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

Relazione sul IV laboratorio

Andrea Barazzuol 601399 barazzu1@dei.unipd.it

Markus Ausserer

Riccardo Alberton

Padova, 25 Febbraio 2011

Indice

1 Raccolta dati

1.1 Impostazione

Per eseguire le campagne di racca

1.2 I dispositivi utilizzati

Per eseguire le misurazioni delle grandezze di interesse sono stati utilizzati i dispositivi equipaggiati *MoteIV Tmote Sky* 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 (1024 Kb, sufficiente per tutto il tempo di raccolta) in cui 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 diminuisce eccessivamente posso 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 ??? è riportato l'andamento della tensione di ogni dispositivo.
Viene riportata qui di seguito una breve descrizione dei sensori utilizzati dai *Tmote*:

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

$$T = b_1 + b_2 T_{raw} - b_3 (T_{raw} - b_4)^2 \quad [C] \tag{1.1}$$

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:

$$RH_{true} = (T - 25)(c_1 + c_2 H_{raw}) + c_3 + c_4 H_{raw} + c_5 H_{raw}^2 [\%] \quad (1.2)$$

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 ???.

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:

$$AR = 625 AR_{raw} \quad [lux] \quad (1.3)$$

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

$$TR = 76.9 TR_{raw} \quad [lux]\,. \quad (1.4)$$

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.

4. *Codice utilizzato* Si fornisce qui di seguito una breve descrizione del codice che viene utilizzato per la gestione dei mote.

Il firmware caricato su ogni mote divide la sessione di aquisizione in tre fasi. 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 1.1.
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 (1.2)-(1.4). La programmazione dei mote è avvenuta eseguendo lo script $EggMote.h$ in cui si precedentemente impostato il tempo di campionamento.

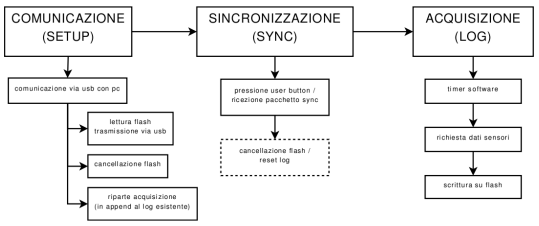


Figura 1.1: Struttura degli stati e delle funzionalità del mote

2 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).

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

$$\begin{cases} x(t+1) &= Ax(t) + Bu(t) + Ke(t) \\ y(t) &= Cx(t) + e(t) \end{cases} \tag{2.1}$$

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:

$$Y_{[t,s]} = \begin{bmatrix} y_t & y_{t+1} & \cdots & y_{t+N-1} \\ y_{t+1} & y_{t+2} & \cdots & y_{t+N} \\ \vdots & \vdots & \ddots & \vdots \\ y_s & y_{s+1} & \cdots & y_{s+N-1} \end{bmatrix} \tag{2.2}$$

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

$$Y_{[0,2t-1]} = \begin{bmatrix} Y_p \\ Y_f \end{bmatrix} = \begin{bmatrix} Y_{[0,t-1]} \\ Y_{[t,2t-1]} \end{bmatrix} \tag{2.3}$$

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:

$$\begin{cases} \hat{X}_{t+1} &\simeq A\hat{X}_t + B\hat{U}_t + KE_t \\ Y_t &\simeq C\hat{X}_t + E_t \end{cases} \tag{2.4}$$

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

$$\Gamma := \begin{bmatrix} C \\ CA \\ CA^2 \\ \vdots \\ CA^f \end{bmatrix} \tag{2.5}$$

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

$$H_d = \begin{bmatrix} 0 & 0 & \ldots & 0 \\ CB & 0 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{f-1}B & CA^{f-2}B & \ldots & 0 \end{bmatrix} \quad , \tag{2.6}$$

$$H_s = \begin{bmatrix} I & 0 & \dots & 0 \\ CK & I & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ CA^{f-1} & CA^{f-2}K & \dots & I \end{bmatrix} \quad . \tag{2.7}$$

Utilizzando ora la relazione

$$\mathbf{Y}_{[p,p+f]} = \Gamma \mathbf{X}_p + H_d \mathbf{U}_{[p,p+f]} + H_s \mathbf{E}_{[p,p+f]} \tag{2.8}$$

e si effettua la proiezione obliqua [1] (mediante decomposizione QR)

$$\Gamma \mathbf{X}_p \simeq \hat{\mathbf{Y}}_{[p,p+f]} := E_{||\mathbf{U}_{[p,p+f]}} \left[\mathbf{Y}_{[p,p+f]} | \mathbf{Z}_{[0,p)} \right] . \tag{2.10}$$

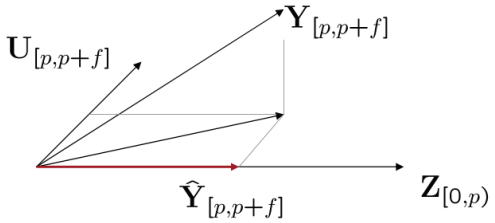


Figura 2.1: Rappresentazione geometrica della proiezione obliqua effettutata in (2.10)

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

$$U_n S_n V_n^T \simeq W_r \hat{\mathbf{Y}}_{[p,p+f]} W_c \tag{2.11}$$

e si ottiene

$$\hat{\Gamma} := W_r^{-1} U_n S_n^{1/2} \qquad \hat{\mathbf{X}}_p := \hat{\Gamma}_r^{-L} \hat{\mathbf{Y}}_{[p,p+f]} \tag{2.12}$$

[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:

$$\hat{E}_{||B_t}[A_t|C_t] = \hat{\Sigma}_{ac|b} \hat{\Sigma}_{cc|b}^{-1} C_t \tag{2.9}$$

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 $\hat{\Sigma}_{ab|c} := \hat{\Sigma}_{ab} - \hat{\Sigma}_{ac} \hat{\Sigma}_{cc}^{-1} \hat{\Sigma}_{cb}$

3 Introduzione agli algoritmi

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. Si può rappresentare ciò 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, come si può vedere in Figura ??. 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 di sistemi di larga scala 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 AIC 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 sia gli algoritmi parametrici tradizionale che ai sottospazi possono cadere in problemi.

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 biseparazione, 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 e molto meno nel 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 [??] che si basa sulla 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 *stablesplinekernel* ed include il vincolo di BIBO stabilità del risultato.

4 LARS

4.1 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.

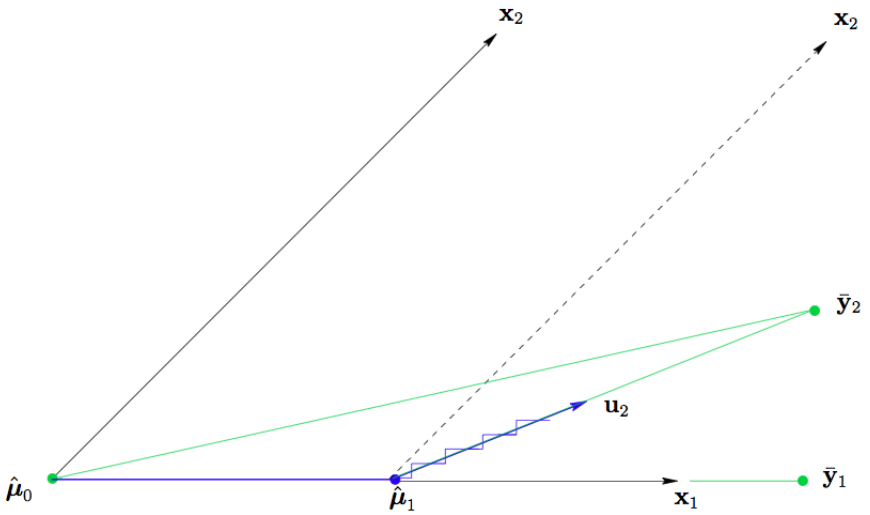


Figura 4.1:

In figura 4.1 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:

$$\mathbf{c}(\hat{\mu}) = X'(\mathbf{y} - \hat{\mu}) = X'(\bar{\mathbf{y}}_\mathbf{2} - \hat{\mu}) \tag{4.1}$$

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 4.1 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:

$$\hat{\mu}_\mathbf{1} = \hat{\mu}_\mathbf{0} + \widehat{\gamma}_1 \mathbf{x}_1. \tag{4.2}$$

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:

$$\hat{\mu}_\mathbf{2} = \hat{\mu}_\mathbf{1} + \widehat{\gamma}_2 \mathbf{u}_2, \tag{4.3}$$

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 4.2.

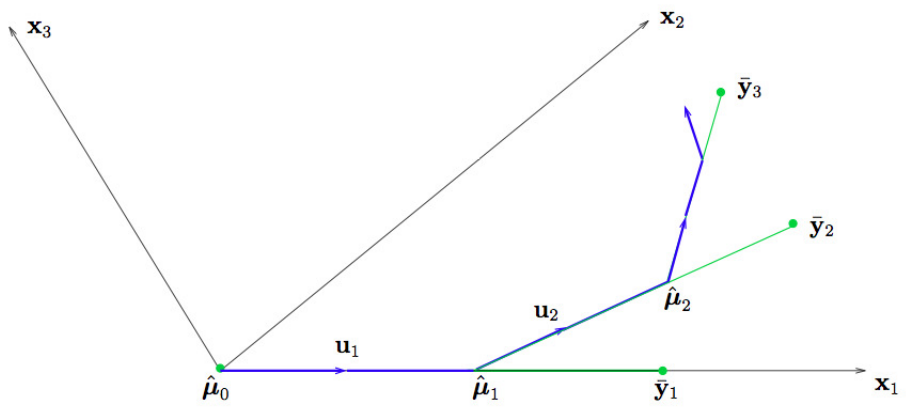


Figura 4.2:

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:

$$\mathbf{X}_\mathcal{A} = (\ldots s_j \mathbf{x}_j \ldots)_{j \in \mathcal{A}} \tag{4.4}$$

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

$$\mathcal{G}_{\mathcal{A}} = X'_{\mathcal{A}} X_{\mathcal{A}} \quad \text{con} \quad \mathcal{A}_{\mathcal{A}} = (1'_{\mathcal{A}} \mathcal{G}_{\mathcal{A}} 1_{\mathcal{A}})^{-\frac{1}{2}} \tag{4.5}$$

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

$$\mathbf{u}_{\mathcal{A}} = X_{\mathcal{A}} w_{\mathcal{A}} \quad \text{con} \quad w_{\mathcal{A}} = \mathcal{A}_{\mathcal{A}} {\mathcal{G}_{\mathcal{A}}}^{-1} 1_{\mathcal{A}} \tag{4.6}$$

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

$$X'_{\mathcal{A}} \mathbf{u}_{\mathcal{A}} = \mathcal{A}_{\mathcal{A}} 1_{\mathcal{A}} \quad \text{e} \quad ||\mathbf{u}_{\mathcal{A}}||^2 = 1. \tag{4.7}$$

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

$$\hat{\mathbf{c}} = X'(\mathbf{y} - \hat{\mu}_{\mathcal{A}}) \tag{4.8}$$

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

$$\widehat{C} = \max_j |\hat{c}_j| \quad \text{e} \quad \mathcal{A} = \{j : |\widehat{c}_j| = \widehat{C}\}. \tag{4.9}$$

Prendendo

$$s_j = sign\{\widehat{c}_j\} \quad \text{con} \quad j \in \mathcal{A} \tag{4.10}$$

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

$$\mathbf{a} \equiv X' \mathbf{u}_{\mathcal{A}}. \tag{4.11}$$

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

$$\widehat{\mu}_{\mathcal{A}_+} = \widehat{\mu}_{\mathcal{A}} + \widehat{\gamma} \mathbf{u}_{\mathcal{A}} \tag{4.12}$$

dove

$$\widehat{\gamma} = \min_{j \in \mathcal{A}^c}{}^{+} \{\frac{\widehat{C} - \widehat{c}_j}{A_{\mathcal{A}} - a_j}, \frac{\widehat{C} + \widehat{c}_j}{A_{\mathcal{A}} + a_j}\} \tag{4.13}$$

con $= \min^+$ che indica che il minimo viene preso solo tra i componenti positivi delle scelte j in 4.13. Usando le formule 4.12 e 4.13 si definisce

$$\mu(\gamma) = \widehat{\mu}_{\mathcal{A}} + \gamma \mathbf{u}_{\mathcal{A}} \tag{4.14}$$

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

$$c_j(\gamma) = \mathbf{x}_j'(\mathbf{y} - \mu(\gamma)) = \widehat{c}_j - \gamma a_j. \tag{4.15}$$

Per $j \in \mathcal{A}$ usando 4.7 4.9

$$|c_j(\gamma)| = \widehat{C} - \gamma A_{\mathcal{A}} \tag{4.16}$$

il che mostra come tutte le correlazioni correnti declinino nella stessa maniera. Per $j \in \mathcal{A}^c$ uguagliando l'equazione 4.15 con la 4.16, 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 4.13 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 4.13,

4.2 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.

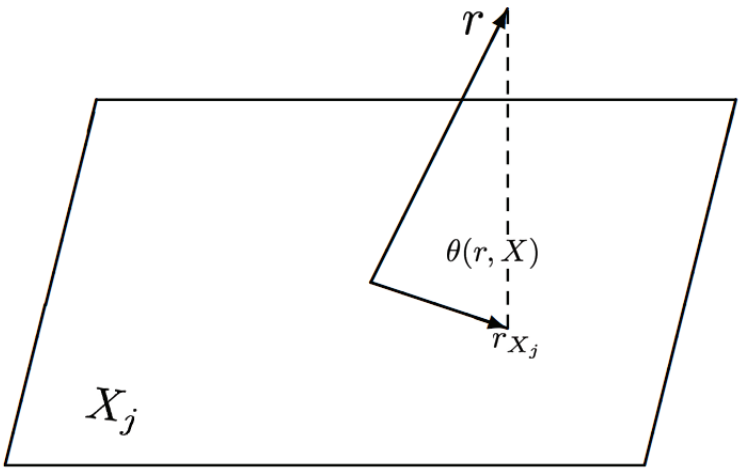


Figura 4.3:

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

$$\cos^2(\theta(r, X)) = ||X_j' r||^2/||r||^2. \tag{4.17}$$

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 tra Y e la proiezione di Y nello spazio creato dagli altri fattori (che chiameremo X_{j2}) che hanno un angolo più piccolo con il corrente residuo,

$$||X_{j1}' r||^2 = ||X_{j2}' r||^2 \tag{4.18}$$

dove r è il corrente residuo. A questo punto la proiezione del corrente residuo nello spazio generato dalle colonne di X_{j1} e di X_{j2} ha lo stesso angolo con i due fattori ed il group LARS procede in questa direzione. Groups LARS continua in questa direzione finché il residuo con $X_j 3$ non è maggiormente 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 gli stessi ingressi, i cui coefficienti sono stati raggruppati in un unico spazio.
Per riassumere, la nostra versione a gruppi dell'algoritmo LARS 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

$$\mathcal{A}_1 = \arg\max_j ||X_j' r[k-1]||^2/p_j. \tag{4.19}$$

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

$$\gamma_{\mathcal{A}_k} = (X_{\mathcal{A}_k}' X_{\mathcal{A}_k}) X_{\mathcal{A}_k}' r^{[k-1]} \tag{4.20}$$

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

- Passo 4:

$$||X_j'(r^{[k-1]} - \alpha_j X\gamma)||^2/p_j = ||X_{j'}'(r^{[k-1]} - \alpha_{j'} X\gamma)||^2/p_{j'} \tag{4.21}$$

  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 $\alpha = 1$.

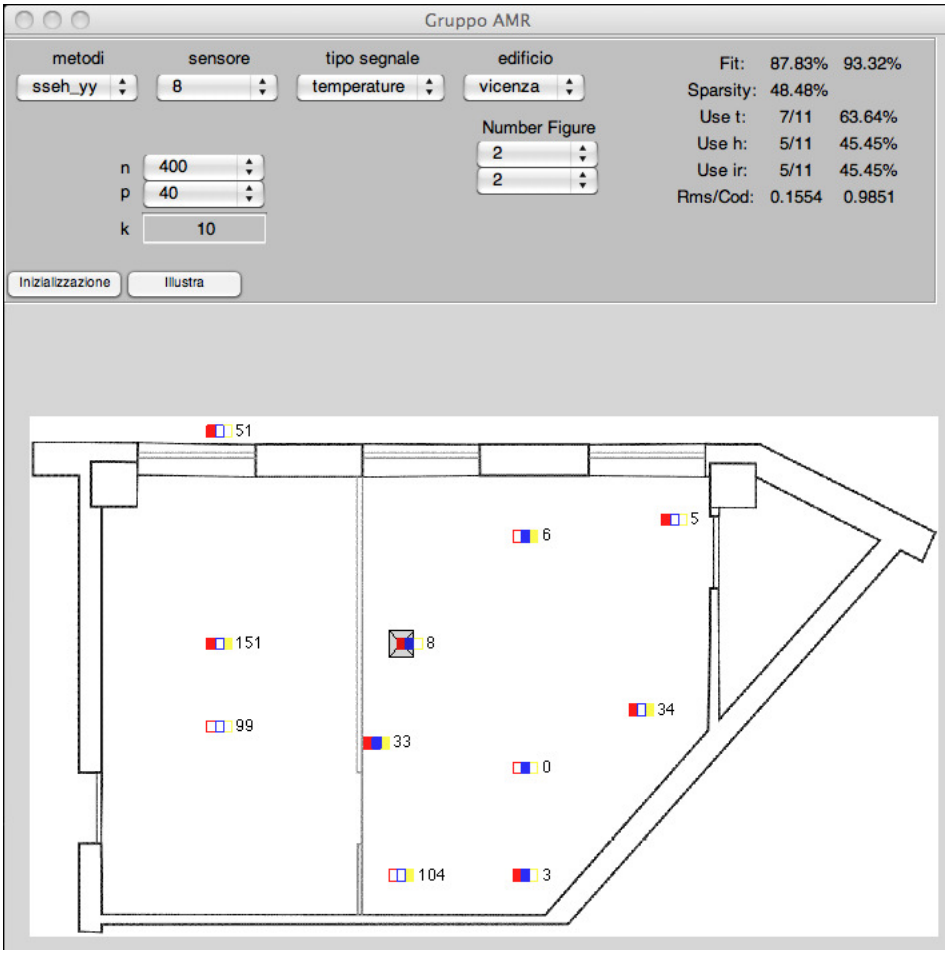
- Passo 6: si aggiorna $\beta^{[k]} = \beta^{[k-1]} + \alpha\gamma$, $r^{[k]} = Y - X\beta^{[k]}$ e $k = k+1$. Si torna al passo 3 finché non si ottiene $\alpha = 1$.

L'equazione 4.21 è una equazione quadratica di α_j che può essere risolta facilmente. Finché j' è il vettore maggiormente correlato, la parte sinistra dell'equazione è minore rispetto alla parte destra nel caso in cui $\alpha_j = 0$. Comunque in accordo con la definizione di γ , il lato destro dell'equazione 4.21 è zero quando $\alpha_j = 1$. Per cui all'ultima iterazione la soluzione di 4.21 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.

4.3 I risultati

5 L'interfaccia grafica

Come visto nei paragrafi precedenti ci si è trovati di fronte ad un enorme quantità di identificazioni legate ad un serie di valori per cui si è valutato interessante lo sviluppo di un interfaccia grafica per il veloce ispeziona mento al variare dei parametri e dei metodi di simulazione.
Per gli algoritmi che davano una forte sparsità era interessante una comprensione spaziale di quali ingressi vengono usati e di quali invece vengono scartati, cosa che è difficoltosa guardando le funzioni di trasferimento e che la visualizzazione grafica aiuta.



L'interfaccia grafica realizzata non è un simulatore, cioè non richiama direttamente le funzioni di identificazione; cosa che visto l'alto tempo di utilizzo di alcuni algoritmi avrebbe fatto diventare la GUI difficilmente utilizzabile. La scelta verso cui si è optato è invece l'utilizzo della struttura Dati in cui si vanno a recuperare tutte le varie simulazioni e si compiono veloci operazioni di calcolo di indici e plottaggio di grafici.
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. Co-

me 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 avendo la presunzione che la struttura Dati venga sempre costruita con le stesse convenzioni e regole).
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 nella situazione di essere già pronti a disegnare un 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 raffigurato 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 di essere stati o meno utilizzati 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.

6 La struttura Dati

Tutti i dati e tutte le simulazioni sono state salvate in un unica struttura di Matalb per avere in un unico luogo tutte le informazioni e le simulazioni. 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 in n elementi , con n il numero dei sensori utilizzati nella raccolta dati. Quindi ogni elemento della struttura ha una corrispondenza univoca con un sensore. La struttura è fatta per essere esplorata velocemente, ogni elemento è formato da 4 campi:

- t: [1x1 struct] : che ha sua volta è una struttura contenenti tutte le informazioni che riguardano la temperatura.
- h: [1x1 struct] : che ha sua volta è una struttura contenente tutte le informazioni che riguardano l'umidità.
- name_of_y , una stringa contenente il nome esteso del sensore.
- id , una stringa contenente il numero del sensore, come id nella rete di sensori.

Le trutture .h e .t contengo a loro volta gli stessi campi, per cui la descrizione che avverrà in seguito sarà fatta per entrambi i rami di esplorazione della struttura Dati. I campi che si trovano annidati dentro Dati(index).t o Dati(index).h sono i seguenti :

- .y_org: [2494x1 double] : vettore contenente il segnale che si vuole identificare. Quindi sarà n-esima temperatura o umidità.
- .y: [2494x1 double] : vettore contenente il segnale che si vuole identificare in media nulla.
- .y_norm: [2494x1 double] : vettore contenente il segnale che si vuole identificare con media nulla e varianza unitaria.
- .Udati_org: [2494x32 double] : Una matrice che contiene tutti i segnali a meno del segnale che si vuole identificare. La parte finale _org indica che non è stata compiuta nessuna operazione.
- .Udati: [2494x32 double] : La stessa matrice Udati_org con normalizzazione in media a 0 ma non in varianza unitaria
- .Udati_norm: [2494x32 double] : La stessa matrice Udati_org con normalizzazione in media a 0 e con varianza unitaria [2].

[2] Si è sempre usata la convenzione che _org indichi il segnale originale, mentre senza nessun parte finale si indica la normalizzazione in media mentre con _norm si indica la normalizzazione sia in media che varianza

- .Colonne_U: [1x32 double] : Un vettore che contiene gli indice dei segnali contenuti in Udati_org per ricostruire poi la mappa dei sensori usati.
- .sseh_yy: [1x2 struct] : Struttura in cui è rinchiusa tutta l'identificazione eseguita con l'algoritmo sseh usando il parametro yy
- .sseh_ny: [1x2 struct] : Struttura in cui è rinchiusa tutta l'identificazione eseguita con l'algoritmo sseh usando il parametro ny
- .n4sid: [1x1 struct] : Struttura in cui è rinchiusa tutta l'identificazione eseguita con l'algoritmo n4sid.
- .ssglar: [1x1 struct] : Struttura in cui è rinchiusa tutta l'identificazione eseguita con l'algoritmo ssglar.
- .pem: [1x1 struct] : Struttura in cui è rinchiusa tutta l'identificazione eseguita con l'algoritmo pem.
- .glar: [1x1 struct] : Struttura in cui è rinchiusa tutta l'identificazione eseguita con l'algoritmo glar.
- .sseh_yy_norm: [1x2 struct] : stessa descrizione della sseh_yy solo con i dati normalizzati con varianza unitaria.
- .sseh_ny_norm: [1x2 struct] : stessa descrizione della sseh_ny solo con i dati normalizzati con varianza unitaria.
- .n4sid_norm: [1x1 struct] : stessa descrizione della n4sid solo con i dati normalizzati con varianza unitaria.
- .ssglar_norm: [1x1 struct] : stessa descrizione della ssglar solo con i dati normalizzati con varianza unitaria.
- .pem_norm: [1x1 struct] : stessa descrizione della pem solo con i dati normalizzati con varianza unitaria.
- .glar_norm: [1x1 struct] : stessa descrizione della glar solo con i dati normalizzati con varianza unitaria.

La struttura Dati(index).h.sseh_ny e Dati(index).h.sseh_yy hanno la stessa forma [3]. Essendoci due variabili che potevano essere modificata, sseh_xx è una struttura 2d la cui dimensione è pari len(vett(n)) x len(vett(p)).

- Mnpf : Struttura idpoly di Matlab contenente il modello identificato

[3] Ovviamente anche per i rami Dati(index).t.sseh_ny e Dati(index).t.sseh_yy

- n_ : Valore del n usato nella simulazione.
- p_ : Valore del p usato nella simulazione.
- dati_val : Struttura iddata di Matlab contenente i dati.

La struttura Dati(index).t.n4sid contiene due strutture:

- struc_n4sid: [1x3 struct] insieme dell'identificazioni fatte
- best_mod: [1x1 struct] miglior identificazione fatta, contiene tutta la struttura.

Ovviamente struc_n4sid e best_mod hanno la stessa struttura:

- dati_id: [601x1x32 iddata] iddata con i campioni per l'identificazione
- dati_val: [800x1x32 iddata] iddata con i campioni per la validazione
- iden_n4sid: [4-D idss] la struttura idss identificata
- n: Campioni usati nell'identificazion
- fit: Fit dell'identificazione

I dati della ssglar sono tutti racchiusi nella struttura mod (Dati(index).h.ssglar(1).mod) la cui forma è la seguente:

- Mnpg: [1x32x186 idpoly] Struttura idpoly di Matlab con la struttura identifica.
- Mnpc: [1x32x186 idpoly] Struttura idpoly di Matlab usando la soglia del 5% per avere una sparsità maggiore.
- serie di costanti
  - beta
  - sigma
  - NT
  - beta2
  - sigma2
  - NT2
  - n
  - pRKHS
- Mgl: [1x32x240 idpoly]Originale

- Mgls: [1x32x120 idpoly]
- deg: 30
- n: 500

Come ultima campo si trova l'identificazione pem contenuta in Dati(index).t.pem.mod con la seguente struttura:

- Mpem_BIC: : [1x32x204 idpoly] Identificazione usando il metodo BIC per la stima dell'ordine del modello.
- Mpem_AICC: [1x32x340 idpoly] Identificazione usando il metodo AICC per la stima dell'ordine del modello.
- n : numero di campioni usati.

7 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 $\varepsilon_\theta(t)$.

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 [**?**].

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 [**?**] 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:

$$AICC = N \cdot \ln \frac{rss}{N} + N \cdot \frac{N+p}{N-p-2}$$
$$BIC = N \cdot \ln \frac{rss}{N} + p \cdot \ln(N)$$

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).

8 Validazione

Una parte molto importante dell'identificazione è la fase di validazione del modello ottenuto. Si cerca di capire la bontà del modello identificato nel descrivere il processo. L'analisi di prestazione deve essere fatta su 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 avendo come ingressi i dati reali 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.

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 [**?**].
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 correlati, 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.

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 plotta un grafico per la auto-correlazione del errore di predizione e un grafico con la cross-correlazione tra l'errore di predizione e ogni ingresso

reale che poi serve per 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

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

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 stato estratto tutta la 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 sono picchi significativamente fuori della 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 sono picche di cross-correlazione che sono fuori dell'intervallo di confidenza questo non è detto per valori di τ negativi. Per

esempio nel caso che $u(t)$ sia un processo bianco questi picchi sono comunque nulle.
Calcolando la correlazione campionaria come

$$\hat{r}_{\varepsilon u}(\tau) = \frac{1}{N} \sum_{t=1-min(0,\tau)}^{N-max(0,\tau)} \varepsilon(t+\tau)u(t)$$

per il test di indipendenza si considera la quantità normalizzata

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

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.

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

$$fit = 100 * [1 - \frac{norm(y_h - y)}{norm(y - mean(y))}]$$

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.

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

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

Questo indice da 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 radice dell'errore di predizione (in inglese root mean square error, RMSE) che è quindi proporzionale alla deviazione standard della stima.

Sparsità La sparsità da 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.