ALMA MATER STUDIORUM

UNIVERSITÀ DEGLI STUDI DI BOLOGNA
SEDE DI CESENA

Seconda Facoltà di Ingegneria con sede a Cesena

Corso di Laurea in Ingegneria Elettronica e delle Telecomunicazioni

IMPLEMENTAZIONE DI ALGORITMI DI SPECTRUM SENSING COOPERATIVO SU PIATTAFORMA USRP2

Tesi in
Laboratorio di Elaborazione Numerica dei Segnali L-A

Presentata da:
MATTEO PARRINI

Relatore:
Prof. Ing.
ANDREA GIORGETTI

Correlatore:
Ing.
ANDREA MARIANI

SESSIONE I
ANNO ACCADEMICO 2011-2012

PAROLE CHIAVE

Radio Cognitiva

Spectrum Sensing

Simulink

Independence Test

Sphericity Test

Indice

Introduzione

Il progresso tecnologico di questi ultimi anni ha introdotto l'utilizzo di innumerevoli servizi nel mondo delle telecomunicazioni. Una ricerca effettuata dalla *Federal Communications Commission* (FCC), ente governativo indipendente degli Stati Uniti incaricato di controllare le frequenze radio, ha messo in evidenza un utilizzo poco efficiente dello spettro. L'accesso a molte bande di frequenze è un problema significativo ma non dovuto alla scarsità fisica di spettro disponibile, bensì a causa di regolamenti e politiche di controllo che ne limitano la capacità di utilizzo da parte di potenziali utenti. Infatti, se si scansiona una porzione di spettro in una qualsiasi area urbana, si nota che molte bande non sono occupate per la maggior parte del tempo, altre sono parzialmente occupate e altre ancora pesantemente sovraffollate di utenti [4]. Questo dimostra appunto, il cattivo utilizzo della risorsa spettro radio.

Ad oggi, sono in atto numerosi studi mirati a migliorare sensibilmente l'efficienza spettrale. Uno di questi si basa sul concetto di **Radio Cognitiva**, cioè di dispositivi intelligenti in grado di adattarsi alle caratteristiche dello spettro in un determinato luogo e periodo temporale. Questi sistemi non hanno solo il compito di comunicare con altri terminali, ma anche di adattarsi alla loro presenza.

In questo consiste il concetto di accesso dinamico allo spettro radio, il cui scopo è di far coesistere, senza interferenza, servizi diversi in una stessa banda.

Per realizzare ciò, è necessario che i dispositivi radio cognitivi riescano a rilevare la presenza di altri utenti (*spectrum sensing*) e di cambiare velocemente i propri parametri di comunicazione per adattarsi alla situazione. Il cambiamento dei parametri non avviene

modificando l'hardware dei sistemi cognitivi ma attraverso elaborazioni di tipo software. Questo è il cosiddetto approccio **Software Defined Radio** (**SDR**).

In questa tesi si è focalizzata l'attenzione sullo *spectrum sensing*, cioè sul rilevamento della presenza di utenti in una determinata banda da parte di sistemi cognitivi. Esistono diverse tecniche di *sensing*, ma in questo elaborato si è affrontato il tema dello ***spectrum sensing* cooperativo**. Esso prevede la scansione di una porzione di banda da parte di più sistemi cognitivi e ciò avviene attraverso l'utilizzo di algoritmi di *sensing*.
Gli algoritmi di *sensing* cooperativo sono molteplici, tuttavia questa tesi è incentrata sull'***Independence Test*** e lo ***Sphericity Test***, che derivano dall'approccio *Generalized Likelihood Ratio Test* (GLRT) [3]. Entrambi sono stati implementati in **Simulink**, il quale prevede un pacchetto dedicato proprio allo sviluppo di *Software Defined Radio* (SDR).

La tesi è così organizzata:

- **Capitolo 1**: si effettua una breve panoramica sui sistemi cognitivi, sullo *spectrum sensig* e sul *Software Defined Radio* (SDR).
- **Capitolo 2**: si effettua una breve digressione sugli algoritmi di *spectrum sensing* cooperativo e si illustrano a livello teorico gli algoritmi di *Independence Test* e di *Sphericity Test*.
- **Capitolo 3**: si descrive l'implementazione in Simulink degli algoritmi di *Independence Test* e di *Sphericity Test* nel caso di pura simulazione, discutendone anche le prestazioni.
- **Capitolo 4**: si analizza il dispositivo USRP2 e si descrive l'hardware e il software utilizzato per l'implementazione degli algoritmi in ambiente reale.

- **Capitolo 5**: si descrive l'implementazione in Simulink degli algoritmi di *Independence Test* e di *Sphericity Test* nel caso reale.

Capitolo 1

Radio Cognitiva

1.1 Definizione

Nella letteratura scientifica non esiste una definizione unica e formale per il concetto di Radio Cognitiva. La prima definizione venne coniata da Joseph Mitola III nel 1998 in [1]:

"A radio that employs model based reasoning to achieve a specified level of competence in radio-related domains".

È descritta come una radio intelligente capace di raggiungere un appropriato livello di conoscenza delle trasmissioni radio presenti nell'ambiente in cui opera. Quindi tale sistema sarà in grado di rilevare i bisogni degli utenti e di conseguenza fornire gli appropriati servizi e risorse radio per poterli soddisfare.
Un'altra significativa definizione è stata fornita dalla *Federal Communication Commission* (FCC) in [2]:

"Cognitive Radio: A radio or system that senses its operational electromagnetic environment and can dynamically and autonomously adjust its radio operating parameters to modify system operation, such as maximize throughput, mitigate interference, facilitate interoperability, access secondary markets".

Viene descritta come una radio o un sistema in grado di analizzare l'ambiente elettromagnetico circostante che di conseguenza può adeguare, dinamicamente e autonomamente, i propri parametri operativi per modificare il comportamento del sistema come, ad esempio, massimizzare la velocità di trasmissione, limitare l'interferenza, facilitare l'interoperabilità e l'accesso a mercati secondari.
Nonostante le numerose definizioni, esistono concetti radio cognitivi da tener conto che esulano da esse. Il primo di questi riguarda la classificazione degli utenti che sfruttano lo spettro radio, i quali possono essere suddivisi in due classi:

- ***Primary Users (PUs)***: utenti che hanno la massima priorità o i diritti legali per l'utilizzo di una certa banda dello spettro radio.
- ***Secondary Users (SUs)***: utenti che hanno priorità inferiore e sfruttano lo spettro radio in modo da non causare interferenze con gli utenti primari PUs.

Quindi, i SUs hanno bisogno di capacità radio cognitive, come ad esempio effettuare un affidabile *spectrum sensing* per la ricerca di porzioni di spettro libere o al momento non utilizzate, dette *spectrum holes*, da PUs.
In questo modo i SUs potranno occupare una *spectrum hole*, senza creare problemi di interferenza con quelli primari.
Come detto in [4], se si effettua una scansione dello spettro radio in una qualsiasi area urbana, è possibile riscontrare le tre seguenti situazioni:

- ***Black Spaces***: bande occupate da un elevatissimo numero di servizi che ne hanno diritto.
- ***Grey Spaces***: bande parzialmente utilizzate.

- ***White Spaces***: bande libere da ogni tipo di interferenze a radio frequenza, eccetto il rumore ambientale.

I SUs, attraverso tecniche di *spectrum sensing* (che verranno analizzate nel paragrafo 1.2), dovranno ricercare *spectrum holes* all'interno di queste tre categorie di bande. Si avrà così un accesso dinamico allo spettro radio passando da una banda all'altra in istanti temporali differenti, il che comporterà ad un conseguente miglioramento del suo utilizzo.

1.2 Spectrum Sensing

Lo *spectrum sensing* è una parte fondamentale del concetto di radio cognitiva, il quale permette di conoscere l'utilizzo di una banda dello spettro radio da parte di un utente primario in una determinata zona geografica.
Volendo darne una definizione, si definisce *spectrum sensing* la capacità di rilevare una possibile banda di frequenze in cui un SU può comunicare senza interferire con altri PUs. Per rilevare la comunicazione di un PU si fa uso di algoritmi di *detection*, i quali si possono basare o sul rilevamento da parte di un singolo SU, come ad esempio l'*Energy Detector*, il *Matched Filter Detection* e il *Feature Detection*, oppure sfruttare il processo di *sensing* effettuato da più dispositivi, detto ***spectrum sensing* cooperativo**.
In questa tesi ci si concentrerà proprio sugli algoritmi di tipo cooperativo, in particolare su quelli che sfruttano gli autovalori della matrice di covarianza (***Eigenvalue Detection***). Essi non prevedono la conoscenza a priori né del segnale né del rumore e godono di prestazioni eccellenti anche con un basso Rapporto Segnale-Rumore (SNR).

Tra gli algoritmi *Eigenvalue Detection*, si è deciso di analizzare e implementare (sia nel simulatore sia in ambiente reale) l'***Independence Test*** e lo ***Sphericity Test*** [3].

1.3 Software Defined Radio

Il concetto di *Software Defined Radio* (SDR) è stato affrontato per la prima volta da Joseph Mitola III. Siccome i sistemi cognitivi devono adattarsi velocemente e cambiare i propri parametri in funzione della presenza di utenti primari, Mitola affermò che tutto ciò non era possibile con un approccio puramente hardware. Infatti il suo intento era quello di riconfigurare i parametri di trasmissione attraverso elaborazioni software, in modo da garantire una maggiore flessibilità e velocità di cambiamento. In questo modo, inoltre, si ha la possibilità di configurare una radio SDR in tempo reale.

1.4 Il problema della Detection

Siccome nei capitoli successivi si utilizzeranno i termini Probabilità di Detection P_D e Probabilità di Falso Allarme P_{FA}, è risultato necessario contestualizzarli in modo appropriato in questo paragrafo.

A causa della non conoscenza a priori del rumore introdotto dal canale in cui si invia il segnale, il ricevitore è costretto a trattare i segnali come processi aleatori. Dato un vettore di valori prodotto da una sorgente che può assumere M possibili stati (denominati "ipotesi"), $H_0,..,H_{M-1}$, a monte di un meccanismo di transizione probabilistica, valori uguali di questo vettore possono derivare da differenti ipotesi. Il problema della Detection consiste nell'associare, ad ogni valore del vettore dei campioni ricevuti, l'ipotesi più appropriata. Quindi, il sistema ricevente sceglierà l'ipotesi in base al confronto tra una metrica T, definita in un qualsiasi algoritmo, e una soglia ξ.

In questa tesi si considera il caso con sole due ipotesi (decisore binario, rappresentato in figura 1.1).

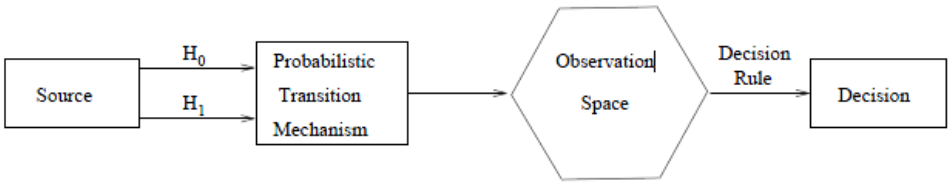

Figura 1.1: Decisore Binario [16]

Una possibile rappresentazione delle due ipotesi è la seguente:

$$y(t) = \begin{cases} n(t) & per\ H_0 \\ s(t) + n(t) & per\ H_1 \end{cases}$$

dove *s(t)* rappresenta il segnale da rilevare e *n(t)* il rumore termico equivalente introdotto dal canale trasmissivo. Quest'ultimo viene descritto come un processo aleatorio Gaussiano a valor medio nullo, additivo e con densità spettrale costante in banda (AWGN).
Indicando con H_0 e H_1 l'assenza e la presenza di segnale rispettivamente, le combinazioni che si possono verificare tra evento accaduto e decisione del detector sono riportate nella seguente tabella:

Segnale inviato	Decisione effettuata	Evento	Probabilità
H_0	H_0	Nessun segnale inviato, decisione corretta	Correct Rejection P_R
H_0	H_1	Nessun segnale inviato, decisone errata	False Alarm P_{FA}
H_1	H_0	Segnale inviato ma non rilevato	Missed Detection P_{MD}
H_1	H_1	Segnale inviato e rilevato	Detection P_D

Capitolo 2

Algoritmi di Spectrum Sensing Cooperativo

Al fine di rilevare porzioni di spettro inutilizzate e monitorare l'attività degli utenti primari (PUs), i dispositivi radio cognitivi devono implementare algoritmi di *spectrum sensing* molto accurati. Molte tecniche di *sensing* sono state descritte in letteratura, come ad esempio l'*Energy Detector* e il *Feature-Based Detector*. Queste tecniche possono essere usate sia in caso di *sensing* attraverso un singolo nodo sia attraverso nodi multipli. Però esse presentano alcune limitazioni, come ad esempio la necessità di effettuare una corretta stima della potenza di rumore. Per ovviare a questo problema, vengono sempre più utilizzati algoritmi detti *Eigenvalue-Based* che sfruttano le proprietà della matrice di covarianza e dei suoi autovalori. Infatti, per questa categoria di algoritmi, non è necessaria né la conoscenza della stima di rumore né alcuna informazione a priori sui segnali degli utenti primari. Nel caso non ci fossero PUs, i differenti nodi cognitivi capterebbero solo rumore incorrelato e quindi la matrice di covarianza sarà diagonale. Altrimenti, quando sono presenti segnali di PUs, la matrice di covarianza non è più diagonale in quanto è presente un certo grado di correlazione tra i segnali ricevuti.

Nella teoria dello *spectrum sensing* cooperativo, si assume che i nodi di rilevazione abbiano tutti la stessa potenza di rumore. Nell'ambiente reale, invece, tutto questo non è assolutamente vero. Infatti, differenti dispositivi cognitivi sono soggetti a differenti potenze di rumore a causa del diverso processo di fabbricazione,

della temperatura e dell'ambiente in cui sono situati. Dispositivi che presentano differenti potenze di rumore si dicono *non calibrati*.
In questa tesi si focalizzerà l'attenzione su due algoritmi cooperativi in particolare: l'***Independence Test*** e lo ***Sphericity Test***, entrambi descritti in [3].
Questi due algoritmi non sono altro che due casi particolari dell'applicazione del *Generalized Likelihood Ratio Test* (GLRT). Infatti, nel caso in cui gli utenti secondari presentino la stessa potenza di rumore, l'approccio GLRT porta al cosiddetto *Sphericity Test*, mentre nel caso sperimentino differenti potenze di rumore ci si riferirà all'*Independence Test*.
Prima di entrare nel dettaglio con questi algoritmi, è interessante descrivere il modello del sistema considerato e l'aspetto più generale del GLRT.

2.1 Modello del sistema e GLRT

2.1.1 Modello del sistema

Considerati n_{R} utenti secondari cooperativi, si assume che, dopo un opportuno filtraggio passa-banda e una conversione in banda base, l'uscita delle antenne riceventi, all'istante *i*-esimo, si presenti nella seguente forma:

$$\mathbf{y_i} = \mathbf{H}\,\mathbf{x_i} + \mathbf{n_i} \tag{2.1}$$

dove $\mathbf{x_i}$ è il vettore dei campioni del segnale trasmesso dagli n_{T} utenti primari, **H** è una matrice complessa $M_{n_R x n_T}(C)$ che descrive il guadagno del canale radio interposto tra gli n_{T} segnali trasmessi e gli n_{R} utenti secondari, mentre $\boldsymbol{n_i}$ è un vettore complesso C^{n_R} che rappresenta il rumore termico additivo.

Si è assunto che gli elementi del vettore $\boldsymbol{n_i}$ siano variabili aleatorie indipendenti con distribuzione Gaussiana a valor medio nullo e che $\boldsymbol{x_i} \sim CN(\mathbf{0},\mathbf{R_x})$. Una volta collezionate n_S istanze di $\mathbf{y_i}$, si forma la matrice

$$\mathbf{Y} = (\mathbf{y_1}| \dots |\mathbf{y_{n_S}}) \tag{2.2}$$

dove i vettori $\mathbf{y_i}$ rappresentano i vettori-colonna di tale matrice.
Osservando la matrice **Y** è possibile decidere se sono presenti o meno degli utenti primari. Si sono indicate queste due ipotesi con H_1 e H_0 rispettivamente.
Sotto l'ipotesi H_0, si avrà $\boldsymbol{y_i} = \boldsymbol{n_i}$ e la matrice di covarianza di **Y**, $\Sigma_{\mathbf{0}}$, sarà diagonale, mentre sotto l'ipotesi H_1 le colonne di **Y** saranno correlate e si indicherà la rispettiva matrice di covarianza con il simbolo $\boldsymbol{\Sigma_1}$.

2.1.2 Generalized Likelihood Ratio Test

Sotto le ipotesi H_j, con j=0,1, la funzione di verosimiglianza di **Y** è data da

$$L(\boldsymbol{Y}|\boldsymbol{\Sigma_j}) = \frac{1}{\pi^{n_R n_S}\left|\boldsymbol{\Sigma_j}\right|^{n_S}} e^{\left(-n_S \, tr\left\{\boldsymbol{\Sigma_j^{-1}} S\right\}\right)} \tag{2.3}$$

dove la matrice **S** è la matrice di covarianza campionaria normalizzata (SCM). Essa è così definita

$$\mathbf{S} = \frac{1}{n_S} \mathbf{Y}\,\mathbf{Y^H} \tag{2.4}$$

Si parla di *matrice di covarianza campionaria* e non semplicemente di matrice di covarianza in quanto il valor medio temporale dei

campioni, nel caso reale, si calcola in una finestra di osservazione finita (la definizione rigorosa comprende, invece, l'operazione di limite che estende la finestra di osservazione all'infinito).
Quindi, il GLR per rilevare l'ipotesi H_0 è

$$\frac{L(\boldsymbol{Y}|\widehat{\boldsymbol{\Sigma}_0})}{L(\boldsymbol{Y}|\widehat{\boldsymbol{\Sigma}_1})}. \tag{2.5}$$

Nell'ipotesi in cui $\boldsymbol{\Sigma_0}$ sia una matrice diagonale, a partire da (2.5), utilizzando (2.3), è possibile formulare il GLRT come [3]:

$$T = \frac{\left|\hat{\Sigma}_1\right|}{\left|\hat{\Sigma}_0\right|} \underset{H_1}{\overset{H_0}{\lessgtr}} \xi \tag{2.6}$$

dove $0 \leq \xi \leq 1$ è definita come soglia per tale decisore. Quindi se T risulterà maggiore di ξ, si ricadrà nell'ipotesi H_1, altrimenti siamo in presenza dell'ipotesi H_0.

2.1.3 Approssimazione mediante distribuzione Beta

Allo scopo di ricavare una distribuzione approssimata per il GLR, in [3] è stato proposto l'approccio attraverso il Metodo dei Momenti. È possibile quindi approssimare la distribuzione di T utilizzando un semplice modello di distribuzione (Gaussiana, Beta, Chi-quadro, ecc) e settare i suoi *k* parametri al fine di approssimare i primi *k* momenti di T.
Considerando che il GLR può assumere valori tra 0 e 1, in [3] si è scelto di utilizzare la *distribuzione Beta* per approssimare la sua funzione densità di probabilità f_T.

Il momento di ordine p di una variabile aleatoria distribuita secondo Beta, avente parametri a e b, è dato dalla seguente espressione

$$m_p^{(\beta)} = \frac{\Gamma(a+b)\Gamma(a+p)}{\Gamma(a)\Gamma(a+b+p)}\,. \tag{2.7}$$

Indicando con m_1 ed m_2 il momento di primo e di secondo ordine di T rispettivamente, è possibile ottenere le seguenti espressioni dei parametri a e b:

$$a = \frac{m_1(m_2 - m_1)}{m_1^2 - m_2} \tag{2.8}$$

$$b = \frac{(1 - m_1)(m_2 - m_1)}{m_1^2 - m_2}\,. \tag{2.9}$$

Quindi, la funzione densità di probabilità approssimata di T sarà data

$$f_T(t) \cong \begin{cases} \frac{1}{B(a,b)} t^{a-1}(1-t)^{b-1}, & 0 \le t \le 1 \\ 0, & altrimenti \end{cases} \tag{2.10}$$

dove $\mathrm{B(a,b)} = \int_0^1 \mathrm{x}^{\mathrm{a}-1}(1-\mathrm{x})^{\mathrm{b}-1}\,\mathrm{dx}$ è la funzione Beta.
Definendo la probabilità di falso allarme di T come $P_{FA} \triangleq Pr\{T < \xi | H_0\}$, si ottiene

$$P_{FA} \cong \int_0^{\xi} \frac{1}{B(a,b)} t^{a-1}(1-t)^{b-1} dt = \tilde{B}(a,b,\xi) \tag{2.11}$$

dove $\tilde{B}(\mathrm{a,b},\xi) = \frac{1}{\mathrm{B(a.b)}} \int_0^{\xi} \mathrm{x}^{\mathrm{a}-1}(1-\mathrm{x})^{\mathrm{b}-1}\,\mathrm{dx}$ è la funzione Beta regolarizzata.
Utilizzando il criterio di Neyman-Pearson, la soglia viene impostata a partire da una probabilità di falso allarme desiderata P_{FA}^{DES}. In

questo caso è possibile ottenere la soglia invertendo l'equazione (2.11):

$$\xi = \tilde{B}^{-1}(a, b, P_{FA}^{DES}) \,. \tag{2.12}$$

2.2 Independence Test

Come anticipato nell'introduzione di questo capitolo, quando i nodi cognitivi sperimentano differenti livelli di potenza di rumore, ci si riferisce all'*Independence Test*. Questa condizione è pressoché sempre verificata in un ambiente reale.

Come descritto in [3], l'*Independence Test* presenta la seguente metrica per il confronto con la soglia:

$$T^{(ind)} = \frac{|\boldsymbol{S}|}{\prod_{k=1}^{n_R} s_{k,k}} \underset{H_1}{\overset{H_0}{\gtrless}} \xi \tag{2.13}$$

dove *s(i,j)* è l'elemento *(i,j)* della matrice di covarianza campionaria normalizzata **S**.

In questa tesi ci si concentra solamente nel caso n_R=2, in quanto sia le simulazioni sia le prove in laboratorio sono state effettuate con due dispositivi cognitivi riceventi (che fungono da SUs).

Come descritto in [3], nel caso n_R=2, la metrica per il calcolo della soglia si semplifica in $T^{(ind)}$= T_2 dove $T_k \triangleq \frac{|S_k|}{|S_{k-1}| \cdot s_{k,k}}$ (con $\mathbf{S_k}$ si indica il minore di ordine *k*-esimo della matrice **S**). Quindi, l'espressione della probabilità di falso allarme P_{FA} si riduce a

$$P_{FA} = \xi^{n_S - 1} \tag{2.14}$$

e la soglia sarà data dalla seguente espressione:

$$\xi = P_{FA}^{\frac{1}{n_S - 1}}. \tag{2.15}$$

Per i casi con n_R>2 e per la completezza dei passaggi matematici si faccia riferimento a [3].

2.3 Sphericity Test

Se i SUs sperimentano la stessa potenza di rumore σ^2, il GLRT viene chiamato *Sphericity Test*. A partire dalla formula (2.6) si ottiene

$$T^{(sph)} = \frac{|\boldsymbol{S}|}{\left(\frac{tr\{\boldsymbol{S}\}}{n_R}\right)^{n_R}} \underset{H_1}{\overset{H_0}{\gtrless}} \xi \,. \tag{2.16}$$

Utilizzando i momenti di $T^{(sph)}$ forniti dalla seguente formula

$$m_p^{(sph)} = \frac{n_R^{n_R p}\Gamma(n_S n_R)}{\Gamma(n_S n_R + n_R p)} \prod_{i=1}^{n_R} \frac{\Gamma(n_S - i + 1 + p)}{\Gamma(n_S - i + 1)} \tag{2.17}$$

si adotta il Metodo dei Momenti applicato alla distribuzione Beta. I parametri $a^{(sph)}$ e $b^{(sph)}$ vengono ricavati inserendo $m_1^{(sph)}$ e $m_2^{(sph)}$ in (2.8) e (2.9). Si sostituiscono $a^{(sph)}$ e $b^{(sph)}$ in (2.11) e (2.12) e si ricava la corrispondente soglia.
Infatti, la probabilità di falso allarme P_{FA} è data da

$$P_{FA} = \tilde{B}\left(n_S - 1, \frac{3}{2}, \xi\right) \tag{2.18}$$

dalla quale si ricava la soglia

$$\xi = \tilde{B}^{-1}(a, b, P_{FA}^{DES}) \,. \tag{2.19}$$

Capitolo 3

Implementazione in Simulink

Nel capitolo precedente si è fatta una panoramica degli algoritmi di *Independence Test* e di *Sphericity Test* e si sono illustrate le loro rispettive peculiarità.

In questo capitolo viene esposta in dettaglio la loro implementazione nel simulatore, descrivendo ogni blocco logico utilizzato ed evidenziando le potenzialità dell'approccio SDR basato su Simulink.

Innanzitutto, mi sono creato uno *script* in MATLAB *Variabili.m*, il quale contiene tutte le variabili globali necessarie per lo sviluppo di entrambi gli algoritmi. Questo script deve essere avviato prima di ogni simulazione premendo il pulsante *Save and Run*. In questo modo carico le mie variabili nel *Workspace* di MATLAB, le quali saranno disponibili per ogni modello *.mdl* in Simulink.

```
%----------Variabili globali------------
%---------------------------------------
%----Cooperative Detection Variables----

nS=500;   % Numero di campioni osservati in una realizzazione
nP=50000; % Numero di realizzazioni del segnale ricevuto
pfad=0.1; % Probabilità di falso allarme desiderata
tS=0.1;   % Passo di campionamento per i segnali simulati
SNRdB=-10;          % Signal to Noise Ratio (dB
SNR=10^(SNRdB*0.1); % Signal to Noise Ratio lineare
dec=512;            % Decimazione USRP2 RX
samp=dec/100e6;     % Sample Time per USRP2

%-------------------------------------------------
% DON'T CONNECT TX AND RX!!!
% Need an attenuator about 40-50 dB.
%
% WBX Transmit Power 30 to 100 mW (14.77 - 20 dbm)
% WBX Receiver:
% Primary and second stage Max power limit -10dbm
% Suggested max -30dbm direct cable connection.
% (tipical use -60dbm on-air)
%-------------------------------------------------
```

La spiegazione di ogni parametro sarà curata nei paragrafi successivi.

Negli algoritmi che seguiranno si è voluta simulare la situazione di un unico segnale (trasmesso da un utente primario, cioè n_T=1) che viene ricevuto da due differente dispositivi cognitivi secondari (cioè n_R=2), i quali posseggono differenti potenza di rumore. Questi ultimi, attraverso tali algoritmi, calcoleranno la probabilità di detection P_D e la probabilità di falso allarme P_{FA}.

3.1 Implementazione Independence Test

La spiegazione a livello teorico dell'algoritmo di *Independence Test* è stata affrontata nel paragrafo 2.2. In questo paragrafo si vuole descrivere l'implementazione di tale algoritmo in un modello *Simulink* e analizzare i risultati ottenuti mediante opportuni grafici.

3.1.1 Independence Test con Rayleigh Fading

Lo schema del modello Simulink *IndependenceTestRayChan.mdl* è riportato in figura 3.1.

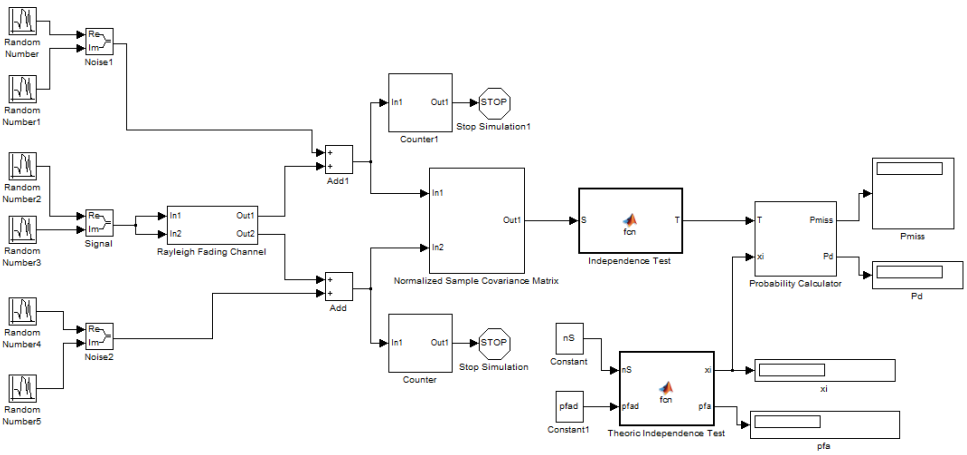

Figura 3.1: Independence Test con Rayleigh Fading

Come detto nel paragrafo 2.2, la soglia ξ viene impostata in base alla probabilità di falso allarme desiderata P_{FA}^{DES} e al numero di campioni osservati n_S in una realizzazione del segnale. A titolo esemplificativo, se si fissa la probabilità di falso allarme desiderata P_{FA}^{DES}=0.1 ed n_S=500, questi due valori dovranno essere inseriti in (15) per il calcolo della soglia ξ. Come si evince da [3], la soglia ξ viene settata indipendentemente dalla stima della potenza di rumore. Una volta calcolata la soglia, questa viene confrontata con $T^{(ind)}$ come mostrato dalla (2.13). Per completezza, risulta noto anche il SNR in quanto è stato fissato nello script *Variabili.m*.

Ora si analizza blocco per blocco lo schema di figura 3.1. Sulla sinistra si possono osservare dei blocchi *Random Number*, i quali generano campioni aleatori di segnale con distribuzione Gaussiana.

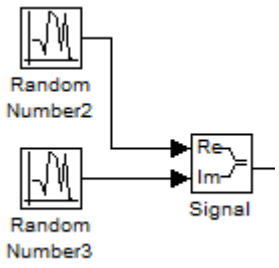

Figura 3.2: Blocchi di segnale

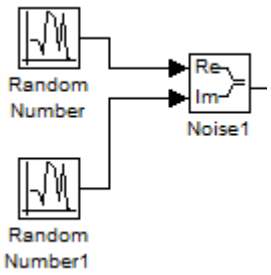

Figura 3.3: Blocchi di rumore

In figura 3.2 e figura 3.3, si è voluto mostrare in dettaglio la generazione dei campioni di segnale e di rumore. In entrambi i casi i campioni sono di tipo complesso e si nota che un blocco *Random Number* è dedicato alla parte reale e l'altro alla parte immaginaria.

I blocchi *Random Number*, come mostrato in figura 3.4, presentano i campi *Mean*, *Variance*, *Seed* e *Sample Time*.

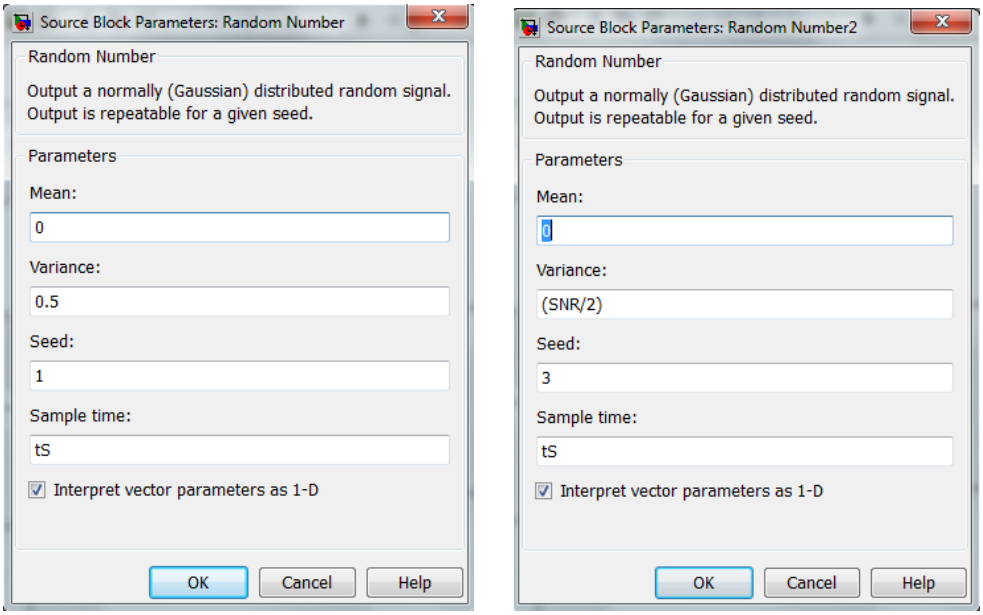

Figura 3.4: Parametri dei blocchi Random Number per rumore e segnale

Il campo *Mean* indica il valor medio del processo, *Variance* indica la varianza cioè la potenza del segnale, *Seed* è un fattore di correlazione tra diversi blocchi *Random Number* (se due blocchi hanno lo stesso *Seed*, generano campioni tra loro correlati), mentre *Sample Time* è il passo di campionamento tra un campione e il successivo.

È utile soffermarsi sulle differenze tra i parametri presenti in figura 3.4. Sulla sinistra di figura 3.4 sono mostrati i parametri del blocco *Random Number* per il rumore. Come detto nel paragrafo 2.1.1, si è assunto il rumore come un processo aleatorio gaussiano $\boldsymbol{x_i} \sim CN(\mathbf{0}, \mathbf{R_x})$. Per semplicità, si è posta la potenza di entrambi i rumori unitaria (cioè le varianze) $\sigma_1^2 = \sigma_2^2 = 1$. Siccome la parte reale e la parte immaginaria dei campioni vengono generate con due blocchi diversi, i valori di *Variance* vengono dimezzati in modo che la loro somma dia il valore richiesto.

Sulla destra di figura 3.4, invece, si sono visualizzati i parametri riguardanti i campioni di segnale. Anche per il dimezzamento del

valore di SNR vale il discorso fatto in precedenza per le potenze di rumore.

È opportuno effettuare una piccola riflessione anche sul campo *Seed*. Affinché i numeri generati da ogni blocco siano incorrelati, è necessario che il *Seed* sia diverso in ognuno di essi, altrimenti se si utilizzasse sempre lo stesso *Seed* verrebbero generati sempre gli stessi numeri. Con la presenza di due blocchi di figura 3.3, si è voluto simulare il rumore prodotto da ogni dispositivo cognitivo.

Proseguendo verso destra nel modello di figura 3.1, si incontra il blocco *Rayleigh Fading Channel*. L'interno di questo sottosistema è mostrato in figura 3.5.

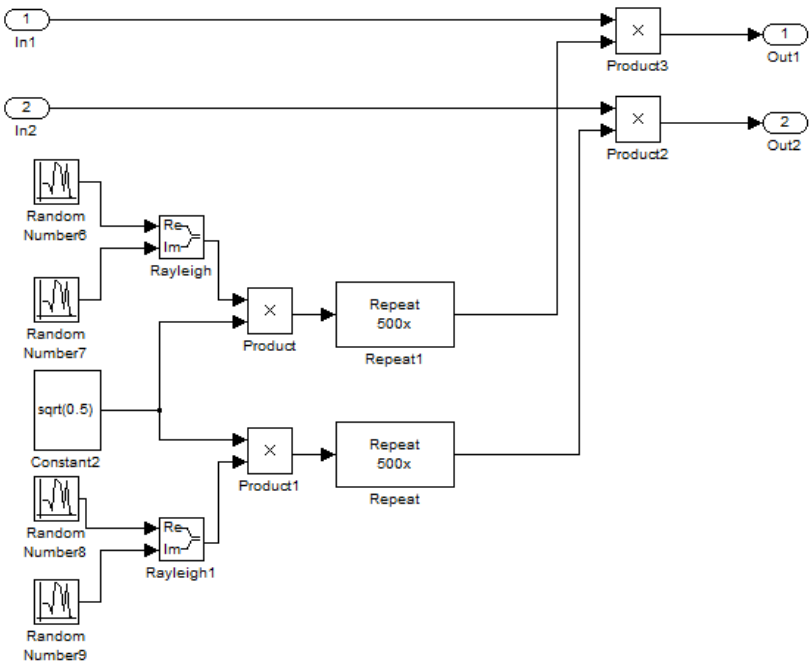

Figura 3.5: Interno del blocco Rayleigh Fading Channel

Lo schema sopra riportato simula il *fading* del canale di trasmissione. Come scritto in [3], si è supposto che sia presente un *fading* alla Rayleigh. Si è assunto, quindi, che gli elementi della matrice di canale **H**, descritta nel paragrafo 2.1.1 e utilizzata nella formula (2.1), siano indipendenti, identicamente distribuiti,

Gaussiani complessi e a valor medio nullo. Inoltre, la matrice **H** deve rimanere costante durante gli n_S campioni di una realizzazione.
Il singolo coefficiente del fading alla Rayleigh si presenta nella seguente forma in codice MATLAB

$$sqrt(0.5) * (randn + 1i * randn)\,. \qquad (3.1)$$

Con il linguaggio a blocchi di Simulink, i comandi *randn* sono stati sostituiti con blocchi *Random Number*, tutti con valor medio nullo, varianza unitaria e *Seed* sempre diverso. Come si vede in figura 3.5, si è assunto che il segnale di un ipotetico PU raggiunga i due dispositivi cognitivi riceventi attraverso lo stesso canale, ma seguendo due cammini multipli diversi (il che comporta una diversa attenuazione per tratta). Questa è la spiegazione per cui si sono realizzati due differenti coefficienti di Rayleigh, uno per ogni dispositivo. Una volta generati i coefficienti, per mantenerli costanti per una intera realizzazione (quindi per n_S campioni) si è utilizzato il blocco *Repeat*. Come mostrato in figura 3.6, tale blocco ripete il coefficiente complesso di Rayleigh per n_S volte, dopodiché preleva un altro coefficiente dai *Random Number* e lo ripete anch'esso per n_S volte e così via. In questo modo si sono riusciti a mantenere costanti i due coefficienti del *fading* per un'intera realizzazione del segnale.

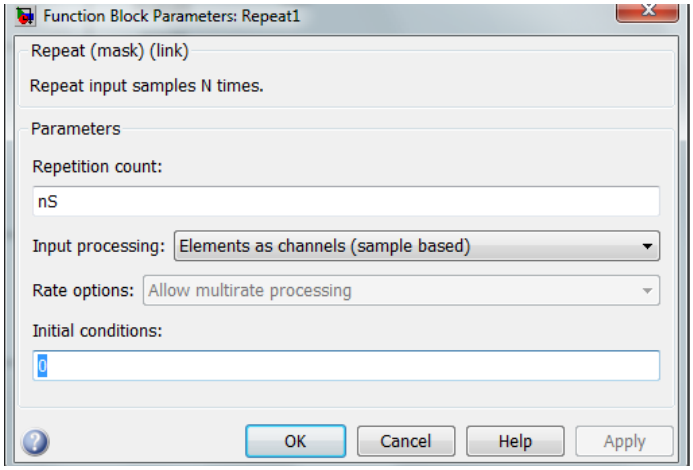

Figura 3.6: Parametri blocco Repeat

Per la (2.1), i coefficienti di Rayleigh vanno moltiplicati, attraverso il blocco *Product*, con gli n_S campioni del segnale. In questo modo si è simulato il *fading* causato dal canale trasmissivo.

Finito di analizzare il blocco *Fading Rayleigh Channel*, si inizia ad entrare nel cuore dell'algoritmo. In figura 3.7 viene mostrato il contenuto del blocco *Normalized Sample Covariance Matrix*, cioè come viene realizzata la matrice di covarianza campionaria normalizzata **S**.

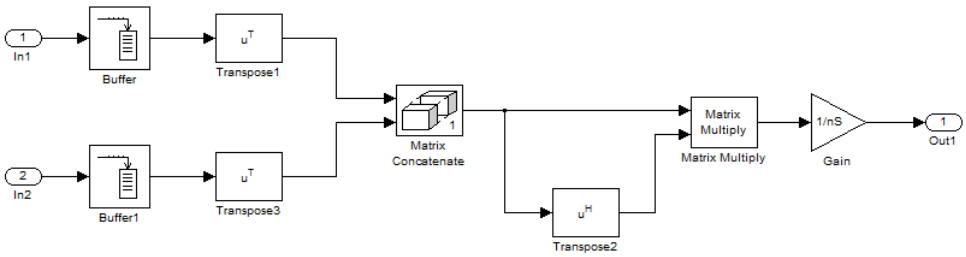

Figura 3.7: Interno del blocco Normalized Sample Covariance Matrix

All'ingresso di questo blocco giungono due flussi di campioni (comprendenti segnale e rispettivi rumori), i quali vengono immagazzinati dai blocchi *Buffer*. Il *Buffer* raccoglie n_S campioni in un vettore-colonna di egual lunghezza. Tramite il blocco *Transpose* si trasforma il vettore-colonna in vettore-riga e i due vettori-riga

vengono poi inviati al blocco *Matrix Concatenate* che ha il compito di creare la matrice **Y** descritta nel paragrafo 2.1.2. Il blocco *Transpose2* effettua la trasposta Hermitiana della matrice **Y** e tramite il blocco *Matrix Multiply* si effettua l'operazione presente nella formula (2.4). Per concludere, il blocco *Gain* effettua la normalizzazione a n_S ed in uscita si avrà la matrice di covarianza campionaria normalizzata **S**.

Il blocco *Independence Test*, mostrato in figura 3.8, racchiude al suo interno la metrica $T^{(ind)}$ nel caso di n_R=2.

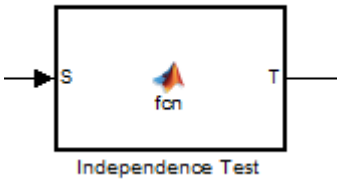

Figura 3.8: Blocco Independence Test

Il blocco sopra riportato è stato programmato in linguaggio MATLAB e riporta il seguente contenuto:

```
function T = fcn(S)
%# T(ind)

T = real(det(S)/(prod(diag(S))));
```

Il codice scritto equivale alla (2.13) nel caso n_R=2

$$T^{(ind)} = \frac{|S|}{s_{1,1} \cdot s_{2,2}} \tag{3.2}$$

dove il denominatore è il prodotto della diagonale principale. Nel codice MATLAB si è dovuto aggiungere il comando *real* perché, a causa delle approssimazioni effettuate dal programma nel calcolo, insorgeva una parte immaginaria molto piccola (dell'ordine di 10^{-19}) inaspettata.

Con il blocco *Theoric Independence Test*, mostrato in figura 3.9, si è calcolata la soglia ξ a partire dai parametri n_S e P_{FA}^{DES}.

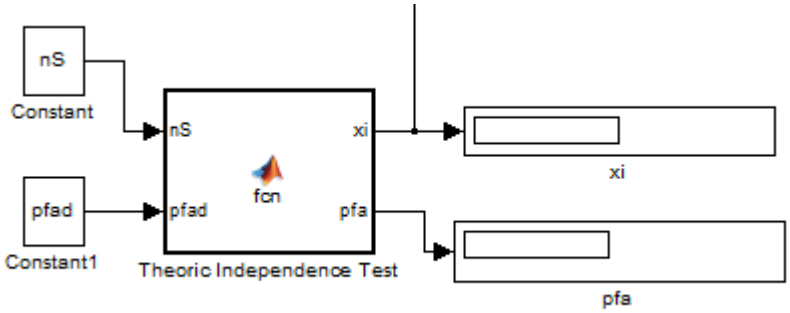

Figura 3.9: Blocco Theoric Independence Test

All'interno di tale blocco si è tradotta in codice MATLAB la formula (2.15) per il calcolo della soglia ξ:

```
function [xi,pfa] = fcn(nS, pfad)

%#Espressioni empiriche della soglia e della Pfa
xi=pfad^(1/(nS-1));
pfa=(xi)^(nS-1);
```

Infine, si giunge all'ultimo blocco di questo algoritmo mostrato in figura 3.10.

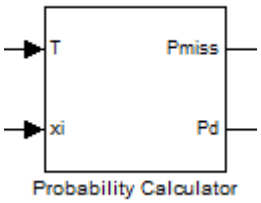

Figura 3.10: Probability Calculator

Nel blocco *Probability Calculator* avviene il confronto fra la metrica $T^{(ind)}$, derivata dalla matrice di covarianza campionaria, e la soglia ξ, calcolata nel blocco *Theoric Independence Test*. In figura 3.11 viene mostrato il suo contenuto.

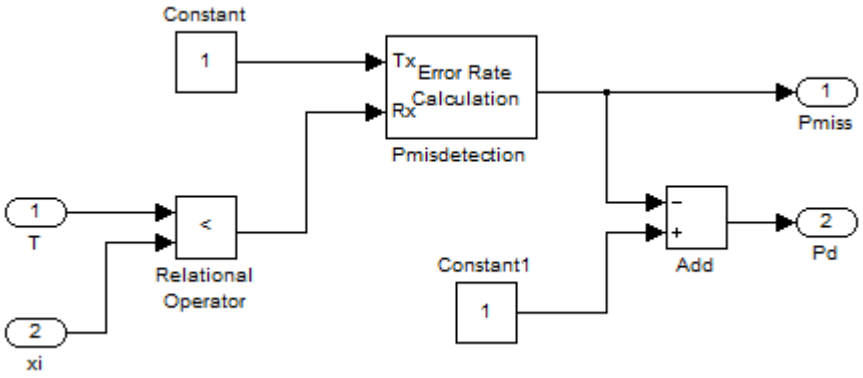

Figura 3.11: Interno del blocco Probability Calculator

Siccome siamo nell'ipotesi H_1, cioè di presenza di segnale, l'espressione (2.13) deve essere considerata con il simbolo "<". Infatti, ogni n_S campioni e per n_R volte (simulazione alla *Monte-Carlo*), il blocco *Relation Operator* confronta il valore di $T^{(ind)}$ con quello della soglia ξ. Se $T < \xi$, tale blocco restituirà un 1 logico, altrimenti uno 0 logico. Scorrendo verso destra nella figura 3.11, si nota la presenza del blocco *Pmisdection*, il quale effettua il calcolo della *Bit Error Rate* (BER) e fornisce in uscita tre valori: la probabilità di errore P_{miss}, il numero di campioni che hanno superato la soglia e il numero di campioni totali. Per ottenere la probabilità di detection P_D, è necessario ricordare il legame tra P_{miss} e P_D:

$$P_{miss} + P_D = 1\,. \tag{3.3}$$

Si inverte la (3.3) in modo da ricavarsi la probabilità di detection P_D, quindi si ottiene:

$$P_D = 1 - P_{miss}\,. \tag{3.4}$$

La (3.4) è stata implementata utilizzando il blocco *Add* e i valori di P_{miss} e di P_D vengo visualizzati per mezzo di due *Display*.

L'ultimo blocco rimasto da descrivere è il *Counter*, il quale è stato utilizzato per il conteggio dei campioni ricevuti e per fermare la simulazione. La figura 3.12 mostra cosa è contenuto al suo interno.

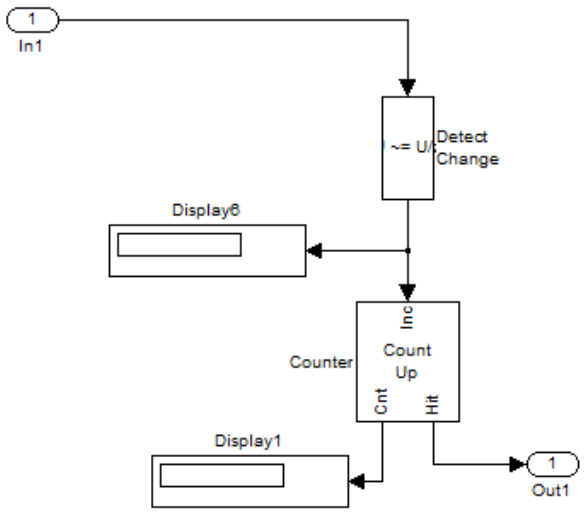

Figura 3.12: Interno del blocco Counter

Si è scelto di fermare la simulazione al valore $n_S \cdot n_P$, dove n_P rappresenta il numero di cicli di Monte Carlo.

Nei paragrafi successivi verranno illustrati gli altri algoritmi simulati. A causa della notevole somiglianza tra gli algoritmi e per evitare futili ripetizioni, nelle descrizioni successive mi limiterò a illustrare solamente le differenze.

3.1.2 Independence Test con canale AWGN

A differenza del paragrafo 3.1.1, tale algoritmo considera il caso di canale trasmissivo di tipo AWGN, non più quindi con fading alla Rayleigh. Un canale di questo tipo presenta un rumore termico additivo descritto come una variabile aleatoria con distribuzione gaussiana, a valor medio nullo e varianza che corrisponde alla

potenza del disturbo. Il modello Simulink *IndependenceTestAWGN.mdl* è mostrato in figura 3.13.

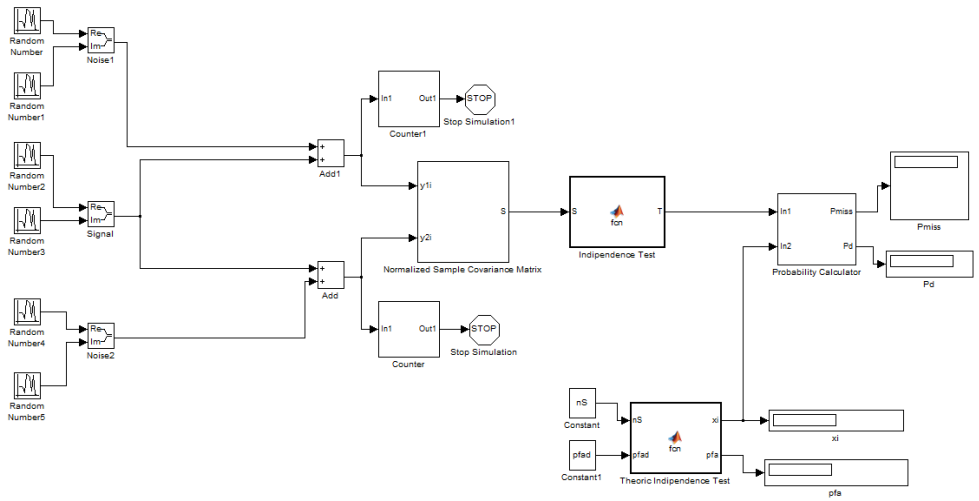

Figura 3.13: Independence test con canale AWGN

Lo schema è identico a quello del paragrafo 3.1.1, l'unica differenza è data dall'assenza del blocco *Rayleigh Fading Channel*. In questo caso, i campioni di rumore comprendono sia il contributo di rumore del canale AWGN sia quello dei due dispositivi cognitivi.

3.1.3 Independence Test con solo rumore

In questa configurazione, mostrata in figura 3.14, si è voluto verificare il corretto settaggio della soglia ξ. Per fare ciò, si è considerata la sola presenza di rumore (ipotesi H_0) e si sono eliminati i blocchi che generano i campioni di segnale. Inoltre, all'interno del blocco *Probability Calculator* si cambiato il segno da "<" a ">" in accordo con (2.13).

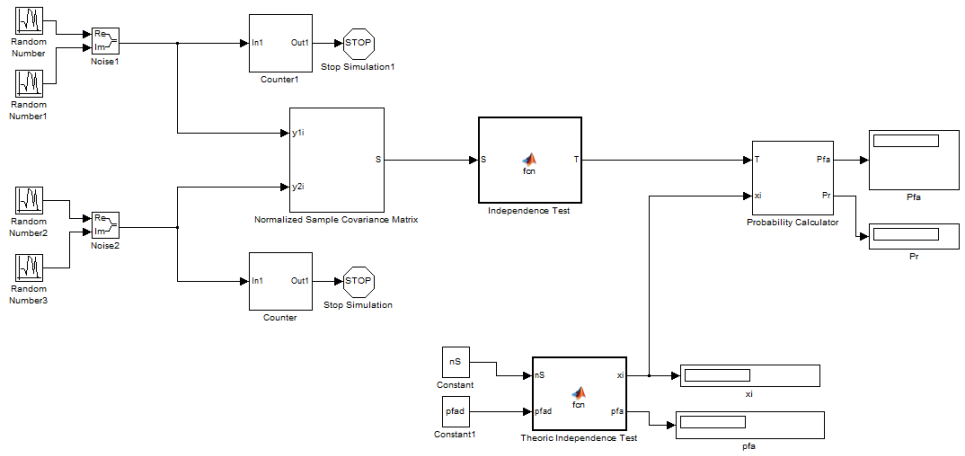

Figura 3.14: Independence Test con solo rumore

Fissando $P_{FA}^{DES} = 0.1$, n_S=500 ed n_P=50000, si ottiene effettivamente una probabilità di falso allarme P_{FA}=0.1. Si può affermare che la soglia ξ è stata fissata correttamente.

3.2 Implementazione Sphericity Test

Lo *Sphericity Test*, così come l'*Independence Test*, utilizza la matrice di covarianza campionaria. La differenza tra i due algoritmi riguarda il calcolo della soglia ξ e della metrica $T^{(sph)}$.

3.2.1 Sphericity Test con Rayleigh Fading

In figura 3.15 è mostrato il modello Simulink *SphericityTestRayChan.mdl* che implementa tale algoritmo.

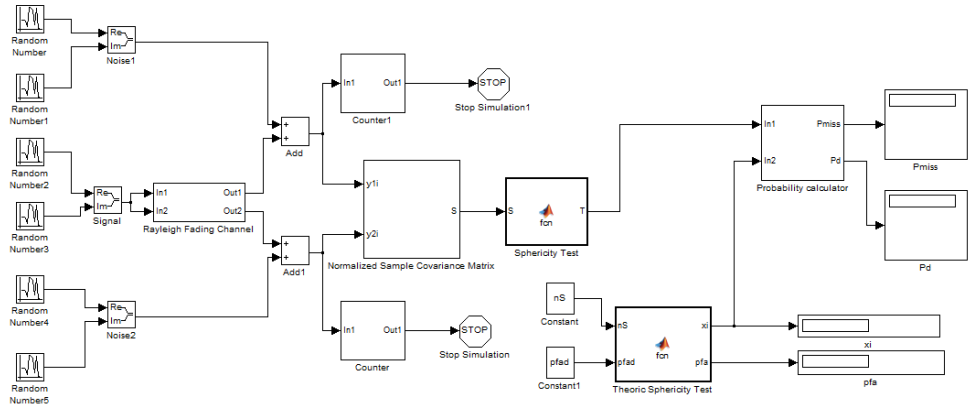

Figura 3.15: Sphericity Test con Rayleigh Fading

Così come per l'*Independence Test*, anche per lo *Sphericity Test* si è considerato il caso di canale trasmissivo con *fading* alla Rayleigh. Lo schema è praticamente lo stesso di figura 3.1, le uniche differenze sono all'interno dei blocchi *Sphericity Test* e *Theoric Sphericity Test*. All'interno del blocco *Sphericity Test* è presente il codice MATLAB che implementa la metrica $T^{(sph)}$. Riporto qui di seguito il contenuto

```
function T = fcn(S)
%# T(sph)

T = real(det(S)/((trace(S)/2)^2));
```

Il codice scritto equivale alla (2.16) nel caso n_R=2

$$T^{(sph)} = \frac{|S|}{\left[(s_{1,1} + s_{2,2})/2\right]^2} \tag{3.5}$$

dove il comando MATLAB *trace(S)* indica la traccia di **S**. Anche in questo caso si è dovuto aggiungere il comando *real* e le ragioni sono le medesime di quelle espresse nel paragrafo 3.1.1.
Nel blocco *Theoric Sphericity Test* è presente il seguente codice MATLAB per il calcolo della soglia ξ:

```
function [xi,pfa] = fcn(nS, pfad)

% Espressioni empiriche dei momenti del primo e del secondo
%ordine nel caso nR=2

m1=4*exp(gammaln(2*nS)-gammaln((2*nS)+2)+gammaln(nS+1)-
gammaln(nS)+gammaln(nS)-gammaln(nS-1));

m2=16*(exp(gammaln(2*nS)-gammaln((2*nS)+4)+gammaln(nS+2)-
gammaln(nS)+gammaln(nS+1)-gammaln(nS-1)));

% Espressioni empiriche dei coefficienti a e b

a=(m1*(m2-m1))/((m1^2)-m2);

b=((1-m1)*(m2-m1))/((m1^2)-m2);

% Espressioni empiriche della soglia xi e della probabilita'
% di falso allarme pfa

xi=real(betaincinv(pfad,a,b));

pfa=betainc(xi,nS-1,3/2);
```

Le espressioni di *m1* ed *m2* sono le espressioni (2.17) per il calcolo dei momenti di primo e secondo ordine nel caso n_R=2. Per maggiore chiarezza riscrivo di seguito le espressioni relative a tale caso

$$m_1^{(sph)} = \frac{2^2\Gamma(2n_S)}{\Gamma(2n_S+2)}\prod_{i=1}^{2}\frac{\Gamma(n_S-i+2)}{\Gamma(n_S-i+1)} \tag{3.6}$$

$$m_2^{(sph)} = \frac{2^4\Gamma(2n_S)}{\Gamma(2n_S+4)}\prod_{i=1}^{2}\frac{\Gamma(n_S-i+3)}{\Gamma(n_S-i+1)}\ . \tag{3.7}$$

Per la scrittura delle (3.6) e (3.7) in codice MATLAB si è scelto di utilizzare, al posto del classico comando *gamma(X)*, il comando *gammaln(X)* per il calcolo della funzione Gamma Γ. Tale scelta è stata dettata dal fatto che con gamma(X), per valori di X superiori a 200, si sono verificati problemi di *overflow*. Il comando *gammaln(X)* computa il logaritmo naturale della funzione *gamma(X)* e, dopo opportuni passaggio matematici e utilizzando le proprietà dei

logaritmi, si sono ricavate le espressioni di *m1* e *m2* presenti nel codice MATLAB.

Per i parametri *a* e *b* si sono utilizzate le formule (2.8) e (2.9), mentre per il calcolo della soglia ξ (nel codice MATLAB è stata indicata con *xi*) si è usato il comando *betaincinv (pfad, a, b)*. Quest'ultimo computa la funzione Beta incompleta inversa, per ulteriori delucidazioni rimando a [5].

3.2.2 Sphericity Test con canale AWGN

A differenza dello schema di figura 3.15, il modello Simulink *SphericityTestAWGN.mdl* , mostrato di seguito in figura 3.16, considera il canale trasmissivo di tipo AWGN e non con *fading* alla Rayleigh.

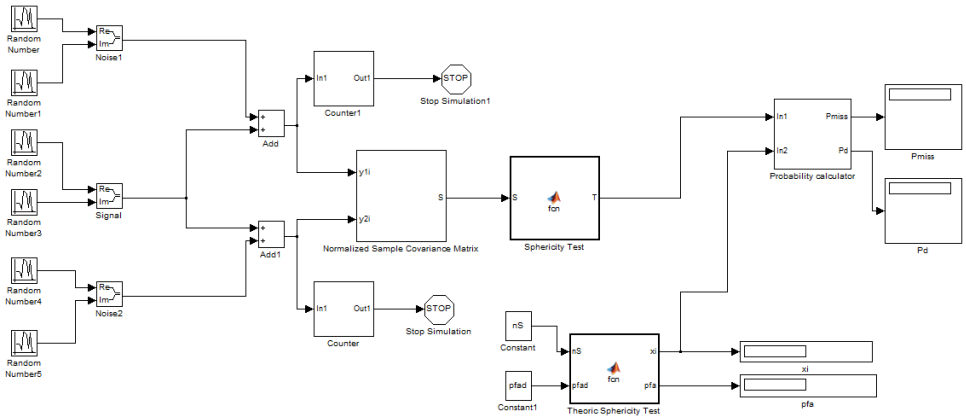

Figura 3.16: Sphericity Test con canale AWGN

Lo schema è identico a quello del paragrafo 3.2.1, l'unica differenza è data dall'assenza del blocco *Rayleigh Fading Channel*. In questo caso, i campioni di rumore comprendono sia il contributo di rumore del canale AWGN sia quello dei due dispositivi cognitivi.

3.2.3 Sphericity Test con solo rumore

Così come è stato fatto per l'*Independence Test*, anche per lo *Sphericity Test* si è voluto verificare il corretto settaggio della soglia ξ. In figura 3.17 viene mostrato il modello Simulink *SphericityTestNoise.mdl*.

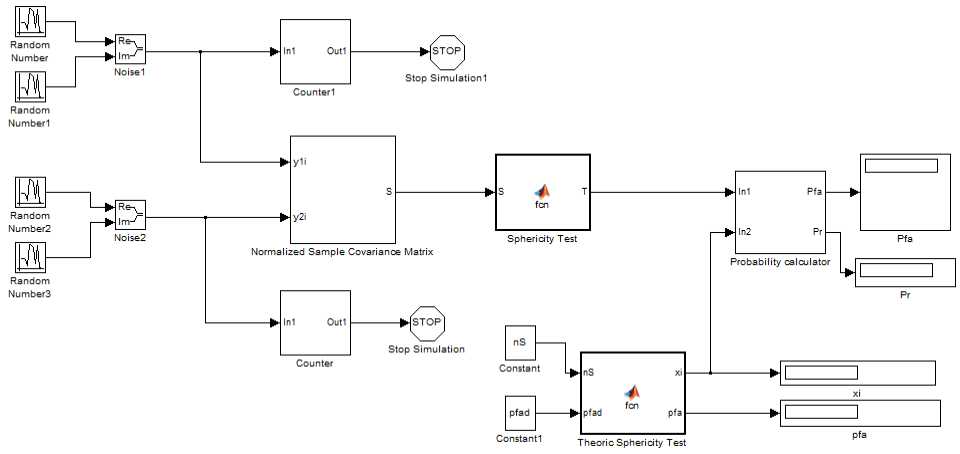

Figura 3.17: Sphericity Test con solo rumore

Si sono eliminati i blocchi che generano i campioni di segnale e di conseguenza si è considerata la sola presenza di rumore (ipotesi H_0). Inoltre, all'interno del blocco *Probability Calculator* si cambiato il segno da "<" a ">" in accordo con (2.16). Fissando $P_{FA}^{DES} = 0.1$, n_S=500 ed n_P=50000, si ottiene effettivamente una probabilità di falso allarme P_{FA}=0.1. Si può affermare che anche la soglia ξ dello *Sphericity Test* è stata fissata correttamente.

3.3 Curve ROC

Dopo aver decritto gli algoritmi in ambiente simulato, è il momento di passare all'analisi dei dati acquisiti. La prima caratteristica che si è andati ad analizzare è l'andamento della probabilità di detection P_D al variare della probabilità di falso allarme P_{FA}. Tutto ciò è possibile

rappresentarlo tramite una curva denominata *Receiver Operating Characteristic* (ROC).

Per quanto riguarda l'*Independence Test*, si sono ottenute le curve ROC (di figura 3.18) appartenenti al caso di canale AWGN e di canale alla Rayleigh, mantenendo fissi i seguenti parametri:

- Numero di campioni per una singola realizzazione → n_S=500
- Numero di prove → n_P=50000
- Rapporto Segnale-Rumore in dB → $(SNR)_{dB}$=-10 dB
- Stessa potenza di rumore per i due SUs → Δ=0 dB

Con Δ si è voluta indicare la differenza di rumore tra i due SUs.

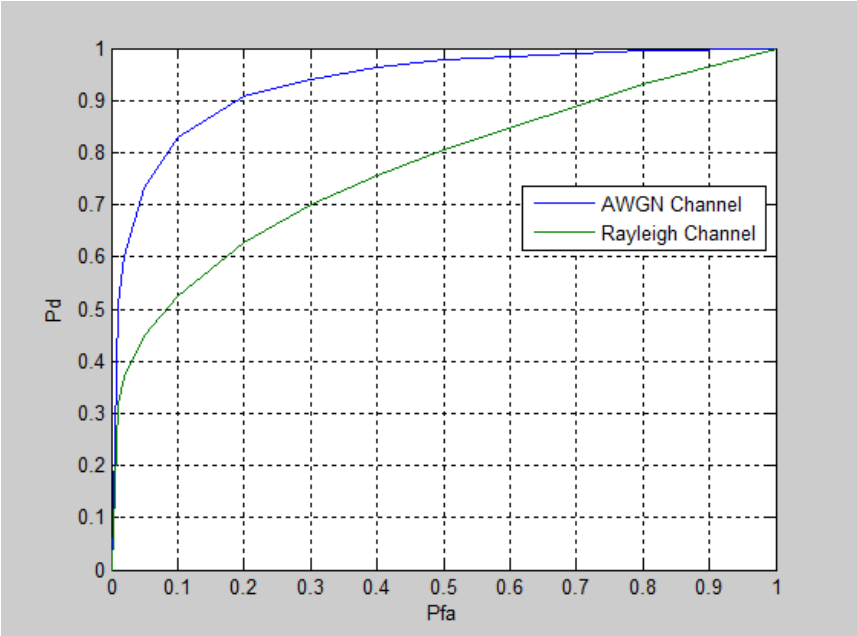

Figura 3.18: Curve ROC Independence Test

Come sperimentato in figura 3.18, l'*Independence Test* risulta più sensibile agli effetti del *fading*, infatti nel passaggio da canale AWGN a Rayleigh si ha una perdita in termini di prestazioni.

Invece, per quanto riguarda lo *Sphericity Test* si sono ottenute le curve ROC di figura 3.19 (utilizzando gli stessi valori dei parametri dell'*Independence Test*).

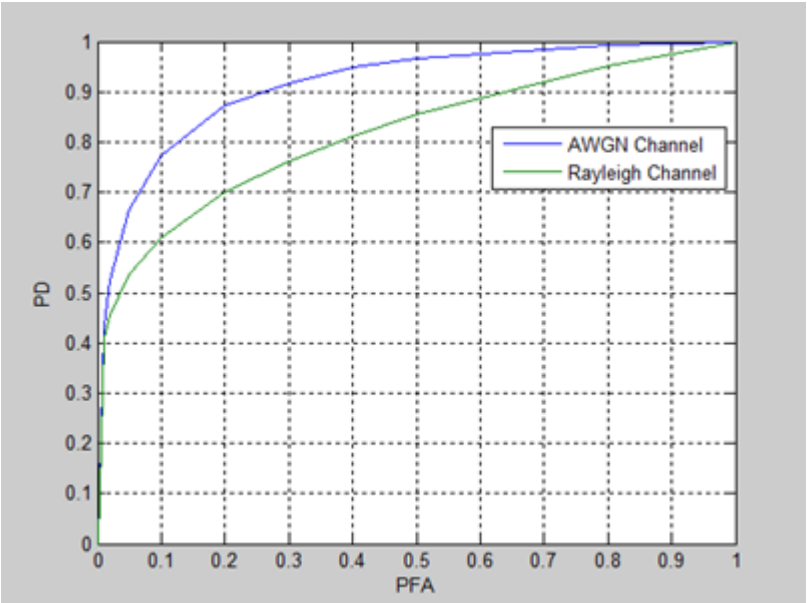

Figura 3.19: Curve ROC Sphericity Test

Stesso discorso vale anche per questo algoritmo, infatti anche lo *Sphericity Test* risulta più sensibile agli effetti del *fading*.

Per poter confrontare le prestazioni dei due algoritmi, si sono sovrapposte tutte e quattro le curve ROC come mostrato in figura 3.20.

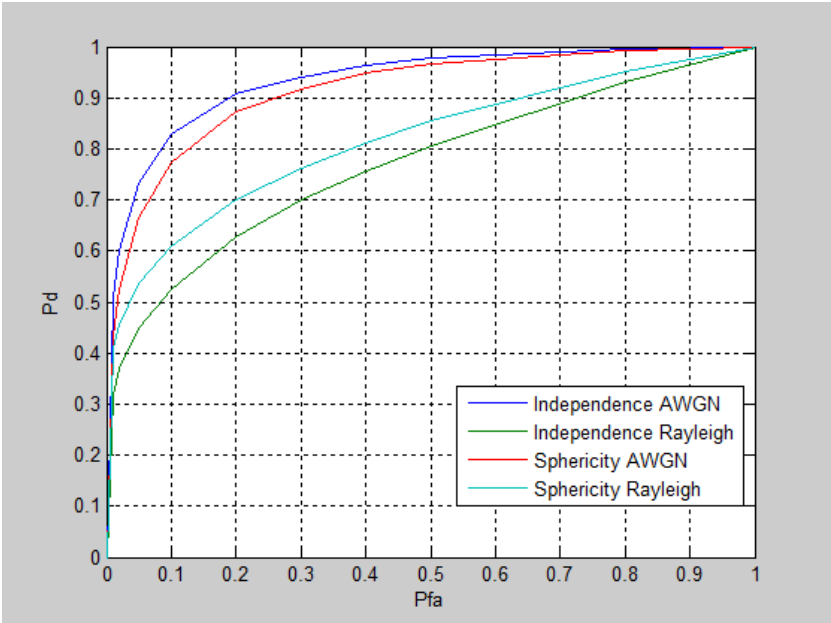

Figura 3.20: Curve ROC

Le curve di figura 3.20 rappresentano le curve ROC nel caso di n_R=2 dispositivi cognitivi riceventi e di un singolo PU. L'andamento risulta coerente con i grafici mostrati in [3]. Si può notare che, con Δ=0 dB, l'*Independence Test* fornisce prestazioni migliori rispetto allo *Sphericity Test* nel caso di canale AWGN, mentre con canale alla Rayleigh è lo *Sphericity Test* a risultare più performante.
Per migliorare le prestazioni di entrambi, si può aumentare n_S, cioè il numero di campioni per singola realizzazione.
Infine si sono confrontate le prestazioni dei due algoritmi nella situazione con Δ≠ 0, cioè con potenze di rumore dei due dispositivi diverse tra loro (ricevitori *non calibrati*). Si è posto Δ= 0.5 dB e, nel caso con *fading* alla Rayleigh, si sono ottenute le curve di figura 3.21.

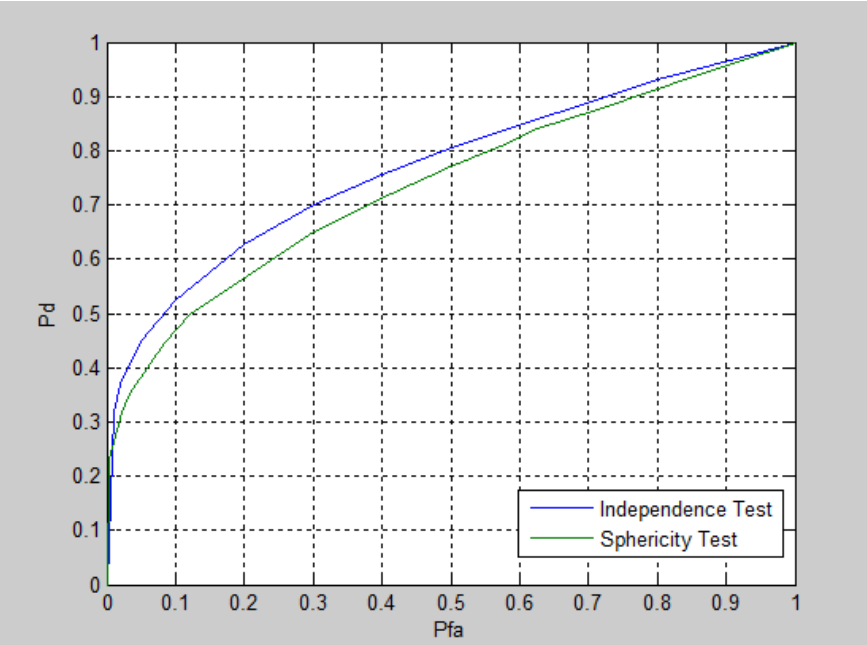

Figura 3.21: Curve ROC con Δ= 0.5 *dB*

Si può notare che l'*Independence Test*, nel caso di ricevitori non calibrati e di canale alla Rayleigh, è più robusto alle differenze di potenza di rumore rispetto allo *Sphericity Test*.

Inoltre, in figura 3.22, si sono rappresentate le curve ROC dell'*Independence Test* nel caso $\Delta = 0\ dB$ e $\Delta = 0.5\ dB$.

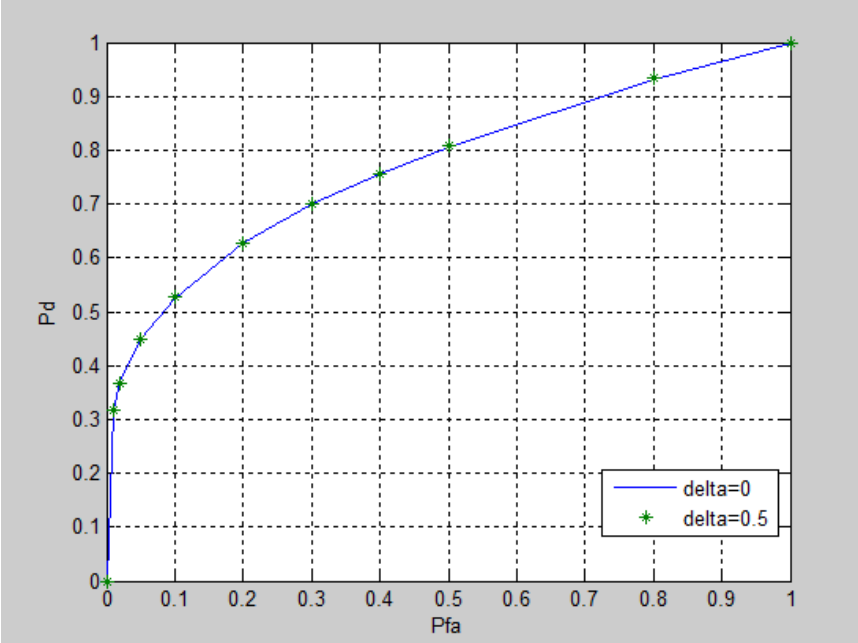

Figura 3.22: Independence Test nel caso $\Delta = 0\ dB$ e $\Delta = 0.5\ dB$

Si può notare che le due curve sono praticamente sovrapposte, quindi si può affermare che le prestazioni dell'*Independence Test* non dipendono dal valore di Δ. Quindi, in presenza di ricevitori non calibrati, è meglio adottare l'*Independence Test*, il quale risulta meno sensibile agli squilibri di potenza di rumore tra gli apparati.

3.4 Probabilità di Detection P_D in funzione di SNR

In questo caso, variando il Rapporto Segnale-Rumore (SNR), si è analizzato il comportamento della probabilità di Detection P_D. In linea teorica, se si fa tendere a $+\infty$ il SNR, la probabilità di Detection tende a 1. Se si fa tendere il SNR a $-\infty$, invece la probabilità di Detection tenderà al valore di probabilità di Falso Allarme desiderato

P_{FA}^{DES}. In figura 3.21 si è mostrato l'andamento della P_D in funzione del SNR in dB, facendolo variare nel range (10 dB÷ -30 dB).

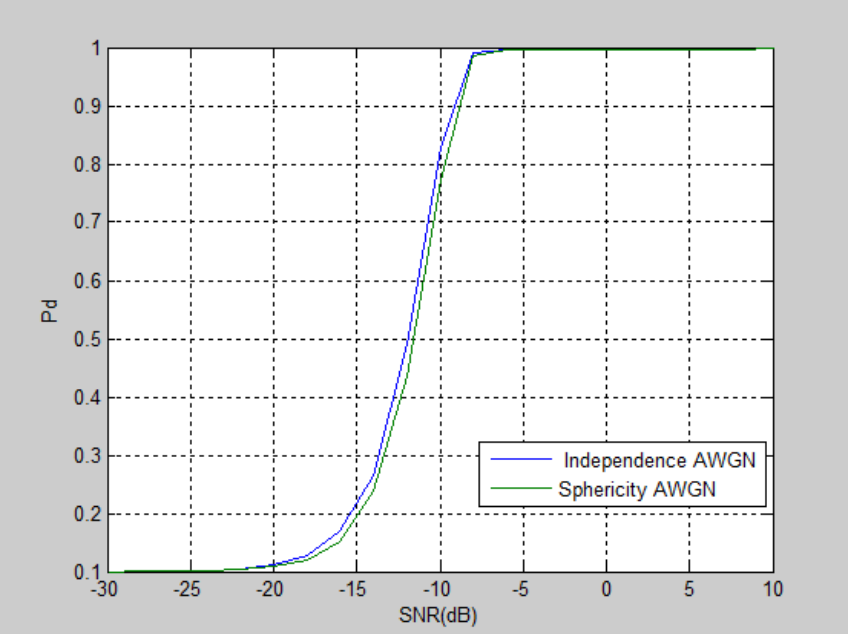

Figura 3.21: P_D in funzione di SNR

Si può notare, nel caso di canale AWGN, che l'*Independence Test* ha prestazioni migliori rispetto allo *Sphericity Test* in quanto, per un valore di SNR fissato, la probabilità di detection P_D presenta un valore più elevato. Anche questo aspetto avvalora la tesi precedente detta che l'*Independence Test* risulta essere più robusto agli squilibri di potenza di rumore tra i due apparati.

Capitolo 4

USRP2

La *Universal Software Radio Peripheral* (**USRP2**)[6] , evoluzione del precedente modello USRP, è stata messa sul mercato nel 2008 dalla Ettus Research ed è diventata in poco tempo uno dei sistemi più utilizzati nell'ambito della Radio Cognitiva.

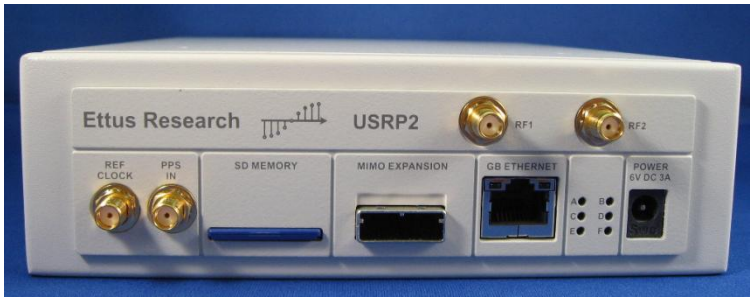

Figura 4.1: USRP2

Il dispositivo include al suo interno una struttura di elaborazione programmabile basata su FPGA e adatta per applicazioni DSP che si basano sull'analisi di forme d'onda complesse e con elevate frequenze di campionamento. Inoltre, il pacchetto GNU Radio [7] - USRP2 costituisce una piattaforma completa per lo sviluppo *Software-Defined Radio* (SDR).

A differenza della versione USRP, sono state introdotte le seguenti novità:

- Interfaccia di comunicazione con l'host tramite Gigabit Ethernet
- FPGA Xilinx Spartan 3-2000
- Doppio ADC 100 MHz 14 bit
- Doppio DAC 400 MHz 16 bit
- Ampiezza di banda RF pari a 25 MHz

- Configurazione tramite schede SD
- Realizzazione di sistemi ad antenne multiple (MIMO)
- Modalità *stand-alone*

Nel mio caso, si è deciso di non usare l'applicativo GNU Radio a favore di MATLAB, che dalla versione R2011a fornisce supporti per l'interfacciamento con le USRP2.

4.1 Principio di funzionamento USRP2

In figura 4.2 è mostrato lo schema a blocchi di un sistema USRP2-Host e per ogni blocco si può notare la loro struttura interna. In questo paragrafo, però, ci limiteremo ad analizzare solo l'architettura interna dell'USRP2.

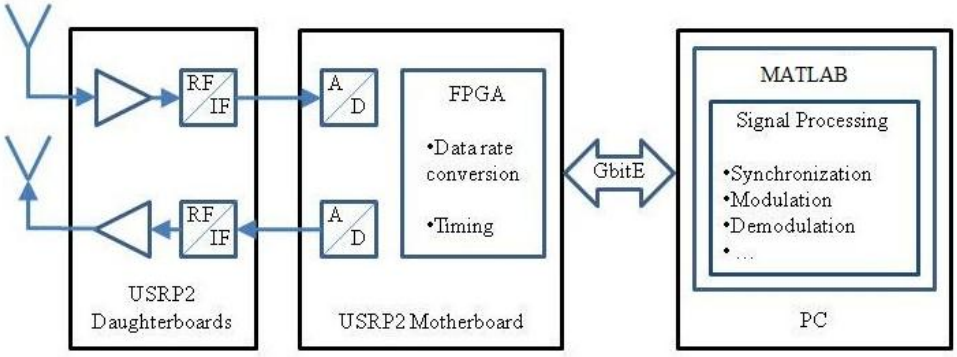

Figura 4.2: Schema a blocchi USRP2-Host

Il segnale ricevuto viene campionato dall'ADC presente sulla scheda madre e lo converte in valore digitale con risoluzione pari a 14 bit. Il numero di campioni osservabili nell'unità di tempo dipende dalla frequenza di campionamento degli ADC: l'USRP2 campiona 100 MSample /secondo.

I campioni, successivamente, sono processati tramite una DDC (*Digital Down Conversion*), la quale li demodula ad una frequenza di interesse. Infine, un decimatore pari a N viene inserito in questa

catena per poter modificare l'ampiezza di banda osservata dal dispositivo USRP2.

Al termine di questi processi di elaborazione, i campioni vengono inviati tramite interfaccia Gigabit Ethernet all'host. La velocità di throughput della Gigabit Ethernet dipende dal fattore di decimazione. Nel mio caso si è deciso di usare il fattore di decimazione N=512 (valore massimo impostabile per l'USRP2), in modo tale da avere una velocità di trasferimento dati la più bassa possibile. La decimazione N, come anticipato sopra, influisce anche sulla banda osservabile del dispositivo. Per le mie applicazioni, tale banda sarà di 200KHz.

$$B = \frac{Frequenza\ campionamento\ ADC}{Fattore\ di\ decimazione\ N} = \frac{100\ MS/s}{512} = 200\ KHz \qquad (5.1)$$

Per la trasmissione, il procedimento è inverso e si parlerà di DUC (*Digital Up Conversion*) e di DAC al posto di ADC.

4.2 Sistema utilizzato

4.2.1 Hardware e software

Il sistema scelto per l'implementazione degli algoritmi di *Spectrum Sensing* cooperativo è composto dai seguenti dispositivi hardware:

- due USRP2 con al loro interno due schede daughterboard WBX Transceiver. In questo caso sono state settate entrambe come RX/TX.
- un host PC Pentium IV, Dual Core, 3.4 Ghz, con due schede di rete Gigabit Ethernet, attraverso le quali avverrà l'interfacciamento con le USRP2
- un *HP Synthesized Sweeper 83752A* utilizzato come generatore di un segnale sinusoidale da inviare alle USRP2

- un analizzatore di spettro *HP Spectrum Analyzer 8596E* per visualizzare lo spettro del segnale sinusoidale

Lato software, il sistema operativo scelto è stato **Ubuntu 10.04 LTS**, mentre per l'implementazione degli algoritmi di Spectrum Sensing cooperativo si è scelto l'ambiente **MATLAB/Simulink R2011a** per la presenza della libreria *SDR Hardware* che permette un facile interfacciamento tra USRP2 e host.

4.2.2 Configurazione Host-USRP2

L'ambiente MATLAB/Simulink è stato scaricato da [8] in versione di prova. Questa versione per Ubuntu 10.04 LTS presenta qualche piccolo bug causato da una versione del compilatore gcc del sistema operativo non completamente supportata dall'applicazione. La guida presente in [9] permette di risolvere questi problemi e quindi di installare in modo corretta MATLAB R2011a. Nel caso in cui si dovessero riscontrare dei problemi nel punto 4) di tale guida, legati all'assenza del file *startup.m*, si consiglia di seguire anche la guida presente in [10].
Una volta terminata l'installazione, si è passati alla configurazione dell'interfacciamento tra USRP2 e Host.
Prima di tutto, è necessario sostituire il firmware delle USRP2, in modo che tale dispositivo riesca a lavorare in coppia con Simulink. Questa procedura di sostituzione avviene caricando il firmware nella scheda SD dell'USRP2 e seguendo le istruzioni presenti in [11]. Ora si può passare alla prima configurazione tra le due USRP2 e l'host. Nell'host sono stati creati due profili di connessione di rete, dedicati esclusivamente alla comunicazione con i due dispositivi. Si sono impostati due indirizzi IP statici privati alle due rispettive schede di rete Gigabit Ethernet in modo tale che risultino appartenenti a due

reti differenti, come suggerito in [12]. I valori impostati per il primo profilo, denominato “USRP2 eth0”, sono i seguenti:

- Indirizzo IP: **192.168.10.1**
- Netmask: **255.255.255.0**
- Gateway: **0.0.0.0**

Per il secondo profilo, denominato “USRP2 eth2”, si sono impostati i seguenti valori:

- Indirizzo IP: **192.168.20.1**
- Netmask: **255.255.255.0**
- Gateway: **0.0.0.0**

Come descritto in [12], è raccomandato che ad ogni interfaccia Ethernet corrisponda una sola USRP2, dove a ciascuna di queste interfacce deve essere assegnata una propria *subnet* e alla corrispondente USRP2 deve essere impostato un indirizzo IP appartenete a quella stessa *subnet*. Infatti nel mio caso si sono impostati ai due dispositivi i seguenti indirizzi IP:

- Indirizzo IP per la prima USRP2: **192.168.10.255**
- Indirizzo IP per la seconda USRP2: **192.168.20.255**

4.3 Test di configurazione

Per verificare se la configurazione USRP2-Host è stata effettuata in modo corretto, si è creato il semplice modello Simulink *ConfigurationMultiUSRP2.mdl*, riportato in figura 4.3, attraverso il quale si sono decisi di visualizzare gli spettri del segnale ricevuto dalle due USRP2.

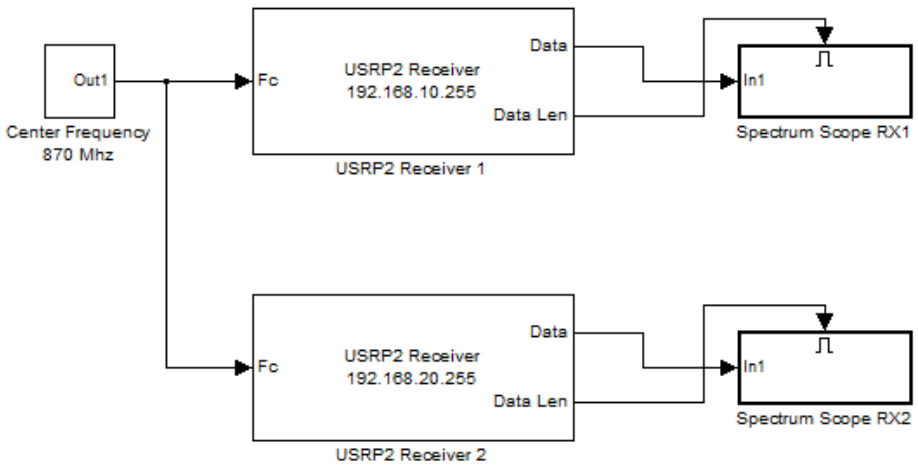

Figura 4.3: ConfigurationMultiUSRP2.mdl

Tale verifica è stata effettuata inviando un segnale sinusoidale a frequenza f_c = *870 MHz*, prelevato dall'uscita *RF OUTPUT* dello *Sweeper*, alle due USRP2 tramite collegamento diretto via cavo (figura 4.4). Infine, l'interfacciamento delle USRP2 con l'host è garantito da due cavi Ethernet.

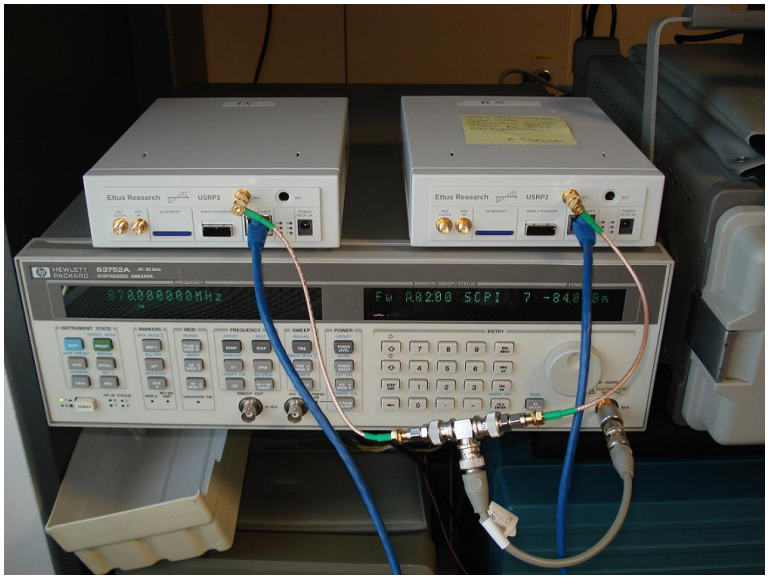

Figura 4.4: Collegamento Sweeper – USRP2

Lo strumento *HP Synthesized Sweeper 83752A* (figura 4.5) fornisce segnali in uscita, analogici o digitali, con frequenze da 2 a 20 GHz.

In questo caso, è stato utilizzato come generatore di un segnale sinusoidale analogico ed è stato collegato direttamente via cavo schermato agli ingressi delle USRP2. In [13] è stato riferito che il collegamento diretto via cavo tra due USRP2 (o tra USRP2 e un generatore di segnali) è possibile solo se è presente un attenuatore di almeno 30 dB tra di esse, altrimenti si potrebbe mettere a rischio l'integrità del sistema. Per risolvere questo problema, si è scelto di utilizzare l'uscita *RF OUTPUT* dello *Sweeper*, nella quale è presente uno stadio attenuatore integrato che limita la potenza in uscita fino a -85 dBm. Onde evitare spiacevoli inconvenienti, si è deciso di prelevare il segnale dall'uscita *RF OUTPUT* con una potenza pari a -85 dBm.

Figura 4.5: HP Synthesized Sweeper

Ora si può tornare alla descrizione del modello Simulink di figura 4.2. Il blocco *USRP2 Receiver* (figura 4.6) supporta la comunicazione fra Simulink e l'USRP2, permettendo simulazione e sviluppo di applicazioni SDR.

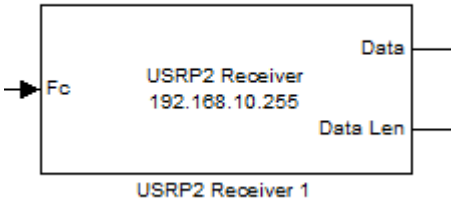

Figura 4.6: Blocco USRP2 RX

Tale blocco presenta un ingresso *Fc* e due uscite denominate *Data* e *Data Len*. Tramite un blocco *Center Frequency 870 MHz* (non è altro che un blocco *Constant*) è possibile fornire la frequenza centrale *Fc* a cui l'USRP2 deve sintonizzarsi, mentre l'uscita *Data* fornisce vettori-colonna, di lunghezza fissa 358x1, contenenti i campioni in forma complessa del segnale ricevuto. L'altra uscita *Data Len* è un controllo, infatti indica quando sono presenti dati validi.
Aprendo il blocco di figura 2, appare la finestra di figura 4.7

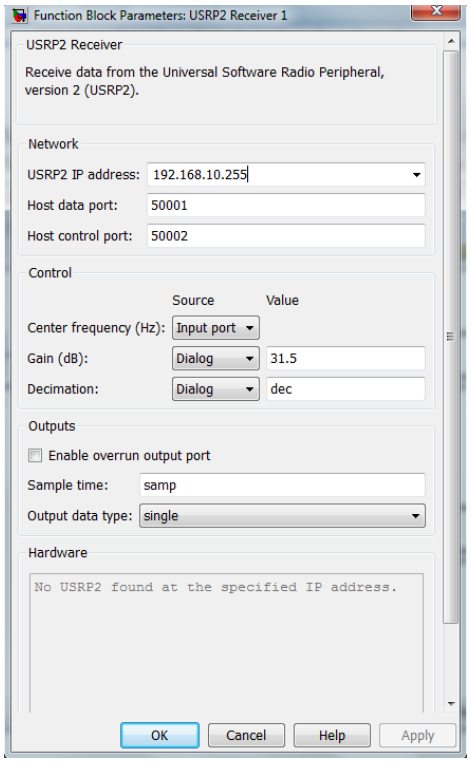

Figura 4.7: Parametri blocco USRP2 RX

Sono presenti alcune impostazioni, tra le quali:

- *USRP2 IP address*: si indica l'indirizzo IP dell'USRP2, il quale deve appartenere alla stessa subnet dell'interfaccia di rete.
- *Host data port*, *Host control port*: si specifica la porta UDP della connessione dati dell'host. Per la scelta delle porte si consiglia di consultare l'help di MATLAB.
- *Center frequency (Fc)*: si specifica la frequenza centrale del segnale in ingresso all'USRP2.
- *Gain (db)*: si specifica il guadagno che il blocco applica al *front end* dell'USRP2. Il valore massimo impostabile è 31.5 dB.
- *Decimation*: si specifica il fattore di decimazione del ricevitore. Questo dato viene utilizzato per effettuare la conversione in banda base (*downconvertion*) del segnale complesso. Il valore massimo impostabile è 512.
- *Sample Time*: si indica il periodo di campionamento. Perché il tempo di campionamento di Simulink corrisponda al tempo reale, come scritto nell'help MATLAB del blocco, occore usare la seguente formula:

$$Sample\ Time = N \cdot 1 * e^{-8}$$

  dove N rappresenta il fattore di decimazione.

Nel mio caso, si sono impostati i seguenti parametri per il blocco *USRP2 Receiver 1*:

- ***USRP2 IP address***: 192.168.10.255
- ***Host data port***: 50001
- ***Host control port***: 50002
- ***Center frequency***: 870 Mhz
- ***Gain***: 31.5 dB

- ***Decimation***: *dec* = 512
- ***Sample time***: *samp* = *dec*/100e6

Mentre per il blocco *USRP2 Receiver 2*:

- ***USRP2 IP address***: 192.168.20.255
- ***Host data port***: 30000
- ***Host control port***: 30002
- ***Center frequency***: 870 Mhz
- ***Gain***: 31.5 dB
- ***Decimation***: *dec* = 512
- ***Sample time***: *samp* = *dec*/100e6

I blocchi *Spectrum Scope RX* sono dei sottosistemi con comando di abilitazione (*Enable*) utilizzati, in questo caso, per contenere il blocco *FFT Scope* per visualizzare lo spettro del segnale ricevuto. Come detto in [14], si consiglia di collegare sempre l'uscita *Data Len* del blocco *USRP2 Receiver* con l'ingresso *Enable* di un blocco *Enabled Subsystem*. L'interno del *Spectrum Scope RX* è mostrato in figura 4.8.

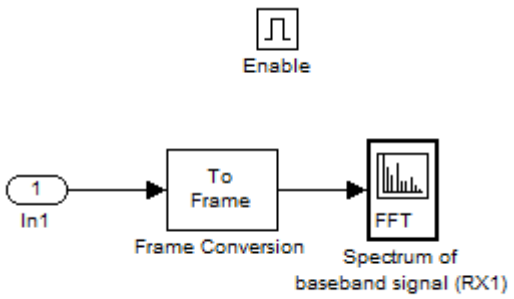

Figura 4.8: Interno blocco Spectrum Scope RX

Il blocco *FFT Scope* (rinominato in questo caso *Spectrum of base band signal RX*) permette la visualizzazione dello spettro del segnale

ricevuto dalla USRP2. Per verificare la corretta ricezione di entrambe le USRP2, si sono quindi confrontati visivamente i due spettri ottenuti tramite *FFT Scope*. In figura 4.9 e 4.10 si sono riportati tali spettri.

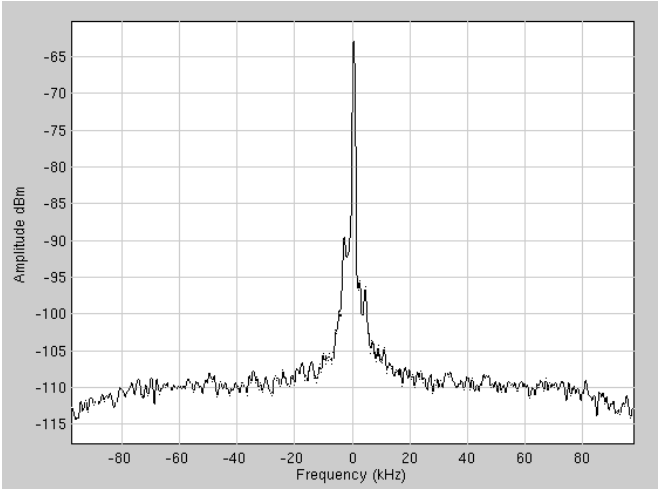

Figura 4.9: Spettro su RX1

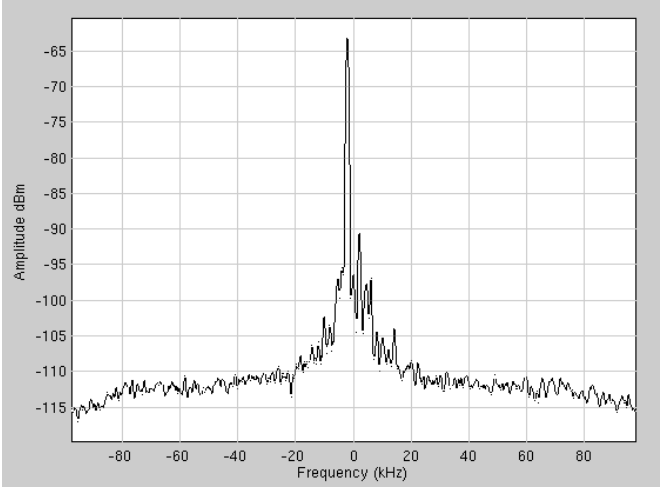

Figura 4.10: Spettro su RX2

Confrontando in maniera visiva i due spettri ottenuti, si nota che il picco a 870 MHz è ben visibile e le potenze sono pressoché simili in entrambe le USRP2 (verificando con un analizzatore di spettro il picco è in realtà a circa -85 dBm, purtroppo Simulink utilizza un

fattore di scala diverso per la visualizzazione e quindi sembra sia a circa -63 dBm).
Si può affermare che il test di configurazione USRP2-Host è andato a buon fine, riportando risultati più che accettabili.

Capitolo 5

Prove su banco

In questo capitolo testati gli algoritmi di *Independence Test* e *Sphericity Test* in ambiente reale, abbandonando quindi il mondo della pura simulazione. Le variabili globali utilizzate, presenti nello script *Variabili.m*, sono state impostate nel seguente modo:

- $\boldsymbol{n_S}$ = 500
- $\boldsymbol{n_P}$ = 50000
- ***pfad*** = 0.1
- ***dec*** =512
- ***samp*** = *dec*/100e6

Verranno riportante ed evidenziate solo le modifiche rispetto ai modelli Simulink puramente simulati per evitare futili ripetizioni.

5.1 Independence Test reale

Lo schema di figura 5.1 rappresenta l'algoritmo di *Independence Test* testato in ambiente reale. Si è utilizzato, come segnale ricevuto dalle USRP2, quello prelevato dall'uscita *RF OUTPUT* dello *Sweeper HP* alla potenza di -85 dBm.

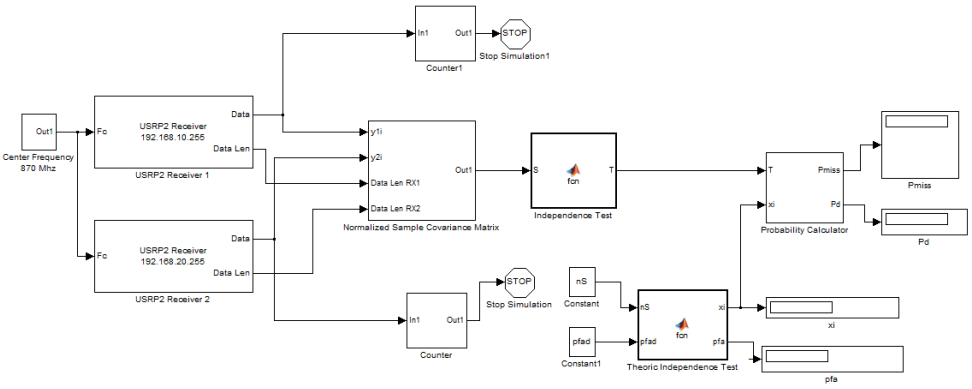

Figura 5.1: Independence Test Reale

Il modello *Usrp2RxIT.mdl* è identico al caso simulato di paragrafo 3.1.2, le uniche due differenze sono le seguenti:

- la presenza dei due blocchi *USRP2 Receiver* al posto dei blocchi *Random Number* per la generazione del segnale e del rumore;
- il blocco *Normalized Sample Covariance Matrix*. Al suo interno si presenta come in figura 5.2:

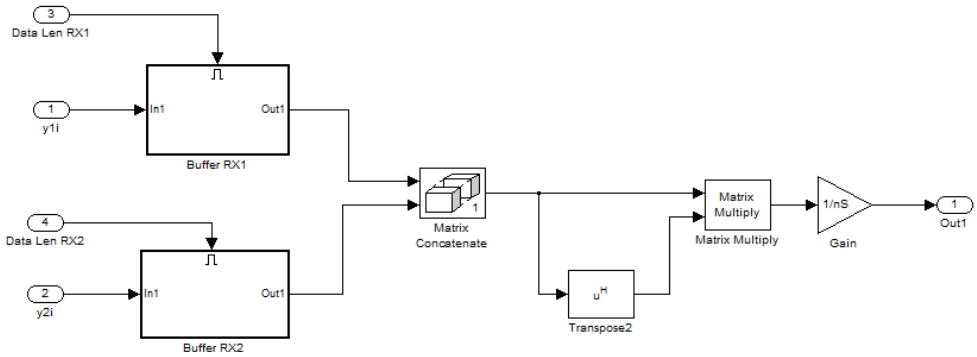

Figura 5.2: Interno blocco Normalized Sample Covariance Matrix

Si sono aggiunti i due *Enabled Subsystem*, chiamati *Buffer RX1* e *Buffer RX2*, in quanto, come detto in [14], si consiglia di collegare le uscite *Data Len* al comando di abilitazione *Enable* di tali blocchi per una migliore gestione del flusso di dati validi.

Il contenuto dei due *Enabled Subsystem*, illustrato in figura 5.3, non è altro che un *Buffer* per la raccolta di n_S campioni e un blocco *Transpose*.

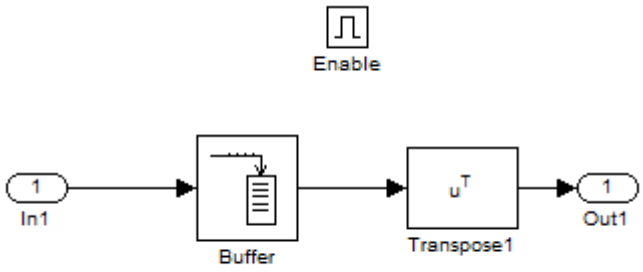

Figura 5.3: Interno blocchi Enabled Subsystem

5.2 Sphericity Test reale

Il modello Simulink *Usrp2RxST.mdl* che implementa l'algoritmo di *Sphericity Test* nel caso reale è illustrato in figura 5.4.

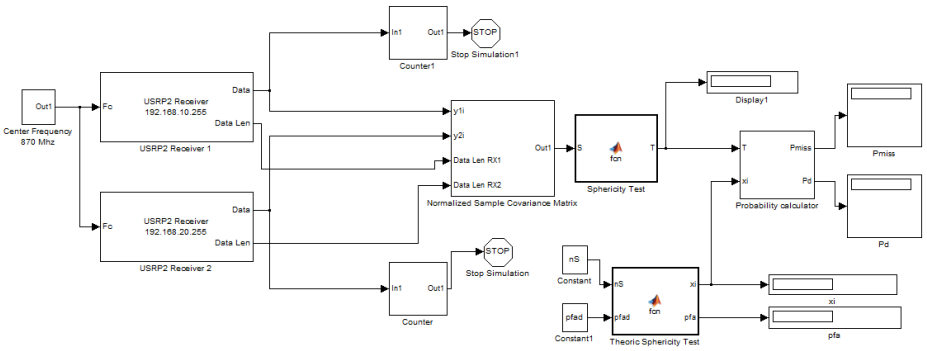

Figura 5.4: Sphericity Test Reale

Lo schema a blocchi sopra riportato è identico al caso simulato di paragrafo 3.2.2 e si sono riportate le stesse modifiche illustrate nel paragrafo 5.1.

5.3 Il concetto di "potenza virtuale"

Finora ci si è limitati allo studio delle prestazioni degli algoritmi di *Independence Test* e di *Sphericity Test* nel caso di potenza di segnale ricevuto dalle due USRP2 pari a -85 dBm. È interessante effettuare prove in ambiente reale con potenza del segnale ricevuto inferiore ai -85 dBm. Nel mio caso, il valore -85 dBm rappresenta un limite sotto al quale non si può andare in quanto l'uscita *RF OUTPUT* dello *Sweeper HP* non può scendere oltre tale valore. Per ovviare a questa barriera imposta dallo strumento, si è dovuto ricorrere ad un piccolo "trucco".

Siccome non si può abbassare la potenza del segnale oltre a -85 dBm, di conseguenza il Rapporto Segnale-Rumore SNR non può essere abbassato ulteriormente dato che la potenza del rumore delle due USRP2 è circa costante e nota. È qui che c'è bisogno di attuare il "trucco".

Il rapporto Segnale-Rumore in lineare è così definito

$$SNR = \frac{P_R}{\sigma^2_{USRP2}} \tag{5.1}$$

dove P_R è la potenza del segnale ricevuto e σ^2_{USRP2} è la potenza di rumore delle due USRP2. Nel mio caso, per abbassare il SNR si può solo aumentare la potenza di rumore in quanto la P_R è fissa a 3.16 nW (cioè -85 dBm). Per fare ciò, si somma a σ^2 (supposto noto e uguale per entrambe le USRP2) un rumore artificiale aggiuntivo σ^2_A. Quindi la nuova formula di SNR è la seguente:

$$SNR = \frac{P_R}{\sigma^2_{USRP2} + \sigma^2_A} = \frac{P_R^{(v)}}{\sigma^2_{USRP2}} \tag{5.2}$$

dove con $P_R^{(v)}$ si indica la "potenza virtuale" del segnale ricevuto. L'aggiunta dell'aggettivo "virtuale" sta ad indicare che questa è una potenza fittizia dato che, aggiungendo la quantità di rumore σ_A^2, è come se la potenza del segnale ricevuto fosse diminuita. Invertendo la (5.2), si ottiene l'espressione della potenza virtuale

$$P_R^{(v)} = P_R \cdot \frac{\sigma_{USRP2}^2}{\sigma_{USRP2}^2 + \sigma_A^2} \tag{5.3}$$

dove P_R e σ_{USRP2}^2 sono noti.
Quindi, variando la potenza artificiale aggiuntiva σ_A^2 si ottiene il valore desiderato di $P_R^{(v)}$. Il valore di σ_A^2 da aggiungere affinché si ottenga la $P_R^{(v)}$ desiderata è dato dalla seguente relazione ottenuta, dopo opportuni matematici, dalla (5.3):

$$\sigma_A^2 = \sigma_{USRP2}^2 \cdot \left(\frac{P_R}{P_R^{(v)}} - 1 \right) \tag{5.4}$$

Per evitare incongruenze tra unità di misura, P_R e $P_R^{(v)}$ vanno espresse sempre in Watt [W], mentre σ_{USRP2}^2 e σ_A^2 si esprimono in Volt^2 [V^2].
Per sistemi a due antenne riceventi, il Rapporto Segnale-Rumore SNR in lineare è possibile esprimerlo nella seguente forma:

$$SNR = \frac{P}{\sigma^2} = \frac{SNR_1 + SNR_2}{2} = \frac{\frac{P}{\sigma_1^2} + \frac{P}{\sigma_2^2}}{2} \tag{5.5}$$

dove σ_1^2 e σ_2^2 rappresentano rispettivamente le potenze di rumore delle due USRP2. Nel caso σ_1^2=σ_2^2, nella (5.4) tali valori si sostituiscono al posto di σ_{USRP2}^2. Invece, per $\sigma_1^2 \neq \sigma_2^2$, al posto di σ_{USRP2}^2 si sostituisce il valore ottenuto dalla seguente espressione:

$$\sigma_{USRP2}^2 = 2 \cdot \frac{\sigma_1^2 \cdot \sigma_2^2}{\sigma_1^2 + \sigma_2^2} . \tag{5.6}$$

5.4 Independence Test reale con rumore additivo

Dopo aver enunciato in linea teorica il concetto di "potenza virtuale", è il momento di passare alla pratica. Il modello Simulink *Usrp2RxITVirtualPower.mdl*, mostrato in figura 5.5, rappresenta l'implementazione dell'algoritmo di *Independence Test* reale di paragrafo 5.1 con rumore artificiale additivo σ_A^2.

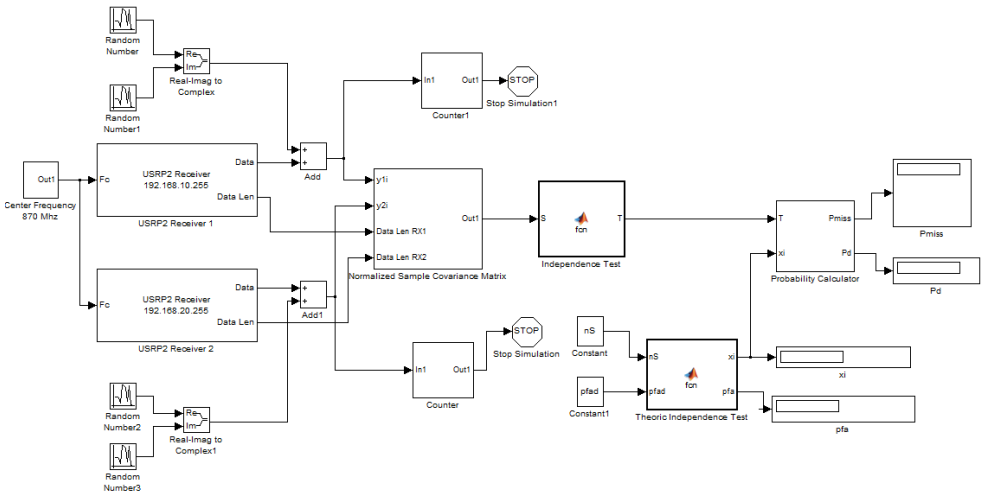

Figura 5.5: Independence Test Reale con aggiunta di rumore

L'aggiunta del rumore artificiale è stata simulata con la presenza di blocchi *Random Number*, i quali generano campioni complessi di rumore avente potenza σ_A^2. Le due potenze di rumore devono avere la stessa varianza e i valori incorrelati tra loro (cioè i blocchi *Random Number* devono avere *Seed* diverso). Note la potenze di rumore delle due USRP2 σ_1^2 *e* σ_2^2, ricavate da [15], si applica la formula (5.6) per ricavare il valore σ_{USRP2}^2. Questo valore va inserito nella (5.4) e ci si ricava σ_A^2. La metà del valore ottenuto va inserito nel campo *Variance* dei blocchi *Random Number* che generano rumore per i motivi esplicitati nel paragrafo 3.1.1.

5.5 Sphericity Test reale con rumore additivo

Il modello Simulink *Usrp2RxSTVirtualPower.mdl*, mostrato in figura 5.6, rappresenta l'implementazione dell'algoritmo di *Sphericity Test* reale di paragrafo 5.2 con rumore artificiale additivo σ_A^2.

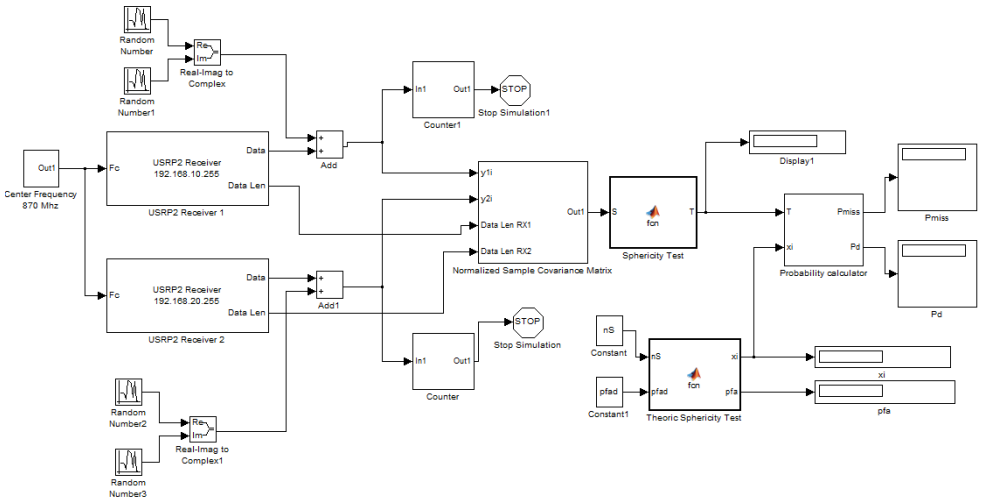

Figura 5.6: Sphericity Test Reale con aggiunta di rumore

L'aggiunta del rumore artificiale è stata simulata nella stessa maniera discussa nel paragrafo 5.4.

5.6 Analisi dei dati nel caso reale

Purtroppo, in tutti gli algoritmi reali, non è stato possibile effettuare alcuna analisi dei dati acquisiti a causa della loro non veridicità. I motivi di ciò sono riassunti nelle seguenti due affermazioni:

- le due USRP2 non sono esattamente sincronizzate, di conseguenza è presente uno sfasamento tra un campione di segnale e l'altro che altera completamente la veridicità della matrice di covarianza campionaria;

- i blocchi *Random Number* non hanno lo stesso periodo di campionamento delle due USRP2.

La mia intenzione era quella di confrontare i dati e le curve ottenute nei paragrafi 3.3 e 3.4 (acquisite nel caso simulato) con quelle ottenute per il caso di ambiente reale.

Conclusioni

In questa tesi si sono sviluppati alcuni temi riguardanti la Radio Cognitiva. È stato affrontato il problema della scarsa efficienza nell'utilizzo dello spettro e si è introdotto il concetto di Radio Cognitiva a partire dalla definizione primordiale di Joseph Mitola III. Si sono illustrate le diverse tecniche di *spectrum sensing*, soffermandosi in particolare su quelle di tipo cooperativo.

Successivamente, sono stati introdotti due algoritmi di *sensing* come l'*Independence Test* e lo *Sphericity Test*. Entrambi gli algoritmi sono stati implementati in Simulink, utilizzando sia segnali simulati che segnali realmente acquisiti. L'analisi dei risultati ottenuti dagli algoritmi con segnali simulati porta ad affermare che l'*Independence Test* ha prestazioni migliori rispetto allo *Sphericity Test* in presenza di squilibri di potenze di rumore tra due sistemi cognitivi.

Poi si è analizzato il dispositivo USRP2, una piattaforma radio che sfrutta a pieno le potenzialità dell'approccio SDR. L'utilizzo di due USRP2 e di un host, nel quale è stato installato MATLAB\Simulink R2011a, ha consentito la realizzazione di un sistema cognitivo ricevente in grado di implementare algoritmi di *sensing*.

Gli algoritmi di *Independence Test* e di *Sphericity Test* sono stati testati anche in ambiente reale anche se non è stato possibile ottenere risultati veritieri in quanto le due USRP2 forniscono campioni di segnale ad instati di tempo non sincronizzati tra loro. Nonostante ciò, l'utilizzo di Simulink con i sistemi USRP2 rappresenta un interessante approccio per la realizzazione di algoritmi di *sensing* a causa della sua versatilità e relativa semplicità di implementazione.

BIBLIOGRAFIA

[1] J. Mitola III, "*Cognitive Radio An Integrated Agent Architecture for Software Defined Radio*", PhD thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2000.

[2] T. Yucek and H. Arslan, "*A survey of spectrum sensing algorithms for cognitive radio applications*", IEEE Tutorials, vol. 11, no. 1, First Quater 2009.

[3] A. Mariani, A. Giorgetti, M. Chiani, "*Test of Independence for Cooperative Spectrum Sensing with Uncalibrated Receivers*", accepted for pubblication in Proc. IEEE Global Comm. Conf. (GLOBECOM 2012), Dicembre 2012.

[4] S. Haykin, "*Cognitive Radio: Brain-Empowered Wireless Communications*", IEEE Journal in Selected Area of Communication, vol.23, no.2, Febbraio 2005.

[5] http://www.mathworks.it/help/

[6] https://www.ettus.com/product/category/USRP_Networked_Series.

[7] http://gnuradio.org/redmine/projects/gnuradio/wiki.

[8] http://www.mathworks.it/programs/trials/trail_request.html?eventid =562747565prodcod=ML&s_cid=SA_Sol_trial .

[9] https://help.ubuntu.com/community/MATLAB

[10] http://carmaux.cs.gsu.edu/change_MATLAB_default_dir.html.

[11] P. Sanzani, "*Implementazione di algoritmi per lo Spectrum Sensing su piattaforme per Radio Cognitiva*", Università di Bologna, Dicembre 2011.

[12] http://files.ettus.com/uhd_docs/manual/html/usrp2.html #addressingthedevice

[13] Ettus Research LLC; http://www.ettus.com

[14] http://lists.ettus.com/pipermail/usrp-users_lists.ettus.com/

[15] E. Forti, "*Analisi e implementazione di algoritmi di detection per sistemi ad antenne multiple su piattaforma USRP2*", Seconda Facoltà di Ingegneria, Università di Bologna, Luglio 2012.

[16] M. Chiani, "*Notes on Detection and Estimation*", appunti del corso di *Progetto di Sistemi Wireless*, Seconda Facoltà di Ingegneria, Università di Bologna, 2003.

RINGRAZIAMENTI

Finalmente, dopo innumerevoli peripezie, sono arrivato alla conclusione di questo ciclo di studi. Ci sono stati momenti di gioia ma anche tanti di sconforto, comunque l'importante è riuscire a raggiungere il traguardo prefissato. Come ogni cosa, anche questa me la sono dovuta sudare.

Ringrazio il Professor Andrea Giorgetti e l'Ing. Andrea Mariani, che senza di loro non avrei mai potuto portare a termine questa tesi, ma soprattutto li ringrazio per la loro completa disponibilità e per la pazienza che hanno avuto nei miei confronti.

Un grazie enorme lo rivolgo ai miei genitori, Paolo e Maria Grazia, e ai miei familiari, in particolare a mio nonno Giovanni, senza i quali non sarei andato mai da nessuna parte. Mi hanno sostenuto economicamente, mi hanno spronato nei momenti di difficoltà, mi hanno rimproverato, ma tutto questo lo hanno fatto solo e esclusivamente per il mio bene. Non smetterò mai di ringraziarvi.

E come dimenticarsi del mio gruppo di amici, che con la loro compagnia mi fanno sempre trascorrere bellissimi momenti, mi permettono di svagarmi e staccare un po' la spina dallo studio. Vi meritate di essere tutti menzionati, ma un semplice nome a fine della mia tesi non renderebbe l'idea di quanto possa tenere a voi. Sono veramente fiero di avere amici come voi, non avrei potuto chiedere di meglio.

GRAZIE A TUTTI