenza di un nuovo timer e l'invio di un nuovo pacchetto di sincronizzazione). Con la terza fase si inizia l'acquisizione vera e propria dei dati. Il modulo radio viene disattivato ed un timer posto uguale al tempo di campionamento comanda la lettura dei dati rilevati dai sensori e la scrittura su flash. Il principio di funzionamento è riportato in Figura 11.1.
L'interfaccia con il PC è sviluppata in Java su piattaforma Linux ed utilizza le librerie di comunicazione seriale e le classi per la gestione dei pacchetti fornite con TinyOS 2.x. La lettura del mote fornisce in outuput un file di testo contenente i valori registati dai sensori che vengono convertiti in unità ingegneristiche da uno script in Matlab secondo (11.2)-(11.4). La programmazione dei mote è avvenuta eseguendo lo script $EggMote.h$ in cui si precedentemente impostato il tempo di campionamento.

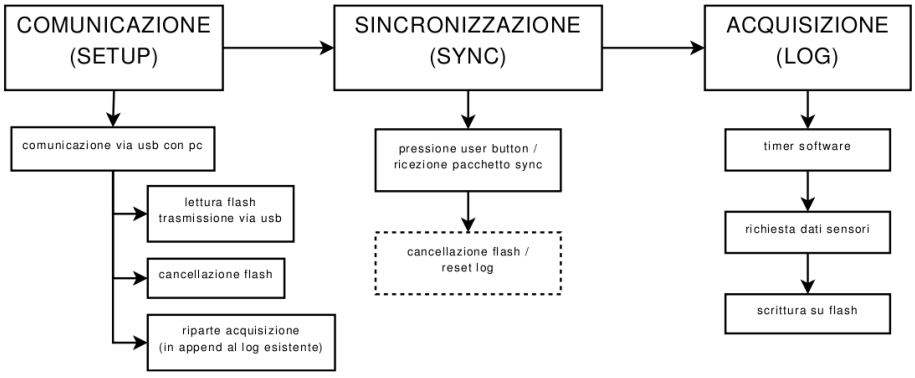
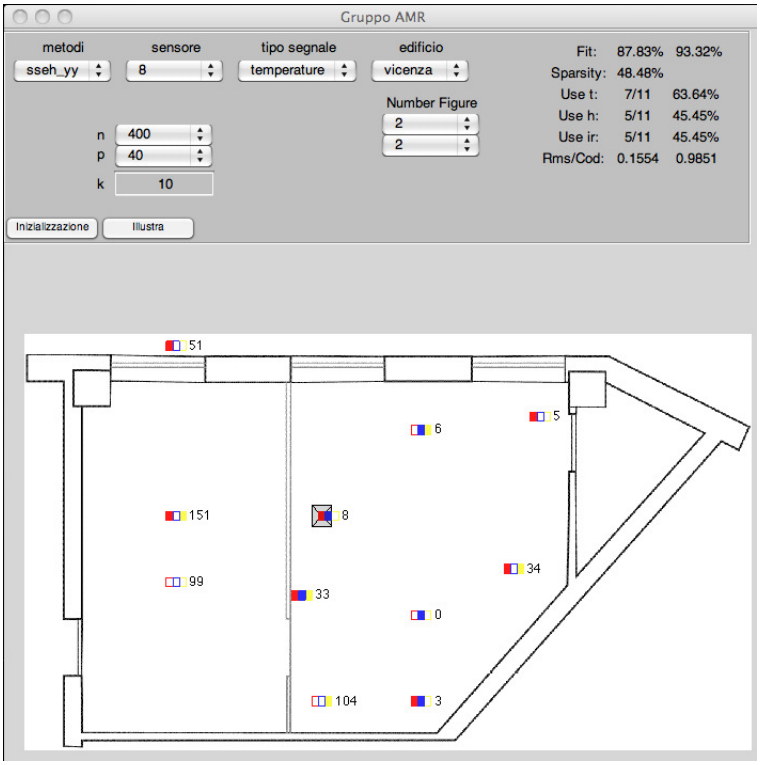


Figura 11.1: Struttura degli stati e delle funzionalità del mote

12 L'interfaccia grafica

Come visto nei paragrafi precedenti per identificare l'andamento termico di un edificio si è dovuto far fronte ad un gran numero di identificazioni da effettuare per tracciare e predire l'andamento delle grandezze termiche in più punti dell'edificio. Per questo motivo si è reso necessario lo sviluppo di un interfaccia grafica per il veloce ispezionamento dello status di ogni sensore per valutarne il suo utilizzo.

Per gli algoritmi che davano una forte sparsità è interessante una comprensione spaziale di quali ingressi vengono usati e di quali invece vengono scartati, cosa che, per sistemi di queste dimensioni, è difficoltosa da effettuare guardando le funzioni di trasferimento e risulta agevolata dalla visualizzazione grafica.



L'interfaccia grafica realizzata non è un simulatore, cioè non richiama direttamente le funzioni di identificazione poichè la quantità di tempo richiesta da certi algoritmi avrebbe fatto diventare la GUI non funzionale. La scelta verso cui si è optato è stata quella della progettazione di una struttura dati in cui sono stati inseriti tutti i risultati delle simulazioni in modo tale da effettuare efficienti operazioni di calcolo per l'analisi dei risultati. Questo è stato un passaggio fondamentale per la gestione di grando mole di dati.

La GUI è composta da due zone principali: la prima la zona di selezione dei parametri , mentre la seconda è una zona grafica dove vengono visualizzati i sensori per illustrare l'utilizzo in base ai vari algoritmi. Come verrà spiegato in seguito si

è scelto di eseguire invece gli altri grafici fuori dell'interfaccia per lasciare all'utente la possibilità di generarli e manipolarli con tutte le potenzialità della libreria plot di Matlab.

Come si vede i campi di scelta sono: "Metodi", "Sensore", "Tipo Segnale" ed "Edificio". Con il primo si seleziona il tipo di algoritmo con cui si è eseguita l'identificazione, i cui metodi supportati sono i seguenti:

- sseh_yy
- sseh_ny
- n4sid
- ssglar
- glar
- pem

Per una descrizione leggere i paragrafo competente in cui vengono spiegati.

Con il secondo menù a tendina si seleziona quale sensore va considerato come uscita , l'elenco conterrà gli id dei vari sensori per una veloce corrispondenza tra la pianta e la selezione. Il sensore scelto verrà evidenziato da un quadrato di sfondo grigio con una croce di S. Andrea che ne indicherà la selezione. Con il terzo menù si sceglie se si vuole identificare la temperatura o l'umidità , mentre con l'ultimo si scegli su quale edificio si vuole attuare questa scelta (questa possibilità è stata prevista in visione di possibili svolgimenti futuri su più edifici presupponendo che la struttura dati venga sempre costruita con i dati ricavati dai motes con il codice matlab sviluppati).

A destra dei quattro menù di selezione vi è una zona in cui vengono illustrati ogni volta i risultati dei fit dell'identificazione e la sparsità dell'algoritmo per una veloce valutazione.

Invece sotto i quattro menù a tendina si trovano due colonne, la prima riguardante i parametri dell'identificazione, la seconda relativa alla creazione delle figure.

Sempre nella parte superiore dell'interfaccia in basso a sinistra si trovano due pulsanti: "inizializza" ed "illustra". Il primo serve a selezionare in un sol colpo tutti menù per essere pronti a visualizzare il risultato di una identificazione, infatti il pulsante "illustra" non viene attivato finché tutti e quattro i menù non siano stati selezionati.

La parte in cui viene visualizzata la mappa dell'edificio ogni sensore è raffigurata da un quadrato di colore rosso per la temperatura, di colore blu per l'umidità e di colore giallo per l'irradiazione. I quadrati possono essere pieni o vuoti, con il significato che il sensore relativo è stato utilizzato o meno dal processo di identificazione. Essendo ogni tmote dotato di tutti e tre i sensori sulla mappa viene illustrato un rettangolo formato dai tre quadrati e dall'id del sensore.

13 La struttura Dati

Tutti i dati e tutte le simulazioni sono state salvate in un unica struttura dati Matlab in modo tale da rendere facilmente reperibile ogni informazione. Come spiegato nelle sezioni precedenti, se è stato utilizzato il codice per i tmote e il successivo codice matlab per la trasformazione dei file .dat la struttura Matlab che viene fornita dal file test_ciclo_MAIN_FIN.m avrà sempre la stessa forma. La struttura Dati è una strutturata è stata suddivisa in un numero di elementi pari ai motes utilizzati nella raccolta dati. Ogni sensore conincide con un'uscita da identificare. All'interno della cella che rappresenta il sensore si trova il campo relativo al nome del mote, un altro con il suo id identificativo nella rete ed un campo per i dati relativi alla temperatura e quelli per l'umidità.
Selezionando la grandezza fisica che si è interessati (nel campo $.t$ e $.h$) a stimare si trovano tutti i dati con i risultati delle identificazioni raggruppati a seconda dell'algoritmo utilizzato suddivisi all'interno di ognuno in base ai parametri ai parametri di identificazione.

14 Risultati sperimentali

In questa sezione vengono analizzati e confrontati i modelli ricavati dalle identificazioni effettuate con i metodi presentati precedentemente. Si darà una descrizione dei risultati ottenuti con i singoli metodi al variare del numero di dati in ingresso (si cerca di valutare la robustezza del metodo in condizioni critiche, ovvero quando le variabili da stimare sono molte rispetto ai dati disponibili) e di alcuni parametri specifici all'algoritmo in questione. Le metriche che vengono utilizzate per valutare la bontà dei risultati ed effettuare i confronti sono quelle della Sezione 10.

I dati sono stati raccolti per due settimane in un edificio adibito ad uso commerciale a Vicenza tra la fine di agosto e l'inizio di settembre. Durante il periodo di acquisizione dei dati l'edificio è rimasto chiuso al pubblico, non si sono così registrate perturbazioni dovute ad agenti esterni (come l'improvvisa apertura di finestre o il semplice riscaldamento termico dovuto alla presenza di un gran numero di persone). Esso è composto da due uffici separati da un muro di cartongesso come illustrato dalla mappa in Figura 14.1. Per la locazione dei sensori si rimanda alle illustrazioni relative alla GUI creata (Sezione 12). Il tempo di campionamento utilizzato è di

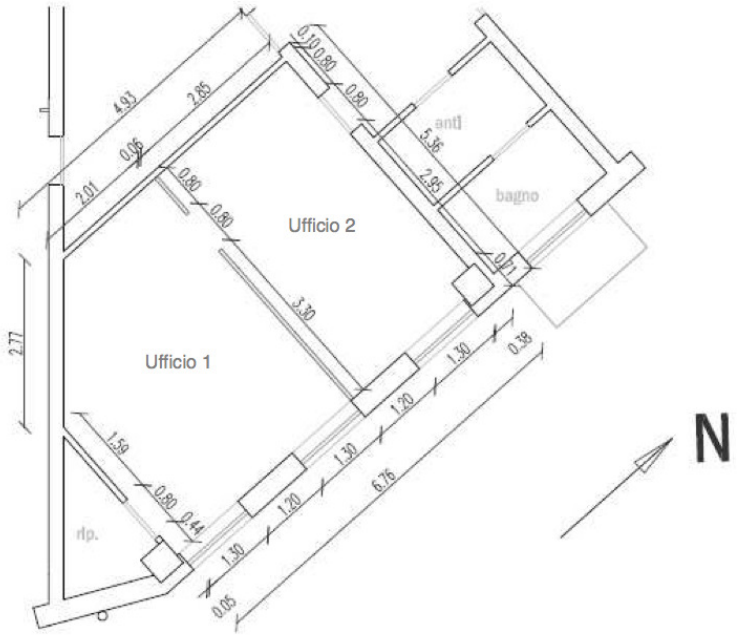


Figura 14.1: Mappa dell'edificio utilizzato per la raccolta dei dati.

8 minuti, un periodo più che sufficiente per le dinamiche termiche di un edificio. Precedenti lavori (si veda [4]) hanno dimostrato che questo non è un parametro critico di progettazione e che anche valori superiori (10-15 minuti) danno risultati soddisfacenti. Una volta ottenuti i dati (seguendo la procedura descritta nella

Sezione **??**), un primo controllo che si è effettuato per valutarne l'integrità è stato quello della tensione di alimentazione. Il grafico in Figura 14.2 mostra l'andamento di tutti e dodici i sensori. Si è visto subito che una misurazione è da scartare poichè la tensione si era abbassata ad un livello non accettabile. Per questo motivo si è deciso di scartare quel sensore.

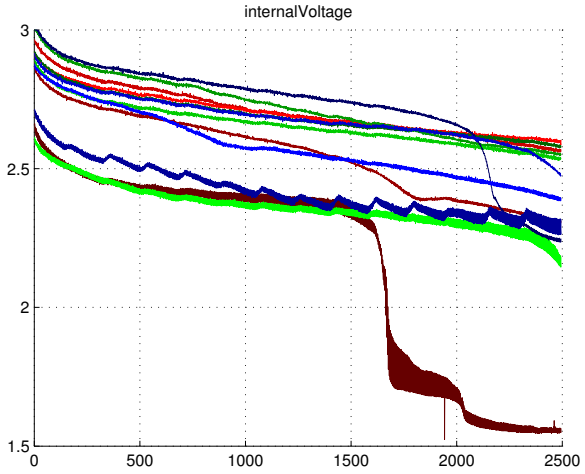


Figura 14.2: Andamento della tensione di alimentazione

Guardando il grafico delle temperature misurate (Figura 14.3) si è osservata la presenza di un rumore anomalo verso la fine dei dati, per circa $n \geq 1750$. Confrontando questo fenomeno con l'andamento delle tensioni se ne capisce immediatamente il motivo; alcuni sensori fin dall'inizio partivano con batterie non pienamente cariche e questo ha fatto si che la loro autonomia fosse minore degli altri. Per non scartare troppi sensori si è scelto di troncare i dati utilizzati. I dati rimanenti sono stati suddivisi in due insiemi dove il primo è stato utilizzato per l'identificazione ed il secondo come set di dati "freschi" per la validazione.

Per eseguire il pre-filtraggio dei dati si è eseguito la classica rimozione della media per evitare che ci siano errori dovuti al bias in fase di identificazione. Analizzando poi il grafico dell'intensità luminosa registrata ci si è accorti che i dati nel complesso presentavano ordini di grandezza troppo differenti (la luminosità è sull'ordine dei $10^5\, lux$). Per evitare problemi di condizionamento numerico degli algoritmi si è preferito effettuare una standardizzazione dei dati togliendo a ciascuna misura la sua varianza. Questi due procedimenti sono leciti poichè sono due informazioni che possono essere facilmente reintrodotte alla fine sommando la media e moltiplicando per la deviazione standard i dati in uscita dalle identificazioni. In Figura 14.4 si può vedere un esempio di come si presentano i dati pronti per la fase di identificazione.

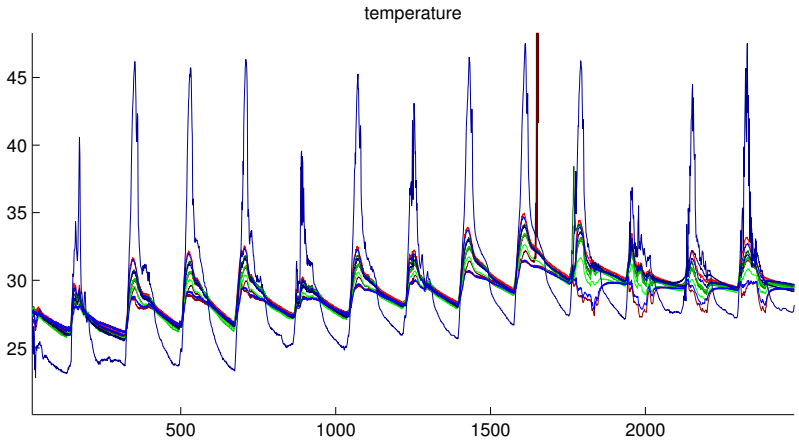


Figura 14.3: Andamento della temperature misurate (quella maggiore si riferisce al sensore posizionato all'esterno). Si noti il rumore finale e la misura del sensore avariato che esce dal range illustrato (arriva fino a 7000 gradi).

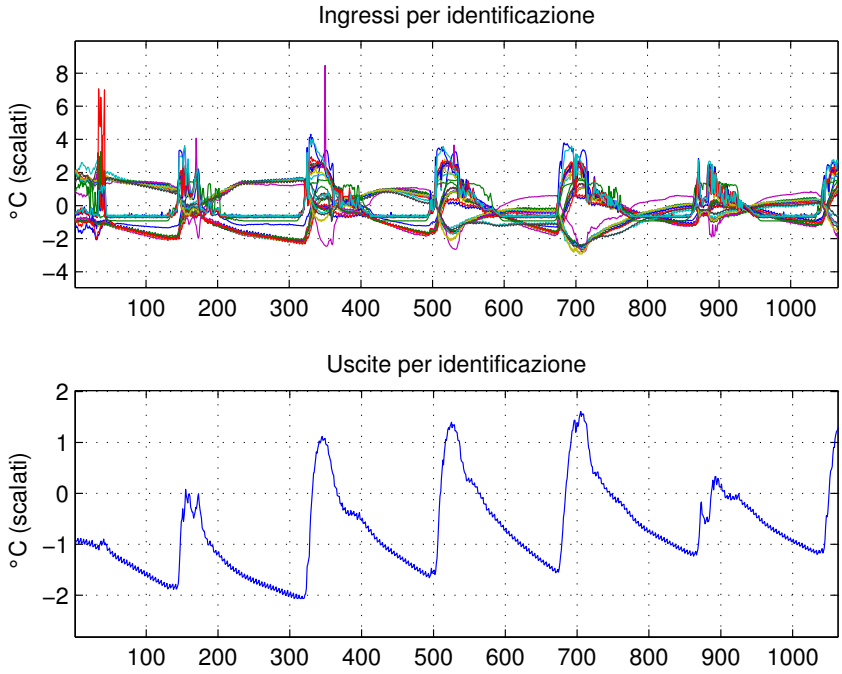


Figura 14.4: Esempio di dati pronti per l'identificazione dell'uscita relativa al terzo sensore.

14.1 Risultati ottenuti con PEM

Il metodo PEM è stato testato con un vettore di dati di lunghezza variabile da 600 fino ad 1500 campioni. Si ricorda che il tempo di campionamento utilizzato è stato di 8 minuti, cioè per l'identificazione si è utilizzato un numero di dati che si sarebbe ottenuto come se l'acquisizione fosse durata da un minimo 3 ad un massimo di 8 giorni. I dati rimanenti sono stati lasciati per la validazione. La classe a cui fa riferimento il metodo è il seguente modello vettoriale:

$$Ay(t) = B\mathbf{u}(t-1) + Ce(t) \tag{14.1}$$

dove $\mathbf{u}$ è il vettore contenente i 32 ingressi del sistema, A, C sono polinomi di ordine m e B è un vettore riga di cui ogni elemento è un polinomio anch'esso di ordine m che consta quindi di $32 \times m$ elementi. Per la scelta della dimensione delle matrici si è scelto di utilizzare gli indici di AICC e BIC (vedi (3.1) e (3.2)) avendo impostato un limite superiore per l'ordine dei modelli ARMAX pari a 15. Confrontando i risultati si è notato nel caso in cui i FIT sono elevati i due metodi danno risultati simili. Spesso però succede che laddove AICC porta ad avere modelli non accettabili (FIT molto bassi o negativi, cioè instabilità) BIC dà buoni risultati. In Tabella 1 viene riportato un esempio di questo fenomeno.

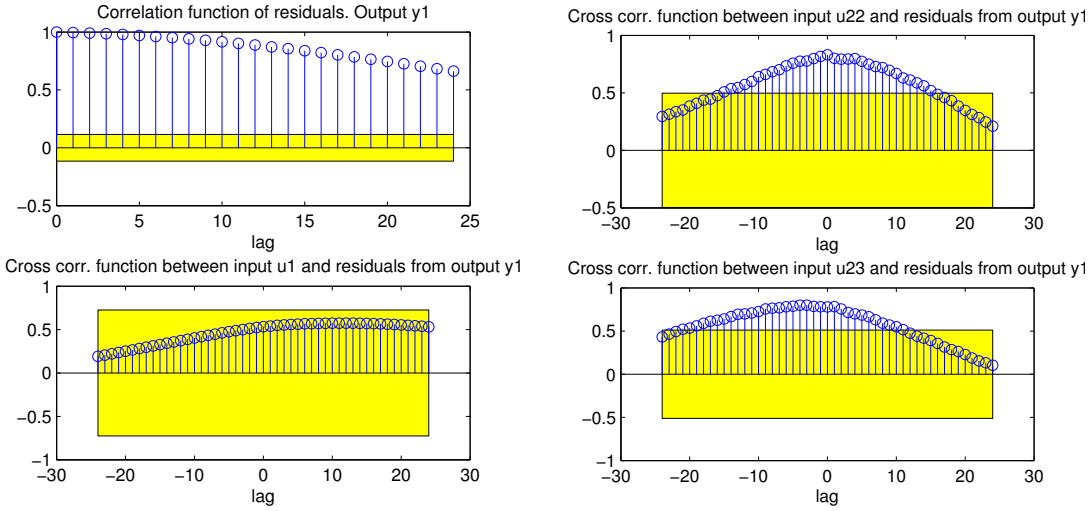
	AICC	BIC
n=600	-0.37	-0.64
n=800	-7.64	0.98
n=1100	-0.96	0.99
n=1500	0.99	0.98

Tabella 1: Confronto tra i risultati ottenuti con PEM+AICC e PEM+BIC in termini di COD (valore medio nei primi 30 passi di predizione) avendo scelto come uscita il terzo sensore.

Dalla Tabella 1 si vede inoltre come varino i risultati in funzione del numero n di campioni in ingresso. Per $n = 600$ quasi la metà dei modelli (nel caso migliore BIC) sono instabili e una buona parte dei rimanenti non ha prestazioni di rilievo. Questo avviene perchè i dati a disposizione sono troppo pochi per stimare l'elevato numero di parametri che presenta questo sistema [4]; l'algoritmo, non facendo alcuna selezione degli ingressi (e delle relative funzioni di trasferimento da stimare), non è così in grado di far fronte a situazioni di questo tipo. Il fatto che la stima effettuta sia pessima si può evincere anche guardando il test dei residui. A titolo di esempio si porta la Figura 14.5, in cui si vede un residuo decisamente non bianco indice che gran

[4]Come già detto l'ordine massimo imposto è stato di 15, in generale però l'ordine dei modelli è stato selezionato fra il 5 e il 10, cioè si hanno da 170 fino a 340 parametri di cui deve essere effettuata una stima.

parte dell'informazione non è stata catturata. Il test di indipendenza dell'uscita con gli ingressi da esiti che in generale sono positivi con quasi tutte le correlazioni che stanno all'interno della banda di confidenza (rappresentata in giallo in Figura 14.5 (a)). Da questo test si trova inoltre conferma all'utilizzo di dati normalizzati, infatti per i dati non normalizzati il test di indipendenza presenta molto spesso valori di cross-correlazione esterni alla banda di confidenza (un esempio è riportato in Figura 14.5 (b)). Dei risultati analoghi si ottengono valutando il test dei residui per ogni altro algoritmo utilizzato, per questo nel proseguo non verrà riportato.
All'aumentare del numero di dati disponibili le prestazioni di PEM aumentano, non di molto per $n = 800$, decisamente meglio per valori di n superiori al migliaio con valori di FIT che si attestano attorno al 98% e di COD sul 0.99; la PEM messa nelle condizioni più favorevoli conferma così il suo carattare di ottimalità.



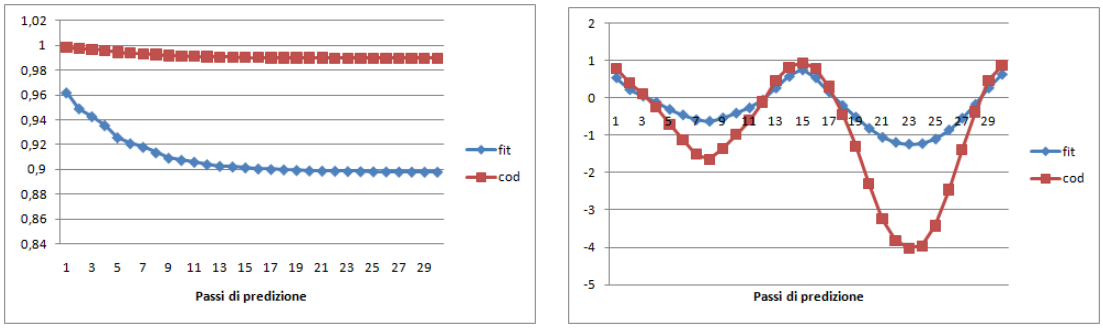
(a) Test di bianchezza

(b) Test di indipendenza

Figura 14.5: Test di bianchezza effettuato con l'uscita nel terzo sensore e particolare del test di indipendenza sul settimo sensore.

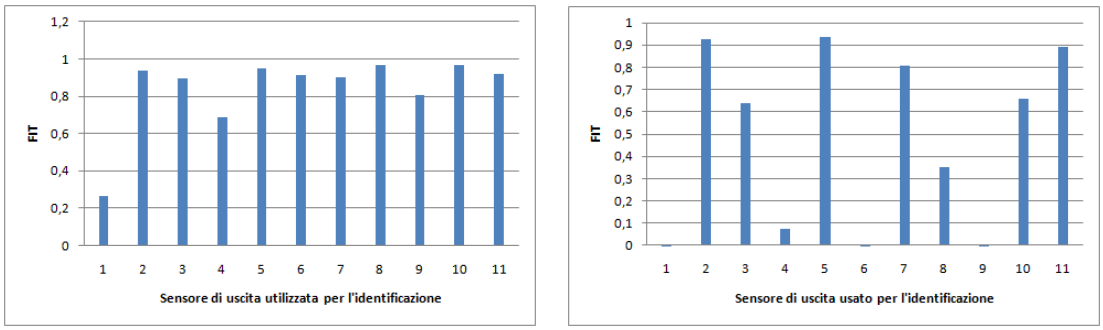
Nell'utilizzare i risultati per effettuare la predizione dell'evoluzione dell'uscita è utile osservare come variano gli indici FIT e COD. Ciò che si verifica è che le prestazioni peggiorano nei primi passi di predizione e poi si stabilizzano da un certo valore di k in poi. Da quell'istante in poi si deve parlare di simulazione invece che di predizione, le uscite del modello non sono più influenzate dalle uscite degli istanti passati e quindi è come se il predittore lavorasse in catena aperta. In Figura 14.6 ne è riportato un esempio per il terzo sensore (per le altre uscite si ottengono risultati analoghi). In questo caso l'andamento si stabilizza dopo una decina di passi per il modello ben identificato (ottenuto con BIC) e continua ad oscillare per il modello non affidabile (ottenuto con AICC). Nella successiva Figura 14.7 si trova

la valutazione del risultato in termini di FIT per le identificazioni di ogni sensore con $n = 1100$ campioni, è evidente la differenza delle prestazioni di BIC e AICC.



(a) PEM+BIC (b) PEM+AICC

Figura 14.6: Andamento della predizione per i modelli ricavati con PEM e i due indici di stima dell'ordine per il sensore numero 3 con $n = 1100$.



(a) PEM+BIC (b) PEM+AICC

Figura 14.7: FIT per ogni sensore di uscita con $n = 1100$.

Per rendersi conto della differenza fra l'uscita reale e quella che si ricava facendo la predizione di 10 passi in avanti si riporta la Figura 14.8. Ottenuta rapidamente dall'interfacci si da qui è possibile vedere che con FIT superiore al 90% (in questo caso 93%) si hanno errori medi minori di 2 decimi di grado (in particolare in questo caso si ha $RMS = 0.16$ e $COD = 0.97$).

14.2 Risultati ottenuti con N4SID

L'algoritmo ai sottospazi N4SID è stato testato al variare del numero di dati in ingresso, si è scelto di utilizzare dai 600 ai 1000 campioni. Per quanto riguarda l'ordine del modello la scelta è stata affidata ad una routine interna all'algoritmo implementato dalla funzione `n4sid` di Matlab. Questo metodo di identificazione

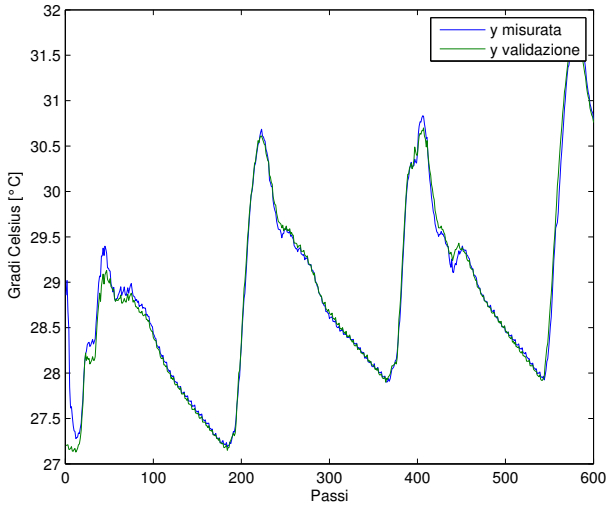


Figura 14.8: Confronto fra uscita reale e predetta a 10 passi con modello ricavato da PEM+BIC.

si è mostrato essere computazionalmente il più efficiente. I tempi di calcolo sono stati in assoluto i minori (dell'ordine di pochi secondi) ottenendo delle prestazioni di poco peggiori rispetto alla PEM. All'aumentare del numero di dati in generale le prestazioni migliorano ma non è stato così per ogni sensore, si è deciso quindi di selezionare di volta in volta i modelli migliori ottenendo i FIT riportati in Figura 14.9, complessivamente volendo fare una media dei migliori fit si trova un valore di 69.08.

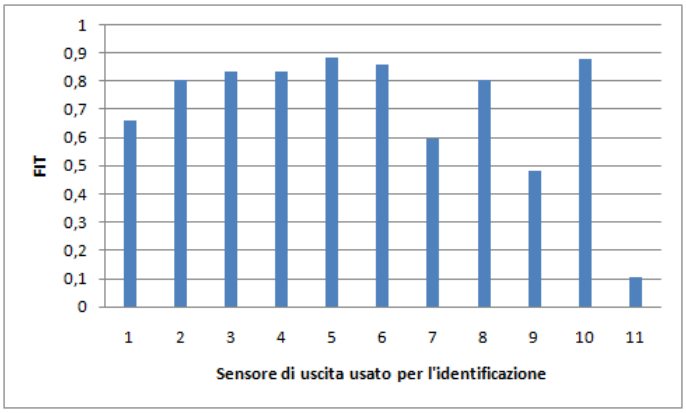


Figura 14.9: Prestazioni migliori per ogni sensore ottenute con n4sid.

In Figura 14.10 viene mostrato l'andamento degli indici FIT e COD in funzione

dei passi di predizione.

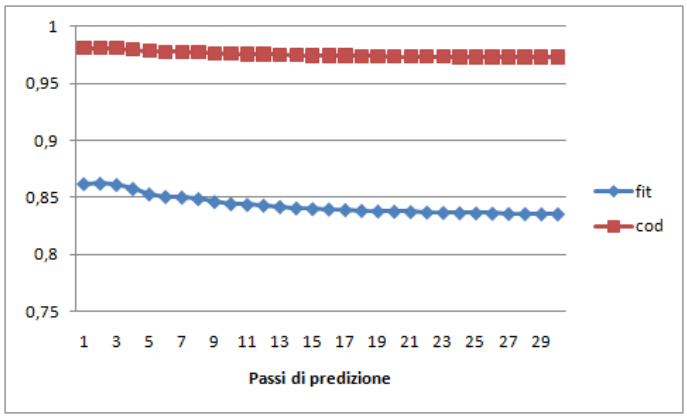


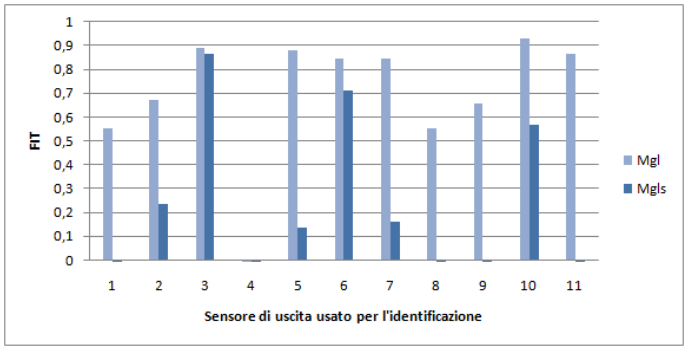
Figura 14.10: Indici di FIT e COD in funzione dei passi di predizione.

Anche in questo grafico si osserva lo stesso fenomeno di stabilizzazione degli indici all'aumentare dei passi k di predizione riportato per la PEM.

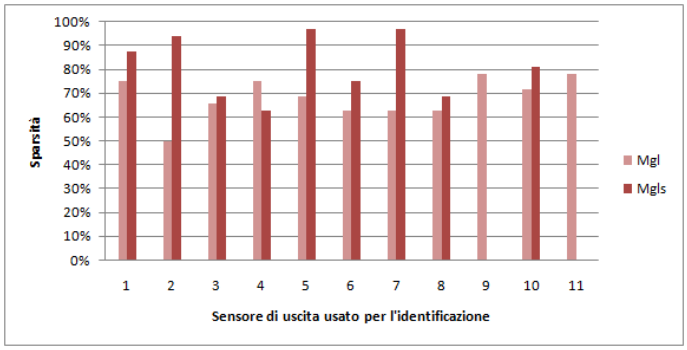
14.3 Risultati ottenuti con GLAR

Come spiegato in Sezione **??** non è immediato trovare il modo di scegliere il numero di sensori che l'algoritmo GLAR deve scartare. Per questo viene usato un ciclo che ad ogni passo aumenta il numero di sensori che deve utilizzare GLAR calcolando anche ad ogni passo l'RMS utilizzando un set di validazione. Alla fine viene tenuto in considerazione solo il modello che ha dato il miglior RMS sul set di validazione. Sono stati provati diversi n ma le simulazioni hanno dato spesso errore. E' stato scelto un valore di $n = 500$ che ha dato risultati stabili per tutte le simulazioni.
Viene creato anche un secondo modello Mgls con l'idea di aumentare ulteriormente la sparsità applicando un'altra volta l'algoritmo ma tenendo in considerazione solo i sensori usati prima. Si è notato però (ad esempio vedi Figura 14.11) (a) che con l'ultima passata spesso non si ottengono dei modelli in grado di descrivere il sistema. In Figura 14.11 (b) viene riportata la situazione di sparsità ricavata con i due differenti modelli. Nel secondo caso la sparsità è effettivamente maggiore ma solo nei casi di modelli ben identificati il che la rende difficilmente utilizzabile in generale.

Il comportamento dei modelli in funzione dei passi di predizione viene riportato in Figura 14.12 solo per il metodo che ha dato i risultati migliori (Mgl) e presenta un andamento regolare.



(a) FIT di GLAR



(b) Sparsità di GLAR

Figura 14.11: FIT (a) e sparsità (b) raggiunta dell'algoritmo GLAR con $n = 500$ dati in ingresso. A confronto la soluzione ottima (Mgl) e sub-ottima (Mgls).

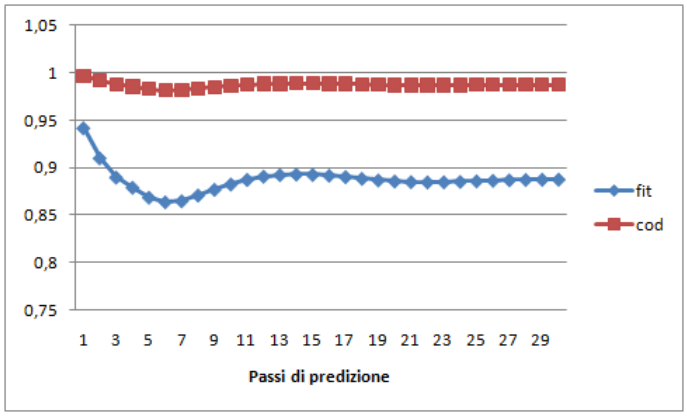
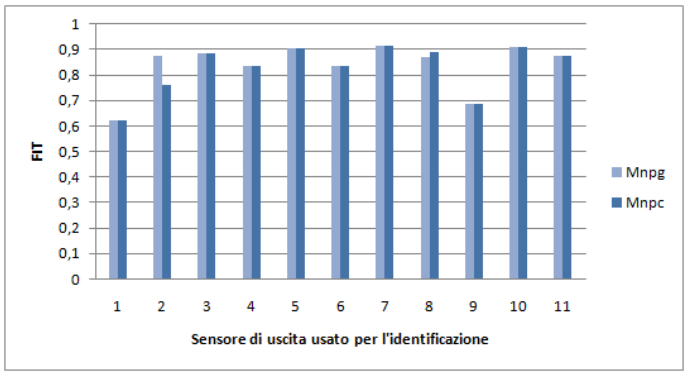


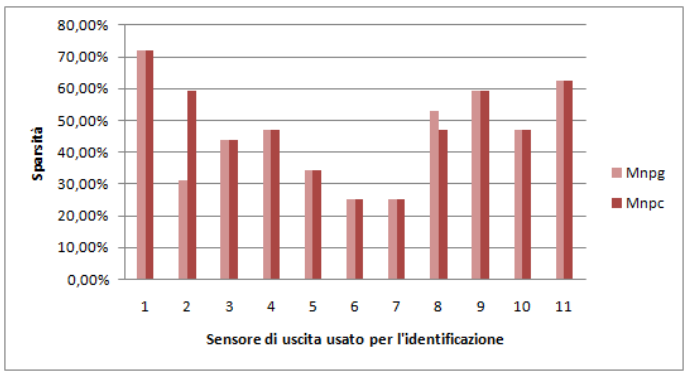
Figura 14.12: Indici di FIT e COD in funzione dei passi di predizione.

14.4 Risultati ottenuti con SSGLAR

L'algoritmo SSGLAR produce in uscita due modelli dinamici. Il primo rappresenta la soluzione ottima eseguendo i passi descritti in sezione 8 mentre il secondo rappresenta una soluzione sub-ottima che cerca di aumentare la sparsità tollerando un peggioramento del 2% delle prestazioni (RMS calcolato su un insieme di validazione).
Come per GLAR sono stati provati diversi n che però in simulazione hanno dato spesso errore. E' stato scelto un valore di $n = 500$ che anche qui ha dato risultati stabili per tutte le simulazioni.
Le prestazioni che si ottengono in termini di FIT sono illustrate a confronto in Figura 14.13 (a) mentre in Figura 14.13 (b) si confrontano le sparsità; solo per il secondo sensore si riscontra un aumento della sparsità significativo (che passa dal 30% al 60% di funzioni di trasferimento messe a zero) con un lieve peggioramento della prestazione. In tutti gli altri casi le prestazioni si mantengono pressochè identiche. Da questi risultati sembra dunque che il metodo sub-ottimo sia da preferirsi all'ottimo per sistemi di queste dimensioni.



(a) FIT di SSGLAR



(b) Sparsità di SSGLAR

Figura 14.13: FIT (a) e sparsità (b) raggiunta dell'algoritmo SSGLAR con $n = 500$ dati in ingresso. A confronto la soluzione ottima (Mnpg) e sub-ottima (Mnpc).

Il comportamento predittivo dei modelli ottenuti si comporta sostanzialmente con un andamento simile a quello riportato in Figura 14.14 ed è del tutto simile tra le due soluzioni trovate.

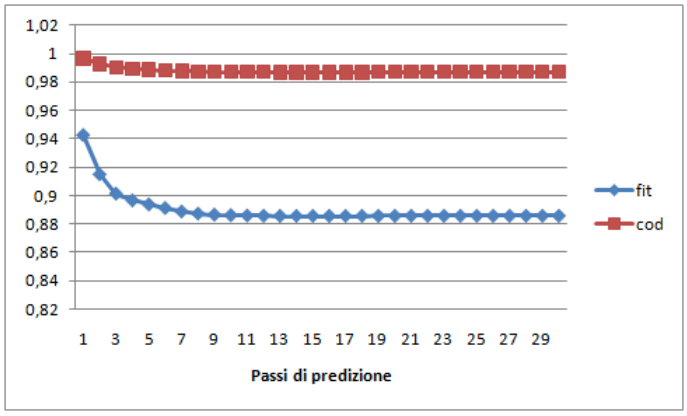


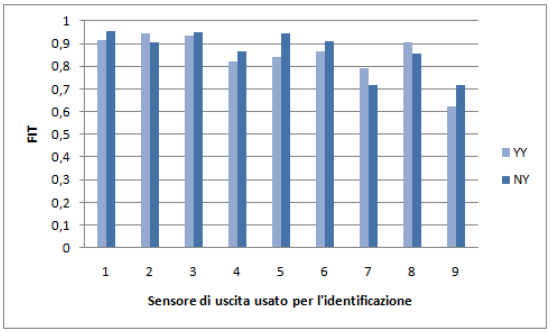
Figura 14.14: Indici di FIT e COD in funzione dei passi di predizione.

14.5 Risultati ottenuti con SSEH

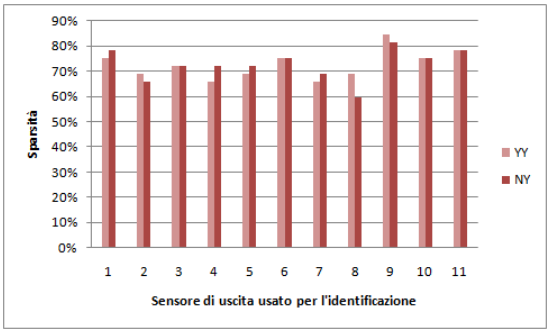
In questo caso sono stati utilizzati due classi di modelli differenti, la prima in cui viene modellizzata la parte autoregressiva (modello ARMAX, nel seguito indicato con yy) e la seconda in cui questa parte manca (modello Output Error, ny). Usando il modello ny per fare predizione si osserva che le prestazioni non hanno transitorio al cambiare del numero di passi k di predizione. Questo fenomeno si presenta, perché non presentando il modello ny la parte autoregressiva, nel fare predizione non viene tenuto conto delle uscite passate e quindi equivale a fare solo simulazione. Questa è la differenza principale delle due classi di modelli, in termini di FIT e sparsità infatti si ottengono risultati simili come si vedrà dalle figure seguenti.
Essendo l'algoritmo da cui teoricamente ci si aspettano le prestazioni migliori si è deciso di prova a testarlo in condizioni più critiche rispetto ai precedenti. Si è scelto così una lunghezza di dati diminuita fino a 100 campioni (cioè circa dodici ore di acquisizione) aumentata poi fino a 400 a passi di un centinaio. Si fornisce ora una descrizione dell'andamento ottenuto nei due casi estremi e poi si passa ad un confronto da un punto di vista della predizione.
Per $n = 100$ e $p = 20$ (lunghezza delle risposte impulsive trovate) il FIT e la sparsità ottenuti sono visibili in Figura 14.15.

Per $n = 400$ e $p = 20$ (lunghezza delle risposte impulsive trovate) invece i risultati sono stati come da Figura 14.16

Un particolare che evidenzia come con n piccoli si ottengono delle percentuali di sparsità molto elevate è riportato nella Tabella 2.



(a) FIT di SSEH

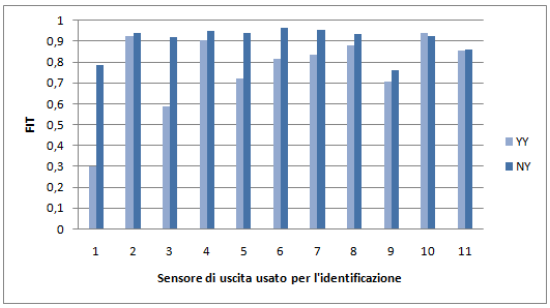


(b) Sparsità di SSEH

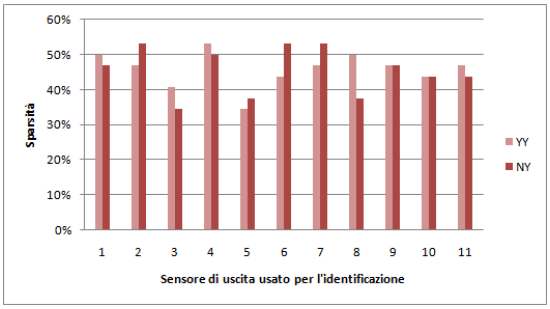
Figura 14.15: FIT (a) e sparsità (b) raggiunta dell'algoritmo SSEH con $n = 100$ e $p = 20$. A confronto la soluzione con modello yy e ny.

n	100	200	300	400
FIT yy	0.91	0.92	0.93	0.93
Sparsità [%] yy	72	60	50	41
FIT ny	0.95	0.93	0.93	0.92
Sparsità [%] ny	72	50	40	34

Tabella 2: FIT e sparsità ottenute per la predizione di un passo nell'identificare l'uscita del terzo sensore al variare di n.



(a) FIT di SSEH



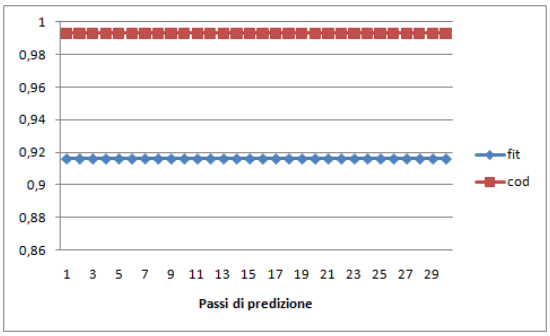
(b) Sparsità di SSEH

Figura 14.16: FIT (a) e sparsità (b) raggiunta dell'algoritmo SSEH con $n = 400$ e $p = 20$. A confronto la soluzione con modello yy e ny.

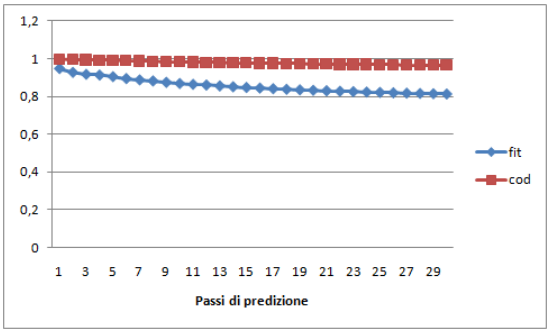
Come viene mostrato successivamente con n piccoli gli algoritmi forniscono modelli che entrano in simulazione da subito anche usando modelli yy (presentano un FIT costante al variare del passo di predizione). In altre parole le predizioni che vengono effettuate con questi risultati non tengono conto delle uscite passate, e dunque sono di fatto delle simulazioni. La differenza fra due andamenti del genere si trova in Figura 14.17 dove a sinistra si vede che non ci sono variazioni all'aumentare di k e invece a destra si vede che il FIT diminuisce solo nei primi passi prima di stabilizzarsi ed entrare in simulazione. In generale si è notato che a partire da circa $n = 200$ iniziano ad esserci modelli yy che presentano transitori del FIT. Questi transitori aumentano all'aumentare del numero di campioni n.

Nell'identificazione SSEH si è cercato anche di capire come cambiano le prestazioni dei modelli al cambiare del parametro p (lunghezza delle risposte impulsive identificate). Nei risultati sono stati notati cambiamenti significativi cambiando p in termini di prestazioni. Non si può però affermare che all'aumentare di p le prestazioni migliorano, ma sembra che queste cambino in modo aleatorio al cambiare di p. In Figura 14.18 è riportato un esempio in cui si conferma quanto appena detto. Una soluzione a questo problema è di scegliere al variare di p il modello che ha dato i miglior prestazioni calcolati su un set di validazione, detto in altri termini fare una stima di p.

In Figura 14.19 viene riportato un particolare della selezione effettuata dall'al-



(a) $n = 100$



(b) $n = 400$

Figura 14.17: Andamento di FIT e COD con l'identificazione SSEH al variare del passo di predizione usando rispettivamente $n = 100$ e $n = 400$ campioni. Nei due casi si sono utilizzati i modelli yy con $p = 20$ per il terzo sensore.

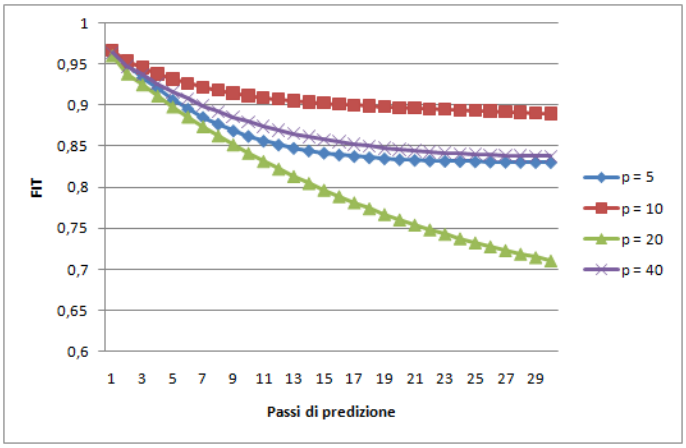


Figura 14.18: Indici di FIT in funzione dei passi di predizione con $n = 300$ e diversi valori di p per il terzo sensore come uscita.

goritmo nell'identificazione della temperatura in uscita del sensore 8 (evidenziato in figura). Essendo una misura di temperatura ci si potrebbe aspettare che i dati che assumono maggior rilievo siano gli andamenti delle temperature circostanti. In verità se si riflette sul funzionamento dell'algoritmo e sul fatto che l'ambiente circostante subisce poche variazioni si capisce che viene presa la prima misura di temperatura e poi tutte le altre vengono scartate perchè molto correlate con la prima. Una volta presa la prima le loro proiezioni sullo spazio attivo saranno molto piccole, indice del fatto che non portano informazione e che vanno scartate. Lo stesso probabilmente avviene per l'umidità differente è invece la questione per la luminosità. Rifrazioni della radiazione luminosa sui muri e orientamento dei sensori porta a misure che sono tra loro incorrelate e vengono quindi selezionate per effetturare la stima.

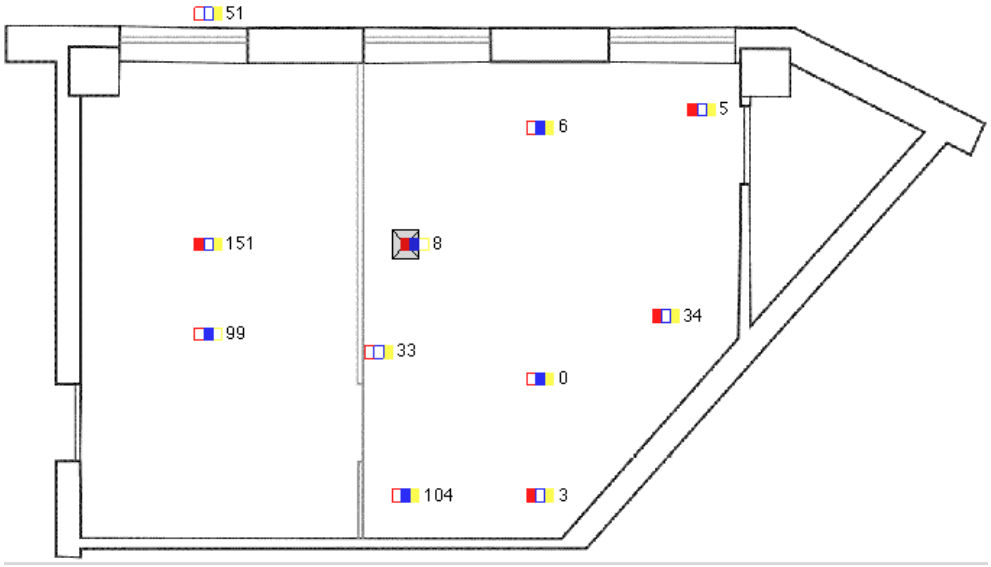


Figura 14.19: Sensori scelti dal modello

14.6 Confronti

Il confronto fra le prestazioni ottenute con i vari metodi è riportato in Figura 14.20. Dalla figura si vede che spesso il metodo PEM produce i modelli più accurati richiedendo però un elevato numero di dati in ingresso (in figura i risultati riportati sono per $n = 1100$). Questo è un punto debole dell'algoritmo che, come si è già detto, con pochi dati rispetto ai parametri non è adeguatamente accurato. Da questo punto di vista il metodo SSEH è senza dubbio il migliore e non ha eguali fra gli algoritmi utilizzati (ha funzionato anche con $n = 100$). Anche gli altri algoritmi a gruppi (GLAR, SSGLAR) hanno bisogno di relativamente pochi dati rispetto ad PEM ed N4SID.

Se si confronta l'onere computazionale invece il migliore è stato N4SID con tempi di calcolo di qualche secondo per la singola identificazione. La PEM ha richiesto più tempo (meno di una decina di minuti per sensore), dovuto soprattutto alla stima dell'ordine che richiede più iterazioni. Gli algoritmi che favoriscono la sparsità invece da questo punto di vista sono peggiori. Richiedono tempi di calcolo che dipende molto dal numero di campioni (e per SSEH dalla lunghezza delle risposte impulsive). Per n elevati facilmente arrivano a superare i tempi di calcolo di un'ora per ogni singolo sensore.

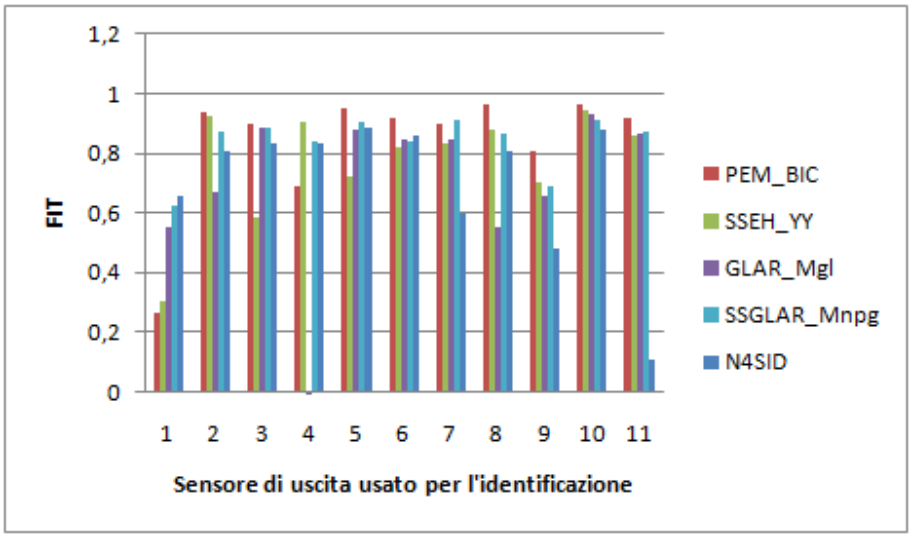


Figura 14.20: Confonto fra i FIT ottenuti utilizzando i modelli migliori dei metodi PEM, N4SID, GLAR, SSGLAR, SSEH.

I risultati sulla sparsità sono riportati nella Figura 14.21. In questo caso si è usata una $n = 500$ per GLAR e SSGLAR e $n = 400$ per SSEH. Dalla figura sembra che il GLAR dia i migliori risultati si è però visto come per esempio nel caso di SSEH questi risultati dipendano fortemente dai parametri con cui si impostano gli algoritmi. Quindi il significato di questa figura è relativo al particolare settaggio degli algoritmi, che dovranno essere scelti anche in funzione della sparsità che si vuole ottenere.

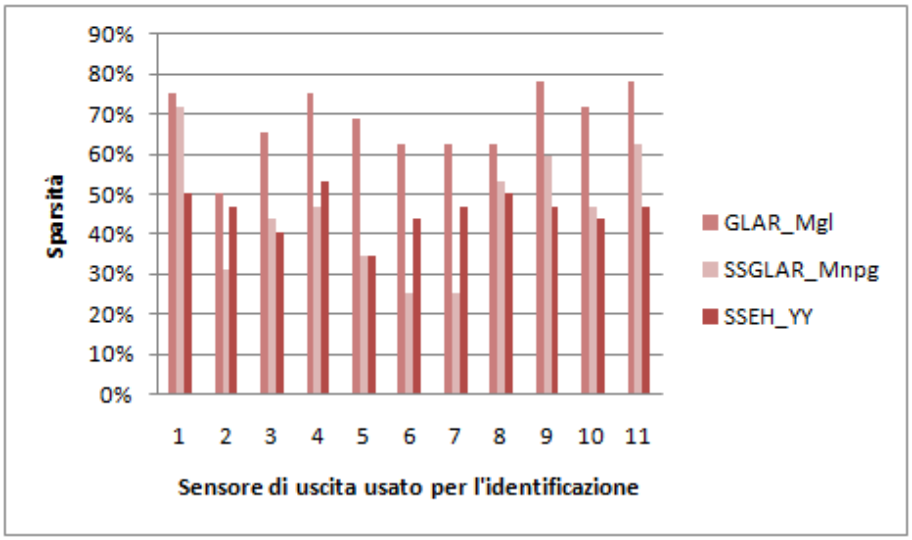


Figura 14.21: Confronto della sparsità ottenuta con i metodi che GLAR, SSGLAR, SSEH.

15 Sviluppi futuri

Uno sviluppo che si potrebbe effettuare con questi algoritmi è quello dell'estensione dell'intervallo di raccolta dei dati ad un periodo più ampio. Vista la difficoltà di implementare una campagna di acquisizione con i motes per lunghi periodi (cruciale è il problema dell'alimentazione che non può essere effettuata tramite usb per il surriscaldamento) sarebbe opportuno effettuare brevi campagne in differenti periodi dell'anno. Si presti attenzione però al fatto che una simile impostazione della campagna di acquisizione porta con sè una nuova problematica. Il comportamento dinamico dell'edificio in situazioni molto differenti potrebbe non essere lo stesso a causa di forti non linearità degli elementi costruttivi. L'identificazione in sostanza viene effettuata attorno a punti di lavoro differenti ed un detrend dei dati rischia di eliminare dell'informazione fondamentale.

Il livello di sparsità che si riscontrato è in media del 40-50%, valore che ci si aspetta più basso all'aumentare delle dimensione dell'edificio e della indipendenza della temperatura fra i vari ambienti. Inoltre i dati raccolti nel periodo temporale hanno avuto poca variabilità e quindi probailmente la dinamica che è stata catturata è stata limitata. Sarebbe interessante comunque vedere il comportamento di questi modelli con dati presi in condizioni differenti per vedere se la sparsità indotta dalle tecniche utilizzate ha portato con sè anche una maggior robustezza del modello. E' chiaro infatti che un modello 'pieno', identificato ad esempio con PEM, risente molto delle variazioni degli ingressi che non erano necessari per l'identificazione. La non selezione dunque non aiuta la fase di identificazione e può invece portare a poca robustezza del modello poichè tutto ciò che è non è stato eliminato con una selezione arreca disturbo.

Come suggerito nelle conclusioni per il futuro sarebbe interessante organizzare lunghe campagne di raccolta dati in grandi edifici per poter, individuare con maggior determinazione i limiti degli algoritmi. Sarebbe interessante vedere non solo come la qualità di questa identificazione sia sufficientemente vicina al modello reale, ma anche se questi modelli siano validi o meno per utilizzarli per controllare la dinamica dell'edificio stesso. Infatti sarebbe vedere se l'errore di identificazione risulta sufficientemente piccolo ai fini del controllo termico.

Inoltre sarebbe interessante durante la raccolta dati usare il pilotaggio degli attuatori per raccogliere informazioni sulle dinamiche di controllo, in maniera da non avere solo dati il cui ingresso esogeno è la temperatura esterna.

16 Conclusioni

Nello svolgimento del progetto i problemi principali che si sono riscontrati sono stati dovuti principalmente alla mancanza di background teorico che ha portato a difficoltà iniziali nel comprendere il principio di funzionamento degli algoritmi ed un grande dispendio di tempo per colmare i gap.
Si sottolinea come l'elevato tempo richiesto dagli algoritmi per fornire dei risultati. Per eseguire gli algoritmi per tutte le uscite ci sono voluti più giorni e questo ha portato ad una complessa pianificazione a priori del lavoro da svolgere.
Il confronto con i risultati ottenuti dai progetti degli anni scorsi non è stato fatto perchè i dati, gli approcci e anche gli obbiettivi erano differenti. La loro selezione dei sensori da utilizzare è stata effettuata con diverse tecniche a priori mentre in questo elaborato questo viene eseguito in maniera automatica man mano che l'algoritmo procede.

Ci si ritiene soddisfatti del lavoro eseguito perchè ha dato modo di comprendere e lavorare con problematiche di identificazione. I nuovi algoritmi hanno funzionato bene in situazioni critiche in cui si hanno pochi dati in relazione ai parametri da stimare, cosa che non sono riusciti a fare le metodologie tradizionali. I modelli che hanno fornito sono stati fisicamente ben interpretabili in quanto hanno effettivamente messo in rilievo la struttura sottostante ai dati raccolti. Per questo motivo si è deciso di implentare una interfaccia grafica che agevoli questo procedimento quando la mole di dati da analizzare è imponente. Si è impostato il lavoro in termini di riutilizzabilità del software sicuri che in futuro questi metodi verranno utilizzati. La presenza quindi di software che sia chiaro ed efficiente è il presupposto per poter procedere con gli sviluppi dell'identificazione termica di edifici.

17 Ringraziamenti

Si ringrazia il Prof. Alessandro Chiuso per avere fornito gli algoritmi e la disponibilità dimostrata nei nostri confronti. Si ringrazia il Prof. Luca Schenato per la disponibilità e la presenza.

Riferimenti bibliografici

[1] Giorgio Picci. *Metodi statistici per l'identificazione di sistemi lineari*. 2011.

[2] Alessandro Chiuso and Gianluigi Pillonetto. Identification of large-scale sparse linear dynamic systems: a regularization based approach. 2010.

[3] Lennart Ljung. *System identification - theory for the user*. 1999.

[4] Federico Bonollo, Daniele Cattai, and Marco Verga. Identificazione termodinamica di un edificio con l'utilizzo di pca e pls. 2007-2008.

[5] Nachman Aronszajn. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 1950.

[6] Alessandro Chiuso and Gianluigi Pillonetto. Nonparametric sparse estimators for identification of large scale linear systems.

[7] Alessandro Chiuso and Gianluigi Pillonetto. Learning sparse dynamic linear systems using stable spline kernels and exponential hyperpriors.

[8] Brandly Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *Journal of the Royal Statistical Society*, 2004.

[9] Gianluigi Pillonetto and Giuseppe De Nicolao. A new kernel - based approach for linear system identification. *Automatica*, 2010.

[10] Torsten Söderström and Peter Stoica. *System identification*. Prentice Hall, 1989.

[11] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society*, 1996.

[12] Elena Toffoli, Giancarlo Baldan, and Guido Albertin. Identificazione termodinamica di un edificio. 2006-2007.

[13] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society*, 2006.

[14] Sensirion AG. *Datasheet SHT1x - Humidity and temperature sensor*, 2010.

[15] Moteiv Corporation. *Tmote sky - humidity, light and temperature sensor with USB*, 2006.