System Requirements. Channel Model Prediction

Vanderpypen, Joël; Schumacher, Laurent

Publication date:
2008

Document Version
Early version, also known as pre-print

Link to publication

Citation for pulished version (HARVARD):
Vanderpypen, J & Schumacher, L 2008, *System Requirements. Channel Model Prediction: FP6 IST SURFACE D2.2 v2.0*..

IST-4-027187-STP-SURFACE

Start date of project: 1st January 2006

Duration: 36 months

Project co-funded by the European Commission within the Sixth Framework Programme

D2.2 v2.0

System Requirements. Channel Model Prediction

Contractual date of delivery to the EC: December 2006

Actual date of delivery to the EC: January 2008

Editor (s): L. Schumacher

Author (s): J. Vanderpypen, L. Schumacher

Workpackage: 2

Estimated person months: 15

Dissemination level[1]: PU

Nature: Report

Version: 2.0

Total number of pages: 86

[1] PU: Public, PP: Restricted to other programme participants (including the Commission Services), RE: Restricted to a group specified by the consortium (including the Commission Services), CO: Confidential, only for members of the consortium (including the Commission Services)

EXECUTIVE SUMMARY

The SURFACE (Self Configurable Air Interface) project aims at studying and evaluating the performances of a novel generalised air interface capable of self-reconfiguring in order to satisfy global network QoS (Quality of Service) requirements. It considers Multiple Input Multiple Output (MIMO) technologies as an option.

Wireless MIMO channels are being investigated extensively nowadays. Their potential benefits for wireless communications, their higher throughput and the increase of their QoS is desired, and needed by new services of the third generation of cellular networks (3G) and its evolutions (Beyond 3G, B3G).

To improve wireless transmissions, the state of the channel should be known by the transmitter to adapt its transmission policies. So the receiver has to send feedback on the channel state to the transmitter. But when this feedback reaches the transmitter side, it is likely to be outdated, and therefore useless. However, if we use the current channel state to predict its likely future state, we are able to transmit more efficiently.

In this deliverable, we aim at extending existing SISO or MIMO simplified channel models that are suited for designing channel predictors. The document first presents the WINNER's SCME channel model, which will be the reference channel for testing predictors. Afterwards it introduces the SURFACE channel model, which is derived from the WINNER's SCME, with parameters adapted to the requirements of the other workpackages.

Several simplified channel models and predictors are then introduced. Two of them seemed particularly relevant for our research work, namely the original ESPRIT algorithm and a modified version by Andersen et al, as having limited computer power requirements. So we investigated and tested them thoroughly. Actually, these two predictors are only for SISO channels, so we extended them into MIMO models.

After several tests which revealed rather good performances, and low quantization feedback robustness, these two MIMO predictors were merged into a single one, taking best of both. The new predictor requires 5 channel observations for a 2x2 MIMO channel, and 17 if 4x4 MIMO. The feedback can be quantized on 2x 4 bits per complex MIMO coefficient without decreasing prediction accuracy. It means it requires 160 bits, collected during 18.75 ms (37.5 LTE TTI) to set up a 2x2 MIMO predictor, and 2,176 bits collected during 63.75 ms (127.5 LTE TTI) to set up a 4x4 one. As far as noise robustness is concerned, performances are similar at SNR of 20 dB and SNR of 100 dB. Performance decreases when SNR is only 3 dB.

We finally investigate to topic of rank prediction of the MIMO channel matrix. For that goal, we use the MIMO predictor to get future MIMO channel matrices, and after their quantization, we compute its rank. It seems a floating point quantization is required to avoid zero matrices after rounding, and only 2x 4 bits per complex MIMO channel coefficient is optimal to get the best results.

Due to a late start, a first restricted version of this deliverable was released in January 2007, only reflecting the work of four person months. This final release reflects the work of fifteen person months.

CONTENT LIST

LIST OF FIGURES

LIST OF SYMBOLS

- n_T - number of transmit antennas
- n_R - number of receive antennas
- $n_{Samples}$ - number of time samples considered by the SURFACE channel model
- n_{Sub} - number of subcarriers considered by the SURFACE channel model
- n_{Symb} - number of symbols per PRB
- n_{Taps} - number of taps considered by the SURFACE channel model
- $h^{m\,n}$ - SISO component of a MIMO channel, relevant for the link from the mth transmitting antenna to the nth receiving one.
- $\hat{h}^{m\,n}$ - predicted value of $h^{m\,n}$
- a_k - amplitude for the kth path
- ϕ_k - spatial Doppler shift for the kth path
- τ_k - tap delay for the kth path
- θ_k - angle between the motion of the receiver and the direction of waves impinging from the kth scatterer
- N_s - number of scatterers local to the receiver (and consequently, the number of NLOS paths)
- $k_\lambda = \dfrac{2\pi}{\lambda}$ - wave number, where λ is the wave length
- $z_k = e^{j\,\phi_k}$ - signal poles
- Φ - diagonal matrix containing the signal poles
- Z - Vandermonde matrix containing the signal poles
- R - correlation matrix of the channel
- $x(t)$ - transmitted signal
- $y(t)$ - received signal
- $w(t)$ - additive noise
- D - degree of the polynomial estimator
- E - number of observations used to compute a predictor
- $\mathcal{H}$ - $n \times m$ Hankel matrix containing the channel observations
- H - MIMO $n_T \times n_R$ matrix, containing the channel observations for a given time step

- $.^{\#}$ - Pseudo inverse matrix operator
- $.^{*}$ - Conjugate transpose matrix operator
- $\overline{\bullet}$ - Complex conjugate matrix operator
- $\cdot_{UP}$ - Matrix operation, removes the last row of a matrix
- $\cdot_{DOWN}$ - Matrix operation, removes the first row of a matrix
- I_n - the $n \times n$ identity matrix

LIST OF ACRONYMS

3GPP	3rd Generation Partnership Program
CPU	Central Processing Unit
CSI	Channel State Information
ESPRIT	Estimation of Signal Parameters by Rotational Invariance Techniques
IST	Information Society Technologies
LOS	Line-of-sight
MECoM	MIMO ESPRIT based on Correlation Matrix
MEHaM	MIMO ESPRIT based on Hankel Matrix
MIMO	Multiple Input Multiple Output
MISO	Multiple Input, Single Output
NMSE	Normalized Mean Square Error
NLOS	Non Line Of Sight
PRB	Physical Resource Block
QoS	Quality of Service
ROMANTIK	ResOurce Management and AdvaNced Transceiver algorIthms for multihop networKs
Rx	Receiving antenna
SCM	Spatial Channel Model
SCME	Spatial Channel Model Extended
SISO	Single Input Single Output
SNR	Signal to Noise Ratio $\equiv 10 \log_{10} \frac{mean\left(\lvert h\rvert^2\right)}{mean\left(\lvert w\rvert^2\right)}$; in decibels (dB)
SURFACE	Self Configurable Air Interface
SVD	Singular Value Decomposition
TDD	Time Division Duplex
Tx	Transmitting antenna
WINNER	Wireless World Initiative New Radio

1 INTRODUCTION

Multiple Input Multiple Output (MIMO) technologies are now really attractive because of their benefits: using several antennas both on transmitter and receiver sides can improve the amount of transmissible information if there is sufficient decorrelation between the antenna pairs. These benefits will be useful to propose new wireless multimedia services, because they require good quality channels, with high data rates.

To properly exploit time varying wireless channel, the transmitter needs to collect knowledge about the channel state, to use the right power level to transmit, or to wait a bit if the channel can not accept information currently, but in TDD modes. The transmitter can not collect knowledge about the state of the channel, unless the receiver feeds back some information. However, when the transmitter receives the Channel State Information (CSI) fed back by the receiver, this CSI can be outdated… So to evaluate the current channel state, the transmitter shall make "guesses", to predict the channel state. Hence, this is the reason why predictions are so useful in wireless communication.

In this deliverable, we aim at extending existing SISO[1] and MIMO simplified channel models in order to design a MIMO predictor of the channel state.

First, we will present the WINNER's SCME channel model. WINNER[2] is a European project [WIN], and it has implemented SCME in Matlab; which is an extension of the 3GPP Spatial Channel Model. We will use it to provide us reference CSI for our computer simulations.

We will then present the SURFACE channel model, which is a wrapper of the WINNER's SCME, where we updated a few parameters to match other SURFACE workpackages requirements.

An overview of relevant channel models and predictors will follow. Two of them revealed particularly interesting and appropriate for our working objectives. They are both based upon the ESPRIT[4] algorithm. We will detail them, and explain their mathematical background. Following this description, we will give results of computations about these two predictors.

Actually, these two predictors are only SISO predictors, so we will extend them into MIMO predictors, and then present some computation results. We will also investigate the robustness against limited feedback quantization used to train the MIMO predictors.

We will then merge the two predictors into a single one, using best of both techniques, and present its performance.

Finally, we will investigate the topic of rank prediction of the MIMO channel matrix. Actually, the whole matrices will be predicted by the MIMO predictor, and then their rank will be assessed.

The document ends with concluding remarks and future work items.

[1] Single Input Single Output

[2] Wireless World Initiative New Radio

[4] Estimation of Signal Parameters by Rotational Invariance Techniques

2 THE WINNER'S CHANNEL SIMULATOR

This section shortly introduces the WINNER's SCME, a comprehensive spatial channel model we will use as reference during our Matlab computations.

WINNER is a European project, from the Sixth Framework Program effort [WIN]. Among its numerous contributions, it extended the 3GPP Spatial Channel Model (SCM) into a new model called SCME [Bau 05], [SCM 05].

The original SCM is a ray-based MIMO channel model, with a stochastic modeling of the scatterers. It is divided into several environment scenarios, both line-of-sight (LOS) and non-LOS. It models the received signal as 6 distinct delay paths (channel taps), and each one is the sum of 20 complex sinusoids. All the parameters (paths' power, delay and angle of arrival/departure) are modeled as random variables, depending on given probability density and cross-correlations functions.

The main extensions brought by SCME are new environment scenarios and bandwidth of 100 MHz in both 2 and 5 GHz carrying frequencies, instead of only 5 MHz at 2 GHz.

The SCME channel model has been implemented in Matlab. Given the environment scenario and the antenna array configurations, it generates channel matrices, then multidimensional arrays containing a specified number (100 per default) channel impulse response samples for all links.

Here are the default parameters of SCME:

- 2 Tx, 2 Rx, 6 paths per Tx-Rx pair (so 2 x 2 MIMO channel, with 6 taps);
- Urban Micro scenario;
- Mobile user velocity: 10 m/s ;
- Sampling : 2 samples per half wavelength,

  so $\Delta t = 3.75 \text{ ms} = 7.5 \text{ TTI} \quad (\text{with LTE TTI} = 0.5\,\text{ms})$;
- 100 time samples are computed;
- Center frequency: 2 GHz.

The resulting multidimensional array has therefore 2 x 2 x 6 x 100 dimensions.

For the purpose of the present work, we will feed our predictors with the channel impulse response samples produced by SCME.

To match other workpackages requirements, the WINNER's SCME has been tuned. The next section presents the modifications we made on the channel model

3 THE SURFACE CHANNEL MODEL

This section presents the SURFACE channel model, which is a wrapper to 3GPP Spatial Channel Model Extended (SCME) implementation provided by FP7 IST WINNER project v1.0 (May 30, 2005) [SCM 05]. Either existing SCME parameters or a few new ones are set in accordance with FP7 IST SURFACE D7.3 [SD 7.3].

Two main scenarios have been implemented, namely Micro-cell and Pico-cell scenarios, both for urban area. The Micro-cell scenario is derived from the 3GPP Micro cell scenario [3GPP 07], and proposes the choice between Outdoor to Outdoor and Outdoor to Indoor alternatives. The Pico-cell scenario is derived from the B1 scenario of the WINNER II IST project [WIN 6.13.7] and is only for outdoor-to-outdoor, with both LOS and NLOS coverage.

Two modeling levels are proposed. The model can be run at link level, and perform simulations at PRB scale, or it can be run at system level where the whole channel bandwidth is considered and no pathloss nor shadowing is applied.

3.1 Input/Output parameters

The SURFACE channel model has been implemented as a MATLAB function with 5 input parameters, and 3 output parameters.

The input parameters are

1. The computation level : “link” or “system”;
2. The scenario : “micro_o2o”, “micro_o2i”, “pico_los” or pico_nlos”;
3. The terminal velocity, in km/h;
4. The number of transmit antennas;
5. The number of receive antennas;

The outcomes of the model depend on the level. Please note that there are three output variables, whether this is a link- or a system-level simulation. However, the meaning of these output variables differs.

For the link level, the output parameters are:

1. The MIMO channel matrix (channel response), where n_T is the number of Tx, n_R is the number of Rx, n_{Sub} is one (channel is computed for the full PRB, and not for each of its subcarriers), and finally n_{Symb} is the number of symbols of a PRB;
2. C is a flag set up at 1 because we emulate one channel matrix per PRB;
3. T is a flag set up at 0 (no time compression).

For the system level, the output parameters are:

1. The MIMO channel matrix, with size $n_T \text{ x } n_R \text{ x } n_{Taps} \text{ x } n_{Samples}$, where n_T is the number of Tx, n_R is the number of Rx, n_{Taps} is the number of taps (6 actually), and finally $n_{Samples}$ is the number time samples;
2. The channel covariance matrix;
3. A structure containing the delays and the powers of each tap, and some other parameters.

3.2 Remarks about the model

For link-level simulations, setting SCME parameters for typical PRB bandwidths leads to a non frequency selective channel. This is why subcarrier scale is useless, and we model the channel at PRB scale.

Still for link-level simulations, the carrier frequency is hard coded to the one of a PRB chosen randomly in the bandwidth of the channel. In Micro-Cell scenarios, there is a total of 50 PRBs in a 10 MHz bandwidth, whereas in Pico-Cell scenarios, there are 60 PRBs in a 20 MHz bandwidth.

For system-level simulations, SCME is hard coded to provide the channel model of a single link (one MS, one BS, whatever the number of antennas). It could however deliver such models for several links simultaneously.

WINNER B4 model is currently not supported in the Pico-Cell NLOS indoor MS scenario.

3.3 Pathloss assumptions

For Pico-cell scenario, the pathloss has been adapted, because it requires some parameters SCME does not have. According to [SD 7.3, Table 7-1], for LOS case, pathloss should be

$$PL_{LOS} = 22.7\log_{10}(d_1[m]) + 41 + 20\log_{10}(f[GHz]/5), \quad d_1 < R_{bp}$$

$$PL_{LOS} = 40\log_{10}(d_1[m]) + 41 - 17.3\log_{10}(R_{bp}) + 20\log_{10}(f[GHz]/5), \quad d_1 \ge R_{bp}$$

$$\text{with } R_{bp} = 4\frac{(h_{BS}-1)(h_{MS}-1)}{\lambda} \text{ and } d_1 = 10m...5km$$

and for NLOS case, pathloss should be

$$PL_{NLLOS} = PL_{LOS}(d_1[m]) + 20 - 12.5n_j + 10n_j\log_{10}(d_2[m])$$

$$\text{with } n_j = \max(2.8 - 0.0024 * d_1[m],\ 1.84) \text{ and } d_2 = w/2...2km,\ w = 15m$$

where d_1 and d_2 are defined as on Fig 3-1.

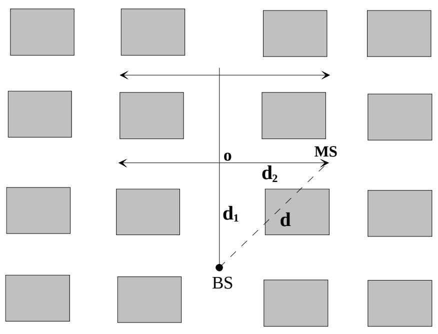


Figure 3-1 : Urban micro environment, position of BS and MS

Actually, WINNER SCME only provides us with distance *d*, and no angles to derive distances d_1 and d_2. So we imposed the following scheme to estimate the desired distances d_1 and d_2, where angle θ worth 45° in NLOS case, or 6.4° in LOS case (if *d*=100m, θ = 6.4° implies d_1= 90m and d_2=10m) :

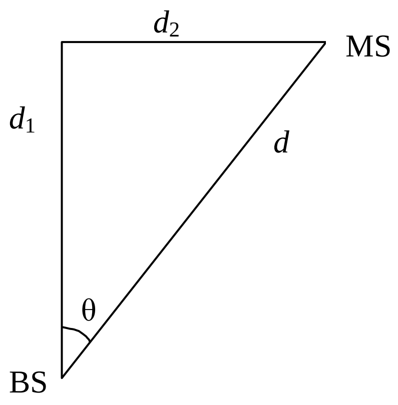


Figure 3-2 : Urban micro environment, assessment on position of BS and MS

The micro-cell pathloss has also been changed from the SCME settings, to match LTE pathloss assumption, according to [SD 7.3, Table 4-1], but it only depends on distance *d*, which value is known by SCME. So the LTE pathloss did not require any special assumption, unlike the WINNER II B1 pathloss of pico-cell scenarios.

The next two subsections details the parameters of the channel model, for each scenario. Then we will move to channel prediction and review a limited set of channel models and prediction strategies that revealed relevant for our prediction work.

3.4 Link level parameters

Parameters		Micro-Cell		Pico-Cell		Source
		Outdoor-to-outdoor	**Outdoor-to-indoor**	**LOS**	**NLOS**	
User-defined input parameters	Level	'link'				-
	Scenario	'micro_o2o'	'micro_o2i'	'pico_los'	'pico_nlos'	-
	Velocity [km/h]	3/30				Internal SURFACE requirements following October 28, 2007 conference call
	n_T	User-defined				
	n_R	User-defined				
Output variables	Channel					
	C	1	1	1	1	
	T	0	0	0	0	
Internal SCME parameters	Carrier frequency [GHz]	Depends on simulated PRB, within [1.75 ; 2.25]		Depends on simulated PRB, within [4.6 ; 5.4]		TS 36.211 v8.0.0
	MS-BS distance	Random				Original SCME
	MS height [m]	1.5				Original SCME
	BS height [m]	15		10		D7.3, Table 4-3
	Number of paths in TDL	1				SCME flat over a PRB
	Delay sampling [µs]	5.67 (= 1/180 kHz)		3.2 (= 1/312.5 kHz)		-
	Number of time samples	14 symbols or 1 ms		12 symbols or 0.3456 ms		D7.3, Table 2-1
	Time sampling [µs]	71.42		28.8		D7.3, Table 2-2
	WINNER scenario	B1				
	Pathloss					D7.3, Tables 4-1 and 4-2
	Shadowing					

3.5 System level parameters

Parameters		Micro-Cell		Pico-Cell		Source
		Outdoor-to-outdoor	Outdoor-to-indoor	LOS	NLOS	
User-defined input parameters	Level	'system'				-
	Scenario	'micro_o2o'	'micro_o2i'	'pico_los'	'pico_nlos'	-
	Velocity [km/h]	3				Internal SURFACE requirements following October 28, 2007 conference call
	n_T	User-defined				
	n_R	User-defined				
Output variables	Channel					
	C	Covariance matrix				
	T	SCME's FullOutput				
Internal SCME parameters	Carrier frequency [GHz]	2		5		
	MS-BS distance	Random				
	MS height [m]	1.5				
	BS height [m]	15		10		D7.3, Table 4-3
	Number of paths in TDL	6				
	Delay sampling [μs]	100		50		
	Number of time samples	100				
	Time sampling [ms]	135		54		
	WINNER scenario	B1				
	Pathloss	No pathloss from SCME (PathLossModelUsed = 'no')				
	Shadowing	No shadowing from SCME (ShadowingModelUsed = 'no')				

4 MODELISATION AND PREDICTION - STATE OF ART

In this section, we will present some channel models and estimators we investigated. There are many more existing channel models, but we are limiting our scope to the ones that have been showed to perform predictions. We will first review the ROMANTIK parametric model and a polynomial estimator. Then we will go through a MISO estimator, and we will end this section with an ESPRIT-based technique, which seems to fit well into our approach.

Before introducing these models, we will say a word about Kalman filters. This kind of filters would be very useful to help our models and predictors to deal with noisy situations. Indeed, Kalman filters could perform smoothing to remove the effects of the noise and provide a better estimate of the channel state. But the main drawback of these filters is that they need significant computational power. We are rather aiming at designing a predictor which can be run on typical energy- and CPU-limited devices. In a word, we are looking for the best trade-off between prediction performance and computer requirements. So we will not discuss Kalman filters further on.

4.1 The ROMANTIK model

ROMANTIK is an FP5 IST project. Its name means **R**es**O**urce **M**anagement and **A**dva**N**ced **T**ransceiver algor**I**thms for multihop networ**K**s. As part of its contributions, it proposed a channel model. The ROMANTIK model was initially a SISO parametric model [Bar 03a]. It has later been generalized into a MIMO model [Bar 03b]. These models also perform prediction of the channel state. It is worthwhile mentioning here that the ongoing FP6 IST project URANUS[1] follows up ROMANTIK activities, in working towards a parametric model of channel estimation.

4.1.1 The SISO model

Here is the SISO channel model, from [Bar 03a], where the channel *h* depends on time t and delay τ

$$h(t,\tau) = \sum_{k=1}^{N_s+1} a_k \; e^{j\,\phi_k\,t} \; \delta(\tau - \tau_{k(t)})$$

This model contains $3(N_s+1)$ parameters: $a_k,\ \tau_k,\ \phi_k \quad \forall k$. The a_k are the amplitudes, the τ_k are the delays and the ϕ_k are the Doppler frequency shifts of each path of the channel.

Actually, it's a (N_s+1)-order model; a sum of (N_s+1) weighted complex sinusoids. To estimate these parameters, sounding is suggested in [Bar 03a].

Hence for the prediction, the scheme is first to estimate the $3(N_s+1)$ parameters of the channel, then to run the model with the estimated parameters to predict future values. Because these parameters are time-varying, predictions will remain reliable only for a limited period of time. The prediction at time $t+\Delta$ hence writes:

$$\hat{h}(t+\Delta,\tau) = \sum_{k=1}^{N_s+1} a_k \; e^{j\,\phi_k\,(t+\Delta)} \; \delta(\tau - \tau_{k(t+\Delta)})$$

[1] Universal RAdio-liNk platform for efficient USer centric access

4.1.2 The MIMO model

ROMANTIK further proposed a MIMO channel model ([Bar 03b]), considering that a $n_T \times n_R$ MIMO channel can be regarded as $n_T \cdot n_R$ spatially correlated SISO channels.

The exploitation of the spatial correlation lead ROMANTIK to assume that the delays τ_k and the Doppler shifts ϕ_k are identical for all the $n_T \cdot n_R$ SISO components of a given MIMO channel. This assumption is motivated by the fact that antennas are relatively close with respect to the distance between the transmitter and the receiver, such that the variation of delay or Doppler shift from an antenna pair to another can be regarded as negligible.

Indeed, the delays depend on the path lengths, and in front of the distance between the transmitter and the receiver, the few centimeters between transmit/receive antennas are quite negligible. The same applies to the Doppler shifts. They depend on the angle between the movement of the receiver and the direction of the waves impinging from the scatterers. If the scatterers are not too close, the variations of angle between receiving antennas are negligible too. However, despite delays and Doppler shifts are identical from one spatial link to the other, those links can appear uncorrelated due to the difference between their amplitudes.

In the ROMANTIK MIMO channel model, the communication link between the *m*th transmitting antenna and the *n*th receiving one is given by:

$$h^{mn}(t,\tau) = \sum_{k=1}^{N_s+1} a_k^{\mathrm{mn}} \; e^{j\,\phi_k\, t} \;\; \delta(\tau - \tau_{k(t)})$$

The prediction scheme is the same than in the SISO case: first estimate the parameters of the channel model by sounding, then run it to compute future values. For a given link, the MIMO prediction at time $t+\Delta$ writes:

$$\hat{h}^{mn}(t+\Delta,\tau) = \sum_{k=1}^{N_s+1} a_k^{\mathrm{mn}} \; e^{j\,\phi_k\,(t+\Delta)} \;\; \delta(\tau - \tau_{k(t+\Delta)})$$

4.1.3 Comments

As we can see, ROMANTIK relies on a parametric model. From a feedback point of view, a parametric model has the advantage to be fully defined with a reduced set of parameters. Indeed, it’s easier to send a small amount of parameters than a lot of CSI. On the other hand, the predictions will remain reliable as long as the parameters remain constant.

The core of this parametric model is actually to estimate these parameters. To perform it, ROMANTIK suggest sounding, and documents [Bar 03a] and [Bar 03b] describe in details several sounding schemes.

As we do not know whether SURFACE will afford sounding, we considered other options. In the next section, we will review a simpler predictor, because it is polynomial.

4.2 A Polynomial Estimator

4.2.1 Proposed scheme

This section is based on [Che 03]. The authors propose first to compute a polynomial model which fits a set of channel observations, then to run this model in order to predict future channel observations. In fact, the model is split into two separate models, one for the real part of the channel state, and the other for its imaginary part.

The polynomial, discrete time channel model, including an initial phase φ_k, writes:

$$h(n) = \sum_{k=1}^{N_s+1} a_k\, e^{j(2\pi\phi_k n + \varphi_k)} = I(n) + j.Q(n)$$

We have to compute the two polynomials *I*(*n*) and *Q*(n) such that

$$I(n) = \sum_{k=0}^{D-1} c_k \cdot n^k = \Re\big(h(n)\big) \equiv \text{Real part of the channel state}$$

$$Q(n) = \sum_{k=0}^{D-1} d_k \cdot n^k = \Im\big(h(n)\big) \equiv \text{Imaginary part of the channel state}$$

where *D* is the degree of the polynomial. Therefore, to derive the c_k and the d_k, we have to solve these linear systems:

$$\begin{pmatrix} 1 & D & \cdots & D^{D-1} \\ 1 & D-1 & \dots & (D-1)^{D-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{D-1} \end{pmatrix} = \begin{pmatrix} \Re\big(h(n)\big) \\ \Re\big(h(n-1)\big) \\ \vdots \\ \Re\big(h(n-D+1)\big) \end{pmatrix};$$

$$\begin{pmatrix} 1 & D & \cdots & D^{D-1} \\ 1 & D-1 & \dots & (D-1)^{D-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{pmatrix} \begin{pmatrix} d_0 \\ d_1 \\ \vdots \\ d_{D-1} \end{pmatrix} = \begin{pmatrix} \Im\big(h(n)\big) \\ \Im\big(h(n-1)\big) \\ \vdots \\ \Im\big(h(n-D+1)\big) \end{pmatrix}.$$

More details about mathematical justification of the algorithm is given in [Che 03].

The predictions, *l* time instants later, are then given by:

$$\hat{I}(n+l) = \sum_{k=0}^{D-1} c_k\,(D+l)^k \quad \text{and} \quad \hat{Q}(n+l) = \sum_{k=0}^{D-1} d_k\,(D+l)^k \,.$$

$$\Rightarrow \quad \hat{h}(n+l) = \sum_{k=0}^{D-1} (c_k + \mathrm{i}\, d_k)\cdot(D+l)^k$$

4.2.2 Comments

This polynomial channel model is very simple, with only two small matrices to invert, which additionally are always the same. Their inverse could therefore be precomputed, which would ease the process.

The model is also very simple because it does not need a lot of observations. We can aim at using polynomials with degree 3, 5 or at least 10. It means that we do not need more than $E = 3$, 5 or 10 observations to produce predictions, where E is the number of channel observations the predictor needs.

However, there are two main drawbacks: predictions are only reliable on very short term and this is a SISO model.

4.2.3 Results of computations

We have implemented this predictor in a Matlab script, and tested it with WINNER's SCME. SCME was used with its default values listed in Section 2. **Figure 4-1** shows two representative runs of the predictor (real part only). The left subfigure is obtained with a polynomial of degree 2, and the right one with a polynomial of degree 7. In the first case, the warm-up, which is the initial period until E channel observations have been collected, is only 3 observations. It is 8 in the second case. During the warm-up period, no prediction is yet available.

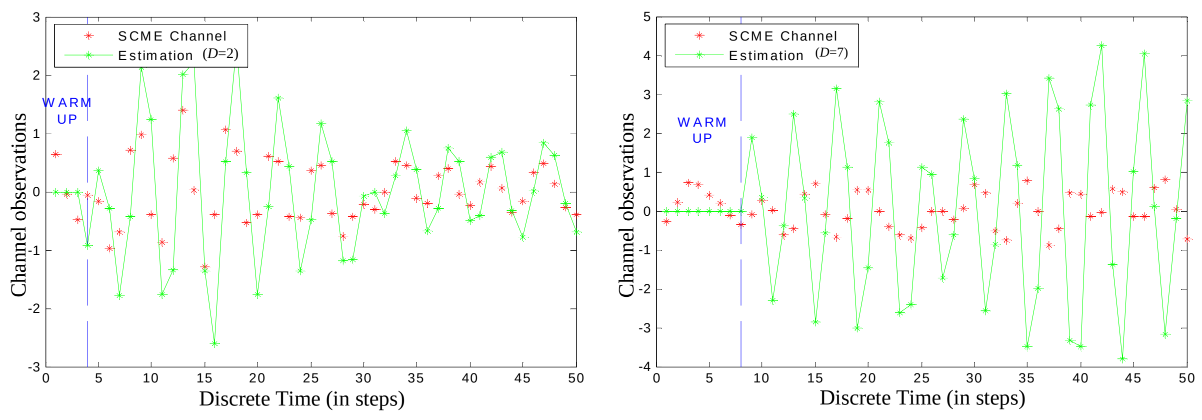


Figure 4-1 : Predictions of the polynomial estimator

These predictions are not reliable with $D = 2$, and its gets worse when increasing the degree of the polynomial.

This model is actually too simple and it can not perform well with a too low sampling rate as the one used in **Figure 4-1**. Therefore, it cannot predict the oscillations of the channel state, and its predictions often fall much below or above the value it has to predict, resulting in a significant prediction error.

We will now review a MISO[1] estimator, which represents the channel as a sum of weighted sinusoids. It is not as simple as the polynomial estimator is. Hence, it should be more accurate.

[1] Multiple Input Single Output.

4.3 A MISO estimator

This subsection is based on the predictor described in [Arr 02]. This predictor is meant to be used with smart antenna base stations, e.g. several (n_T) transmitting antennas, and only one receiving antenna. This is a MISO channel model. This is the model, with the *l* index standing for the transmitting antenna this ‘‘sub-channel’’ is related with:

$$h^l(t) = \sum_{k=1}^{N_s+1} a_k{}^l \, e^{j\,t\,\phi_k}$$

As mentioned above, the channel model is a weighted sum of complex sinusoids. With the help of mathematical properties detailed in [Arr 02], it can be expressed as a weighted sum of itself at past instants:

$$h^l(t) = -\sum_{k=1}^{N_s+1} b_k \, h^l(t-k)$$

such that we just have to compute the b_k, to be able to predict future channel observations from a linear combination of past values. To derive the b_k, one has to solve a huge linear system, with the pseudo inverse of a large matrix to compute. More details about this predictor and the way to compute the b_k coefficients are to be found in [Arr 02].

To exploit the spatial correlation between the n_T channels, the authors of [Arr 02] suggest to use the same b_k coefficients for each transmitting antenna.

To draw a conclusion about this method, we can say that it seems complex. It deals with larger matrices, which contain future measurements. The non causality of the method and the heavy computations motivated our choice not to test it.

We will now shortly present a channel model which seems us to be really promising. The next two sections will dig deeper into this SISO channel model and the MIMO model we have derived from it.

4.4 The Andersen et al. predictor

This last subsection is based on the paper [And 99]. Its authors model the channel as follows:

$$h(t) = \sum_{k=1}^{N_s+1} a_k \, e^{j\,t\,\phi_k} = \sum_{k=1}^{N_s+1} a_k \, z_k{}^t \,, \qquad z_k = e^{j\,\phi_k} \;\; \forall k.$$

It means that the channel impulse response $h(t)$ is considered as a weighted sum of (N_s+1) complex sinusoids. The a_k are the weighting coefficients and the ϕ_k are the spatial Doppler frequencies. So, the main goal of this method is to find the z_k, called signal-poles, and the a_k (amplitudes).

Actually, this model can be seen as a modified version of the ESPRIT technique[1]. We performed Matlab computations with this channel model and it seems to work great with WINNER's SCME. So we have investigated it deeper,

The next two sections provide more details about the working scheme of this ESPRIT-based technique. In Section 6, we will review deeply the Andersen et al. predictor, but the next section will first review the original ESPRIT technique.

[1] Estimation of Signal Parameters by Rotational Invariance Techniques

5 THE ESPRIT PREDICTOR

This section will detail the working scheme of the ESPRIT method [Sto 05]. We will discuss about the channel model and its parameters, then we will give its mathematical background, and we will conclude this section with a summary algorithm.

5.1 Channel Model

This is the basic channel model we will use, depending on distance *x*:

$$h(x) = \sum_{k=1}^{N_s+1} a_k\, e^{i\, k_\lambda\, x\, cos(\theta_k)},$$

where N_s is the number of scatterers, a_k is the amplitude, θ_k is the angle between the motion of the receiver and the direction of arrival of the received signal, k_λ is the wave number, depending on the wavelength ($k_\lambda = 2\pi/\lambda$).

To reduce the problem of modeling the channel state to a classical frequency estimation problem, we will assume that velocity is constant (so constant sampling in time is constant sampling in space) and that amplitude variations may be neglected. We will also transform the model into a discrete time model.

So, we have $e^{k_\lambda\, \Delta x\ cos(\theta_j)}$, where Δx is the move between two discrete time steps. Actually, $k_\lambda\, \Delta x\ cos(\theta_k)$ is the spatial Doppler shift. We will write it ϕ_k. We will also write $z_k = e^{i\,\phi_k}$. Therefore, $h(n) = \sum_{k=1}^{N_s+1} a_k\, e^{i\,n\,\phi_k} = \sum_{k=1}^{N_s+1} a_k\, z_k{}^n$.

For our predictions, we have to find the $z_k = e^{i\,\phi_k}$ (so-called signal poles) and the a_k (amplitudes). We will now discuss about a way to derive these parameters.

5.2 Basic idea

At first, we consider $\mathcal{A}$, a E x N_s Vandermonde matrix such as

$$\mathcal{A} = \begin{pmatrix} 1 & \cdots & 1 \\ e^{i\phi_1} & \cdots & e^{i\phi_{N_s}} \\ e^{2i\phi_1} & & e^{2i\phi_{N_s}} \\ \vdots & \ddots & \vdots \\ e^{(E-1)i\phi_1} & \cdots & e^{(E-1)i\phi_{N_s}} \end{pmatrix}.$$

Actually, $\mathcal{A}$ is linked to the channel matrix, because the $e^{i\phi_j}$ are the poles of the channel.

Be $J_{UP} = \left(I_{E-1}\,,(0,\ldots 0)^T\right)$ and $J_{DOWN} = \left((0,\ldots 0)^T\,, I_{E-1}\right)$, where I_{k-1} is the square *(E-1)* x *(E-1)* identity matrix. So J_{UP} and J_{DOWN} are *(E-1)* x E matrices such that

$$J_{UP}\cdot\mathcal{A} = \begin{pmatrix} 1 & \cdots & 1 \\ e^{i\phi_1} & \cdots & e^{i\phi_{N_s}} \\ \vdots & \ddots & \vdots \\ e^{(E-2)i\phi_1} & \cdots & e^{(E-2)i\phi_{N_s}} \end{pmatrix} \overset{\Delta}{=} \mathcal{A}_{UP}\,, \quad \text{and}$$

$$J_{DOWN}\cdot\mathcal{A} = \begin{pmatrix} e^{i\phi_1} & \cdots & e^{i\phi_{N_s}} \\ e^{2i\phi_1} & & e^{2i\phi_{N_s}} \\ \vdots & \ddots & \vdots \\ e^{(E-1)i\phi_1} & \cdots & e^{(E-)i\phi_{N_s}} \end{pmatrix} \overset{\Delta}{=} \mathcal{A}_{DOWN}\,.$$

Note that these matrices are *(E-1)* x N_s. Actually, $\mathcal{A}_{UP}$ is the $\mathcal{A}$ matrix without its last row, and $\mathcal{A}_{DOWN}$ is $\mathcal{A}$ without its first row.

It's easy to see that $\mathcal{A}_{UP}\cdot\Phi = \mathcal{A}_{DOWN}$ means that

$$\Phi = \begin{pmatrix} e^{i\phi_1} & & & 0 \\ & e^{i\phi_2} & & \\ & & \ddots & \\ 0 & & & e^{i\phi_{N_s}} \end{pmatrix}$$

5.3 The ESPRIT technique (mathematical background)

We will consider now the vector $y(n) = h(n)\cdot x(n)+w(n)$ which is the received signal from the channel to study; where x is the information to transmit, and w is the noise (n is discrete time).

Actually, because $h(n) = \sum_{k=1}^{N_s+1} a_k\, z_k{}^n$, we have this matrix equation :

$$h(n, n-1,\ldots, n-E) = \mathcal{A}\cdot a\,.$$

So $y(n) = \mathcal{A}\cdot a\cdot x(n)+w(n)$.

Be R the covariance matrix of the received signal, the * denotes the conjugate transpose of the matrix.

$$R = E\left\{y(n)\cdot y(n)^\star\right\} = E\left\{\mathcal{A}\cdot a\cdot x(n)\cdot x(n)^\star\cdot a^\star\cdot\mathcal{A}^\star\right\} + E\left\{w(n)\cdot w(n)^\star\right\}$$

$$\Rightarrow\; R = \mathcal{A}\cdot P\cdot\mathcal{A}^\star + \sigma^2 I\,;\quad where\; P \overset{\Delta}{=} E\left\{a\cdot x(n)\cdot x(n)^\star\cdot a^\star\right\}$$

Please pay attention that we need the hypothesis that the noise is a white noise to write:

$$E\left\{w(n)\cdot w(n)^\star\right\} = \sigma^2\, I$$

Notice too that we have to be in discrete time to compute our predictors, so in the matrices *R*, $\mathcal{A}$, *P*, there are discrete time values.

We will write l = rank($\mathcal{A}\cdot P\cdot \mathcal{A}^\star$). In fact, $l = N_s + 1$, the number of paths.

Actually, $\mathcal{A}\cdot P\cdot \mathcal{A}^\star$ has $N_s + 1$ positives eigenvalues, and $k - (N_s + 1)$ zero eigenvalues. They are linked to the eigenvalues of R.

If we write $\tilde{\lambda}_j$ for the eigenvalues of $\mathcal{A}\cdot P\cdot \mathcal{A}^\star$, and λ_j for those of R, we have that

$$\lambda_j = \begin{cases} \tilde{\lambda}_j + \sigma^2 & \forall\ j = 1, \ldots, N_s + 1 \\ \sigma^2 & \forall\ j = N_s + 2, \ldots, E \end{cases}$$

Now let's make a Singular Value Decomposition (SVD) of the R matrix (a square E x E matrix).

$$R = (S \quad G)\cdot \begin{pmatrix} \lambda_1 & & & & & 0 \\ & \ddots & & & & \\ & & \lambda_{N_s+1} & & & \\ & & & \lambda_{N_s+2} & & \\ & & & & \ddots & \\ 0 & & & & & \lambda_E \end{pmatrix} \cdot \begin{pmatrix} S^\star \\ G^\star \end{pmatrix}$$

where S contains the eigenvectors of the $N_s + 1$ greatest eigenvalues, and G contains the remaining eigenvectors.

So
$$R = S\cdot \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N_s+1} \end{pmatrix} \cdot S^\star + G\cdot \begin{pmatrix} \lambda_{N_s+1}2 & & 0 \\ & \ddots & \\ 0 & & \lambda_E \end{pmatrix} \cdot G^\star .$$

Let's calculate $R\cdot S$:

$$R\cdot S = S\cdot \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N_s+1} \end{pmatrix} \cdot \underbrace{S^\star\cdot S}_{=I} + G\cdot \begin{pmatrix} \lambda_{N_s+2} & & 0 \\ & \ddots & \\ 0 & & \lambda_E \end{pmatrix} \cdot \underbrace{G^\star\cdot S}_{=0} = S\cdot \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N_s+1} \end{pmatrix} ;$$

(It's because of the SVD: the eigenvectors are orthonormalised, and then $S \perp G$.)

$$\Rightarrow R\cdot S = S\cdot \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N_s+1} \end{pmatrix}$$

But because R also equals $\mathcal{A}\cdot P\cdot \mathcal{A}^\star + \sigma^2 I$, $R\cdot S$ also equals $\mathcal{A}\cdot P\cdot \mathcal{A}^\star\cdot S + \sigma^2 S$!

$$\Rightarrow R\cdot S = \mathcal{A}\cdot P\cdot \mathcal{A}^\star\cdot S + \sigma^2 S$$

Therefore,
$$\mathcal{A}\cdot P\cdot \mathcal{A}^\star\cdot S + \sigma^2 S = S\cdot \begin{pmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N_s+1} \end{pmatrix} ;$$

And then

$$\mathcal{A} \cdot P \cdot \mathcal{A}^{\star} \cdot S \;=\; S \cdot \begin{pmatrix} \lambda_1 - \sigma^2 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N_s+1} - \sigma^2 \end{pmatrix} \;\overset{\Delta}{=}\; S \cdot \Lambda \;.$$

So, $$S \;=\; \mathcal{A} \cdot \underbrace{P \cdot \mathcal{A}^{\star} \cdot S \cdot \Lambda^{-1}}_{\overset{\Delta}{=}\, C} \;=\; \mathcal{A} \cdot C$$

$$\Rightarrow\; S \;=\; \mathcal{A} \cdot C \text{ and } \mathcal{A} \;=\; S \cdot C^{-1}.$$

Now remember that we proved that $J_{UP} \cdot \mathcal{A} \cdot \Phi = J_{DOWN} \cdot \mathcal{A}$.

So, because $\mathcal{A} \;=\; S \cdot C^{-1}$,

$$J_{UP} \cdot S \cdot C^{-1} \cdot \Phi \;=\; J_{UP} \cdot \mathcal{A} \cdot \Phi \;=\; J_{DOWN} \cdot \mathcal{A} \;=\; J_{DOWN} \cdot S \cdot C^{-1} \;;$$

And then $J_{UP} \cdot S \cdot \underbrace{C^{-1} \cdot \Phi \cdot C}_{\overset{\Delta}{=}\,\mathcal{D}} \;=\; J_{DOWN} \cdot S$

Finally we have that $S_{UP} \cdot \mathcal{D} \;=\; S_{DOWN}$, where

$S_{UP} \;=\; J_{UP}\; S \;\equiv$ (E-1) first rows of S, and

$S_{DOWN} \;=\; J_{DOWN}\; S \;\equiv$ (E-1) last rows of S.

Because $\mathcal{D} \;=\; C^{-1} \cdot \Phi \cdot C$, and C is nonsingular, $\mathcal{D}$ and Φ share the same eigenvalues. They are related by a similarity transformation. Their eigenvalues are actually the diagonal elements of Φ, i.e. $e^{i\,\Phi_1}, e^{i\,\Phi_2}, \ldots, e^{i\,\Phi_{N_s}}$, our signal poles. Therefore, to find the $z_k = e^{i\,\phi_k}$, we have to compute the SVD of R to derive S. From S we can get the $\mathcal{D}$ matrix, whose eigenvalues are the desired signal poles $e^{i\,\Phi_1}, e^{i\,\Phi_2}, \ldots, e^{i\,\Phi_{N_s+1}}$.

5.4 How to compute the correlation matrix *R*?

First, we have to fill $\mathcal{H}$, a Hankel matrix with the received signal. Please note that we will need $E = m + n - 1$ observations to fill the matrix.

$$\mathcal{H} = \begin{pmatrix} h(1) & h(2) & h(3) & \cdots & h(m) \\ h(2) & h(3) & h(4) & \cdots & h(m+1) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h(n) & h(n+1) & h(n+2) & \cdots & h(m+n-1) \end{pmatrix}$$

Then, we can get an estimation of *R* by computing $\frac{\mathcal{H} \cdot \mathcal{H}^{\star}}{m}$.

Indeed,

$$\frac{\mathcal{H} \cdot \mathcal{H}^{\star}}{m} = \frac{1}{m} \begin{pmatrix} h(1) & h(2) & \cdots & h(m) \\ \vdots & \vdots & \ddots & \cdots \\ h(n) & h(n+1) & \cdots & h(n+m-1) \end{pmatrix} \cdot \begin{pmatrix} \bar{h}(1) & . & \bar{h}(n) \\ \bar{h}(2) & . & \bar{h}(n+1) \\ \vdots & . & \vdots \\ \bar{h}(m) & . & \bar{h}(n+m-1) \end{pmatrix}$$

$$= \frac{1}{m} \begin{pmatrix} h(1)\bar{h}(1) + h(2)\bar{h}(2) + \cdots + h(m)\bar{h}(m) & \cdots & h(1)\bar{h}(n) + h(2)\bar{h}(n+1) + \cdots + h(m)\bar{h}(n+m-1) \\ \vdots & \ddots & \vdots \\ h(n)\bar{h}(1) + h(n+1)\bar{h}(2) + \cdots + h(n+m-1)\bar{h}(m) & \cdots & h(n)\bar{h}(n) + h(n+1)\bar{h}(n+1) + \cdots + h(n+m-1)\bar{h}(n+m-1) \end{pmatrix}$$

$$\Rightarrow \frac{\mathcal{H} \cdot \mathcal{H}^{\star}}{m} = \begin{pmatrix} \frac{\sum_{k=1}^{m} h(k)\,\bar{h}(k)}{m} & \cdots & \frac{\sum_{k=1}^{m} h(k)\,\bar{h}(n+k-1)}{m} \\ \vdots & \ddots & \vdots \\ \frac{\sum_{k=1}^{m} h(n+k-1)\,\bar{h}(k)}{m} & \cdots & \frac{\sum_{k=1}^{m} h(n+k-1)\,\bar{h}(n+k-1)}{m} \end{pmatrix}$$

$$\overset{\Delta}{=} \begin{pmatrix} \rho_0 & \rho_1 & \rho_2 & \cdots & \rho_{n-1} \\ \rho_{-1} & \rho_0 & \rho_1 & \ddots & \vdots \\ \rho_{-2} & \rho_{-1} & \rho_0 & \ddots & \rho_2 \\ \vdots & \ddots & \ddots & \ddots & \rho_1 \\ \rho_{-(n+1)} & \cdots & \rho_{-2} & \rho_{-1} & \rho_0 \end{pmatrix}$$

where ρ_k is the time correlation between channel observations $\{h(n)\}$ and $\{h(n+k)\}$.

As a result, we can get an estimation of *R*, the correlation matrix of the received signal, by computing:

$$\hat{R} \approx \frac{\mathcal{H} \cdot \mathcal{H}^{*}}{m}$$

The shape of the Hankel matrix, e.g. the ratio between the number of columns *m* and the number of rows *n*, has an influence on the performance of the predictor. Increase *m* improves the precision of the correlations ρ_s . Increase *n* enlarges the range of time correlations inside the correlation matrix *R*. For our computations, we chose $\mathcal{H}$ to be nearly square matrices, with one more row than columns ($n \geq m$).

5.5 Computing the amplitudes

Once we have found the signal poles, we can compute the amplitudes by solving this linear system:

$$\underbrace{\begin{pmatrix} h(1) \\ h(2) \\ \vdots \\ h(E) \end{pmatrix}}_{Y} = \underbrace{\begin{pmatrix} 1 & 1 & \cdots & 1 \\ z_1 & z_2 & \cdots & z_{N_s+1} \\ \vdots & \vdots & \ddots & \vdots \\ z_1^{E-1} & z_2^{E-1} & \cdots & z_{N_s+1}^{E-1} \end{pmatrix}}_{Z} \cdot \underbrace{\begin{pmatrix} a_1 \\ a_2 \\ \vdots \\ a_{N_s+1} \end{pmatrix}}_{a}$$

$$Y = z \cdot a \quad \Rightarrow \quad a = (Z^* Z)^{-1} \cdot Z^* \cdot Y$$

5.6 Producing predictions

Consider we have computed the signal poles and the associated amplitudes from the $E = n + m - 1$ channel observations *h*(1), *h*(2), … , *h*(*n* + *m*-1). It means that

$$h(1) = \sum_{k=1}^{N_s+1} a_k \; ; \quad h(2) = k \sum_{j=1}^{N_s+1} a_k \; z_k \; ; \quad h(3) = \sum_{k=1}^{N_s+1} a_k \; z_k^{\,2} \; ; \quad \ldots \quad h(E) = \sum_{k=1}^{N_s+1} a_k \; z_k^{\,E-1} \, .$$

In real life, the environment changes, and the channel impulse response changes accordingly. The channel ought not to vary too fast, unless the computed parameters (signal poles and amplitudes) would quickly become outdated within a certain time span.

Assuming those parameters to be valid within a given horizon, we just have to reuse the computed parameters for predictions as follows:

$$\forall n = E+1, E+2, \ldots \quad \hat{h}(n) = \sum_{k=1}^{N_s+1} a_k \; z_k^{\,n-1} \, .$$

5.7 Summary Algorithm

Here is the ESPRIT prediction algorithm.

1. Compute an estimate of the correlation matrix *R*, $\hat{R} \approx \frac{\mathcal{H} \cdot \mathcal{H}^*}{m}$, where $\mathcal{H}$ is a Hankel matrix containing the channel observations.
2. Compute the SVD of *R* $\left(\rightarrow R = U.\sum.V\right)$, and derive the *S* matrix, which comes from the first ($N_s + 1$) columns of *U*. ($N_s + 1$) is the number of singular values of *R* larger than the variance of the noise (assumed here to be known).
3. Find the $\mathcal{D}$ matrix by solving $S_{UP} \cdot \mathcal{D} = S_{DOWN}$.
4. The signal poles $\{z_k\}$ are the eigenvalues of $\mathcal{D}$.
5. Find the amplitudes from $Y = Z \cdot a \Rightarrow a = (Z^* Z)^{-1} \cdot Z^* \cdot Y$, where *Z* is a Vandermonde matrix filled with the poles, and *Y* is a vector containing the channel observations.

Finally, here are the predictions:

$$\forall n = E+1, E+2, \ldots \quad \hat{h}(n) = \sum_{k=1}^{N_s+1} a_k \, z_k^{\,n} .$$

In Section 7, we will discuss the robustness of this predictor, and we will also show results of Matlab computations. But before coming to this discussion, we will detail the modified ESPRIT algorithm of [And 99] briefly introduced in Section 4.4.

6 ANOTHER PREDICTOR BASED ON ESPRIT

In their paper [And 99], the authors introduced a variant of the typical ESPRIT algorithm we just reviewed. Their idea is to apply the ESPRIT analysis to a Hankel matrix which contains the channel observations.

6.1 Basic idea

First, we have to fill the n x m Hankel matrix $\mathcal{H}$, with $n + m = E + 1$, and $m \leq n$. E still stand for the number of observations collected to perform predictions.

$$\mathcal{H} = \begin{pmatrix} h(1) & h(2) & h(3) & \cdots & h(m) \\ h(2) & h(3) & h(4) & \cdots & h(m+1) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ h(n) & h(n+1) & h(n+2) & \cdots & h(m+n-1) \end{pmatrix}.$$

Let's compute a SVD of $\mathcal{H}$:

$$\mathcal{H} = U \cdot \Sigma \cdot V = \begin{pmatrix} U_1 & U_2 \end{pmatrix} \cdot \begin{pmatrix} \Sigma_1 & 0 \\ 0 & \Sigma_2 \end{pmatrix} \cdot \begin{pmatrix} V_1 & \star \\ V_2 & \star \end{pmatrix},$$

where U (n x m) and V (m x m) contain orthonormal columns, and Σ (m x m) is a diagonal matrix whose (diagonal) elements are sorted by value. U_1 contains the first $(N_s + 1)$ columns, where $(N_s + 1)$ is the dimension of the signal space.

Let us now compute a $\mathcal{D}$ matrix such as

$$J_{UP} \cdot U_1 \cdot \mathcal{D} \approx J_{DOWN} \cdot U_1$$

where the z_j are the eigenvalues of the $\mathcal{D}$ matrix. In [And 99], it is suggested to discard poles with an amplitude larger than 1.05. This threshold is set arbitrarily. Figure 6-1 shows an example of the set of poles we can identify. The ones lying beyond the red circle are discarded.

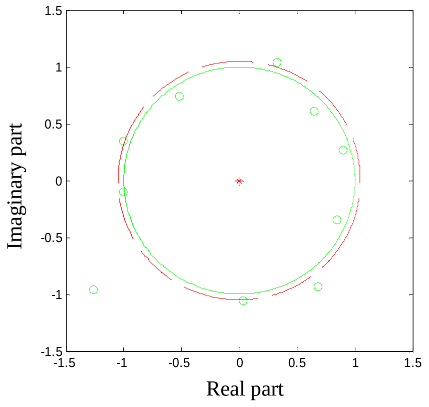


Figure 6-1 : A plot example of computed poles (green plots). The green circle is the unitary circle, and the red curve stands for the threshold

When the poles are known, we can compute the amplitudes, and build the predictor in a similar way to the standard ESPRIT algorithm:

$$a = (Z * Z)^{-1} \cdot Z * \cdot Y$$

where the predictions[1] are given by: $\forall n = E+1, E+2, \ldots \quad \hat{h}(n) = \sum_{k=1}^{N_s+1} a_k \, z_k{}^t n$.

6.2 Mathematical justification

Be $\mathcal{H}$ the Hankel matrix which is used it this method, and R the correlation matrix used by ESPRIT. So $R = \dfrac{\mathcal{H} \cdot \mathcal{H}^*}{m}$

Let us make singular value decompositions:

$$R \overset{SVD}{=} U_R \,.\, \Sigma_R \,.\, V_R{}^*$$

$$\mathcal{H} \overset{SVD}{=} U_H \,.\, \Sigma_H \,.\, V_H{}^*$$

$$\Rightarrow \mathcal{H}^* = V_H \,.\, \Sigma_H{}^* . U_H{}^*$$

Therefore, because $R = \dfrac{\mathcal{H} \cdot \mathcal{H}^*}{m}$, we have

$$R = \frac{1}{m} \cdot U_H \cdot \Sigma_H \cdot V_H{}^* \cdot V_H \cdot \Sigma_H{}^* \cdot U_H{}^*$$

Since V_H has been obtained from a SVD, V_H is a unitary matrix.

It means that $V_H \,.\, V_H{}^* = I = V_H{}^* . V_H$.

$$\Rightarrow \quad R = U_H \cdot \left(\frac{1}{m} \cdot \Sigma_H \cdot \Sigma_H{}^* \right) \cdot U_H{}^* .$$

Keep in mind that $R = U_R \,.\, \Sigma_R \,.\, V_R{}^*$,

where U_R is a E x E unitary matrix, like U_H;

where $V_R{}^*$ is a E x E unitary matrix, like $U_H{}^*$;

where Σ_R is a E x E diagonal matrix, like $\left(\frac{1}{m} \cdot \Sigma_H \cdot \Sigma_H{}^* \right)$ actually.

[1] Oddly, we discovered that if we take for U_1 the $(N_s + 1)$ first columns of the $\mathcal{H}$ matrix, we will get the same results as if we perform the SVD and use the U matrix.

So in the end, $R = U_H \cdot \left(\frac{1}{m} \cdot \Sigma_H \cdot \Sigma_H * \right) \cdot U_H *$ is an Eigen Value Decomposition (EVD) of *R*, and the singular vectors derived from the SVD of $\mathcal{H}$ are the eigenvectors of R .

Therefore, using the U_R matrix for the SVD of *R* or the U_H matrix from the SVD of $\mathcal{H}$ does not make a difference.

6.3 Differences between this algorithm and ESPRIT

ESPRIT can deal with noise, because the number of poles to identify is drawn from the number of singular values larger than σ^2, the variance of the noise (assumed to be known). The Andersen et al. predictor does not explicitly deal with the noise. The number of poles to identify remains always the same, whatever the noise level. Consequently, one may expect ESPRIT to be more robust against noise than the modified version of Andersen et al.

On the other hand, ESPRIT delivers normalized poles whereas the Andersen et al. predictor does not. The latter only suggests discarding poles with a too large modulus. Indeed poles with a modulus larger than unity will bring instability in the system, and the influence of poles smaller than unity fade along time.

6.4 Summary Algorithm

Here is the summary of this alternate predictor.

1. Compute the SVD of the $\mathcal{H}$ matrix (a Hankel matrix containing the channel observations).
2. Take the first $(N_s + 1)$ columns of the *U* matrix (U_1) to compute the $\mathcal{D}$ matrix: $J_{UP} \cdot U_1 \cdot \mathcal{D} \approx J_{DOWN} \cdot U_1$.
3. Therefore, the poles are the eigenvalues of $\mathcal{D}$ which are smaller than 1.05.
4. Then compute the amplitudes: $a = (Z * Z)^{-1} \cdot Z * \cdot Y$

Finally, the predictions are given by: $\forall n = E+1, E+2, \ldots \ \hat{h}(n) = \sum_{k=1}^{N_s+1} a_k \, z_k^{\,n}$.

In the last two sections, we have reviewed two ESPRIT-based predictors. We will now proceed to a section presenting the results of the Matlab computations we performed with these two algorithms.

7 SISO COMPUTATIONS

In this section, we will show the results of SISO computations, performed with Matlab. We have tested the ESPRIT predictor and the Andersen et al. one, and we have used the WINNER's SCME to provide us with a reference channel to predict. For all the results described in this section, SCME was used with its default values as listed in Section 2. Additionally, we used Matlab in its default resolution, which is 64 bits.

In the next subsection, we will show results obtained without noise, and compare the two methods. Then, we will show the effects of adding white Gaussian noise on the channel.

7.1 Sample computations

In this subsection, we will compare the two methods in noise free computations.

Figure 7-1, **Figure 7-2** and **Figure 7-3** show a few sample results obtained with *E* = 14 channel observations (warm-up period of 14 x 7.5 x 0.5 ms = 52.5 ms). These graphics show the real part of the values. Their imaginary parts look quite the same. On the left handside, one observes the actual channel observations and their predictions. On the right handside, the prediction error is illustrated.

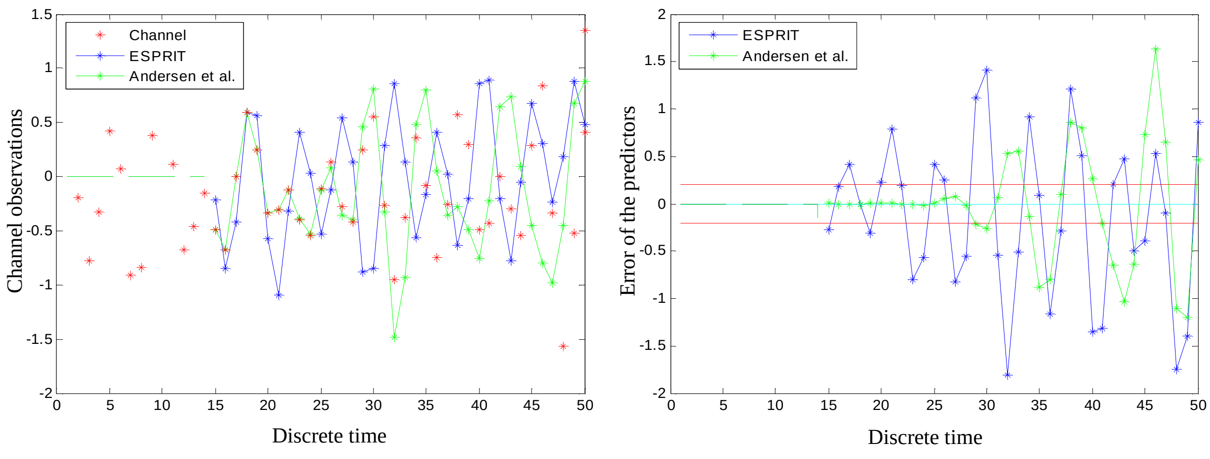


Figure 7-1 : A sample result of the SISO predictors (*E* = 14).

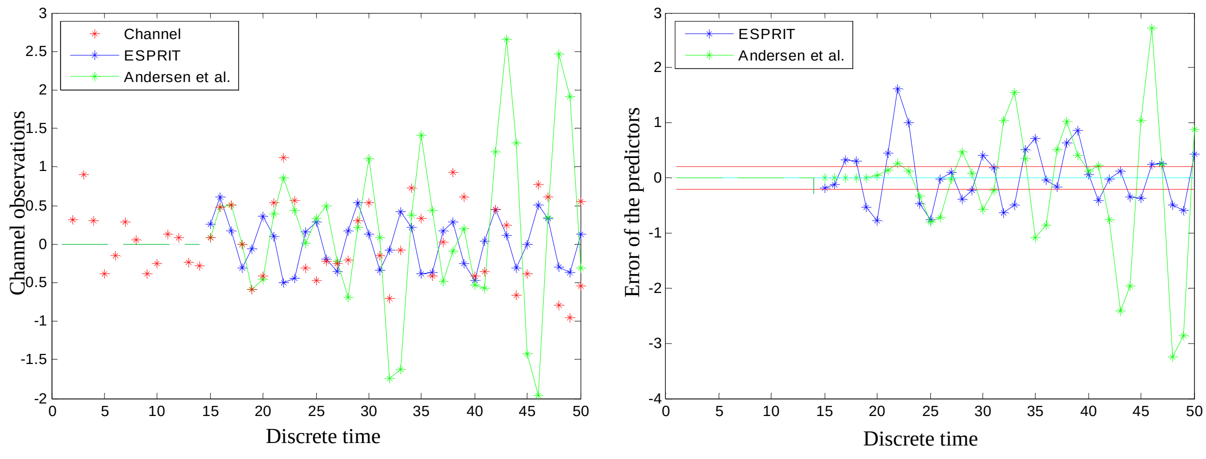


Figure 7-2 : Another sample result of the SISO predictors (*E* = 14).

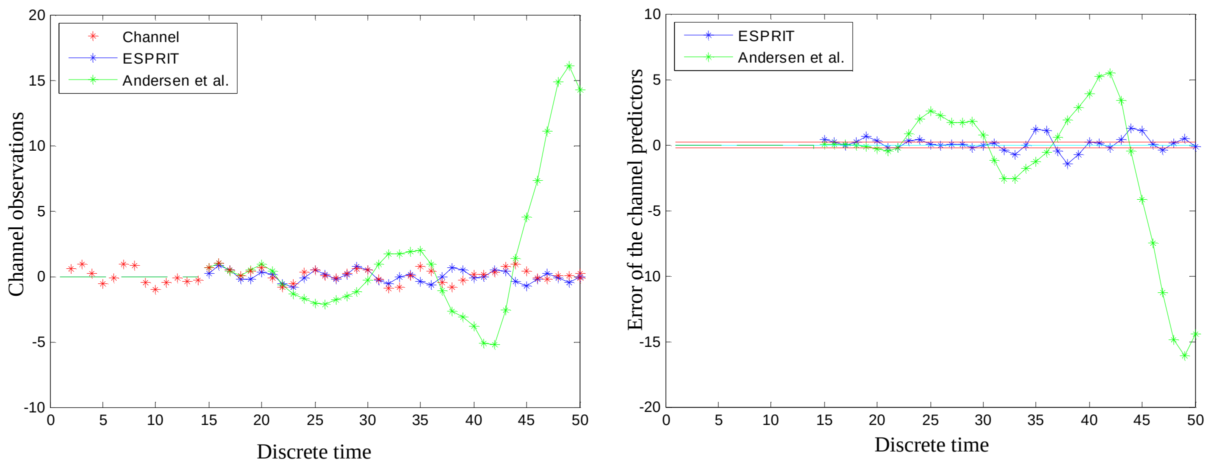


Figure 7-3 : More sample results of the SISO predictors (E = 14).

Figure 7-4 presents another execution of the predictors, but the predictors are now computed with E = 20 values (warm-up period of 20 x 7.5 x 0.5 = 75 ms).

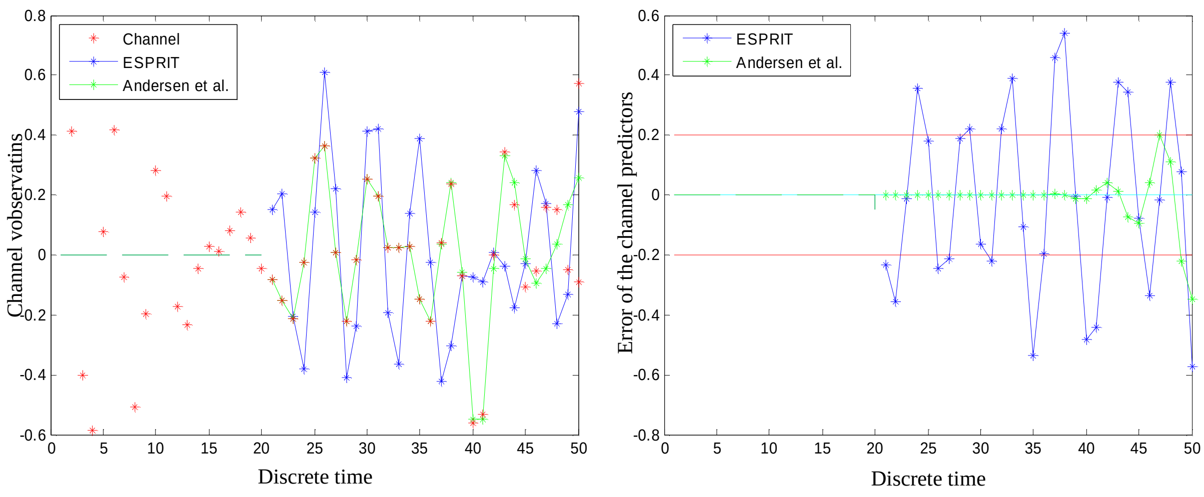


Figure 7-4 : Sample result of our SISO predictors (E = 20).

The Andersen et al. predictor seems to perform much better than the standard ESPRIT method. Its predictions remain reliable for a much longer time span. On the other hand, the error of the Andersen et al. predictor is exponentially increasing, which is not the case with the ESPRIT method.

The exponentially increasing error of the Andersen et al. predictor appears because the signal poles are not normalized. Actually, they should be unitary, because of their physical meaning. If the modulus of a computed pole is larger than unity, it will bring instability to the system. On the contrary, if its modulus is smaller than unity, the effect of the pole will fade. Therefore, adding a normalization step while computing the signal poles will make the Andersen et al. predictor less instable.

To assess the benefit of this normalization step, a run of the two predictors, built from E = 14 observations, where the Andersen et al. predictor has an exponentially increasing error, is shown in **Figure 7-5** and **Figure 7-6**.

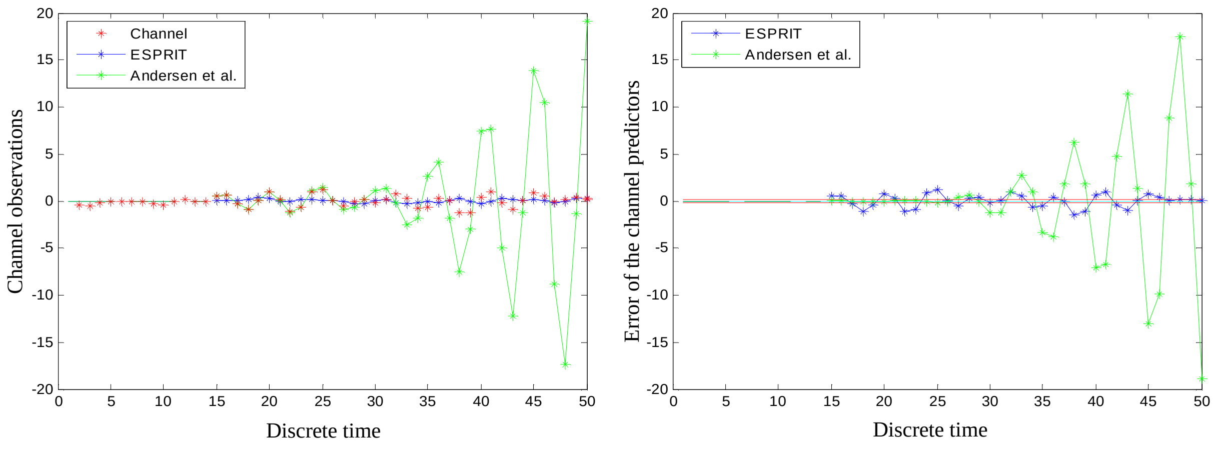


Figure 7-5 : A result sample of our SISO predictors

With the same observations, and a normalization step, the outcome of the Matlab computations significantly changes, as shown in **Figure 7-6**. Pay attention to the difference of scale!

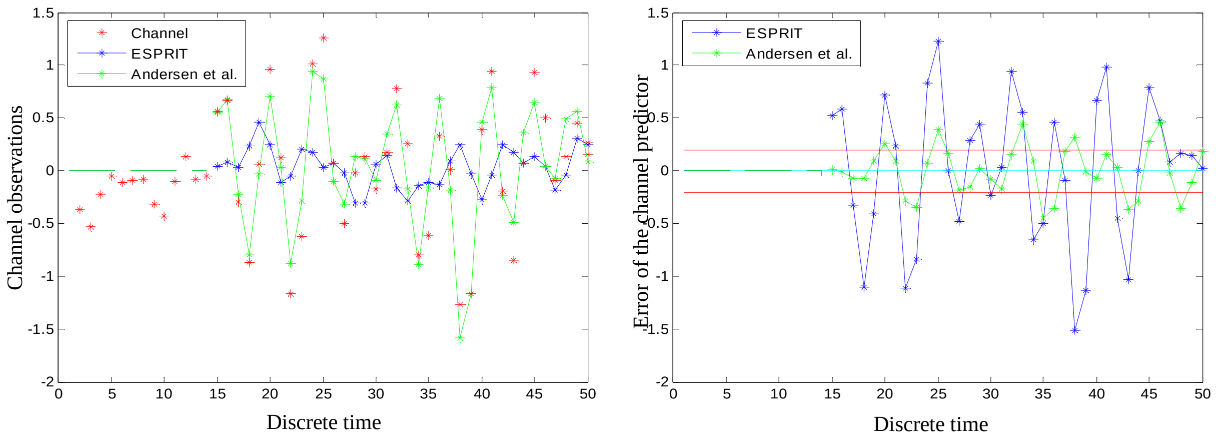


Figure 7-6 : The same result sample of our SISO predictors as in the figure above, with normalized poles, for the Andersen et al. predictor

The normalized Andersen et al. predictor has an error level below 0.2 during the 5 first steps, whereas the unnormalized Andersen et al. predictor has an error below 0.2 for 12 steps! But if one sets 0.5 as threshold for the maximum tolerated error, the predictor with normalized poles will remain below the threshold until the end of the Matlab run. With unnormalized poles, the error raises above the threshold as soon as the 28th prediction step.

Hence, it seems that the normalization step is better for long term predictions, but short term predictions are not so good with that step. With normalization, the Andersen et al. predictor is rather good all along the computation run. Without it, it is really accurate in the short term, but its performance degrades along time. For the rest of this section, the investigated Andersen et al. predictor is the original one, e.g. without normalization of the signal poles.

Please keep in mind that there is no rationale to set the maximum tolerated error level. Of course 0.2 is better than 0.5, but we can not decide what is affordable and what is not. Therefore we will now use an averaging metric to assess the performance of the predictors.

7.2 Performance indicator

The following sections will discuss the Normalized Mean Square Error (NMSE) achieved by the predictors. This performance indicator will produce averaged results, instead of sample results as in the former subsection.

The NMSE for a given future time instant t_0 (with $t_0 > E$) is defined by

$$NMSE(t_0) = \sum_{t=E+1}^{t_0} \frac{\left|\left(h(t) - \hat{h}(t)\right)^2\right|}{(t_0 - E) \cdot Px}$$

It is the average of the squared difference between the reference channel (h) and the prediction ($\hat{h}$). It is normalized with respect to the average power Px of the channel, which is computed from

$$Px = \sum_{t=E+1}^{E+K} \left|(h(t))^2\right|;$$

where K stands for the considered prediction horizon.

7.3 Performance of the predictors

In this section, we will assess both predictors, in noise free conditions. First, we will see the behavior of the predictors when initialized from 14 channel observations. It corresponds to an 8x7 Hankel matrix. **Figure 7-7** shows the time evolution of their NMSE, averaged on 500 runs. There is not a great difference between the two techniques.

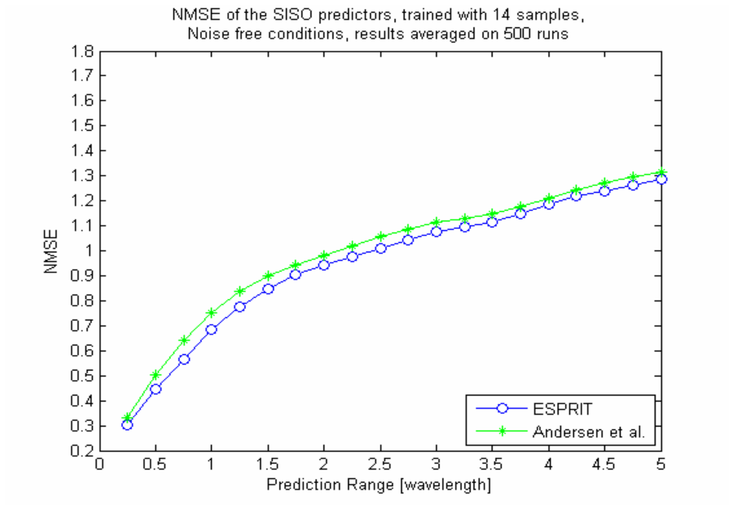


Figure 7-7 : Time evolution of the NMSE for the SISO predictors in noise free conditions (E = 14)

We also tried to collect different numbers of observations to build our predictors, by choosing E = 20, and E = 8. These results are presented at **Figure 7-8**. The computations point out the similar behavior of the two predictors, and the influence of the number of received signal observations used to build the predictors. The more observations we used, the smaller the NMSE is.

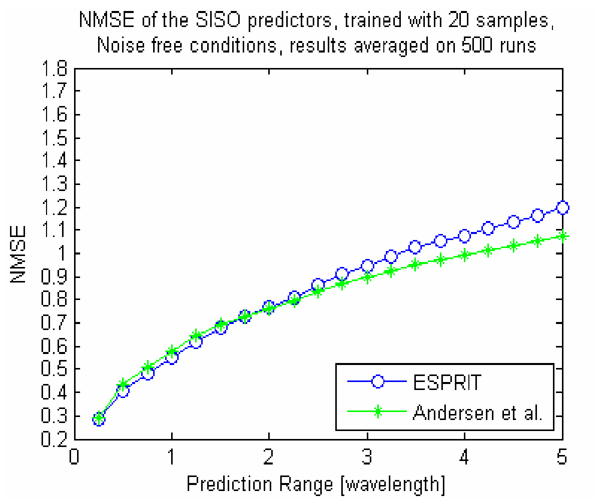


Figure 7-8 : Time evolution of the NMSE for the SISO predictors in noise free conditions(E = 20 and E = 8)

Figure 7-9 compares the influence of the number of observations for each predictor separately. Using only 8 observations seems to be not enough, but the difference between 14 and 20 observations is slight. Keep in mind that for collecting 8 observations, we need to wait 30 ms, so 60 TTI, and for 20 observations, it requires 75 ms, so 150 TTI. Moreover, the more observations we use, the more the computations are demanding.

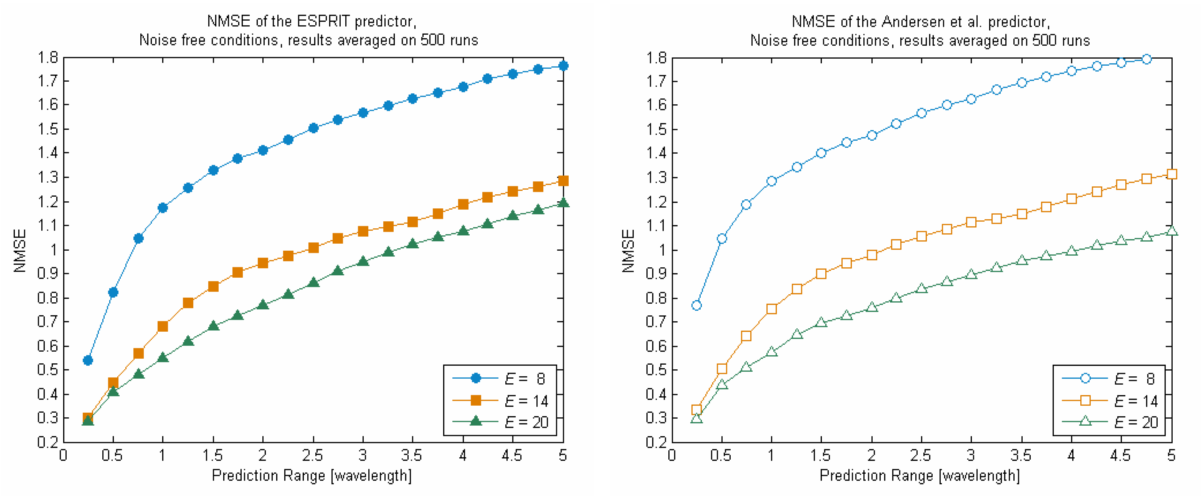


Figure 7-9 : Time evolution of the NMSE for the SISO predictors, noise free conditions

All these computations were performed in noise free conditions. Let us see how the predictors are robust against noise.

7.4 Noisy computations

Now, we want to check the robustness of our predictors against noise. So we added white Gaussian noise to the results of SCME, and ran our predictors on these modified data. Here are some graphics showing, for each algorithm, the time evolution of the NMSE with four noise levels - the Signal to Noise Ratio (SNR) is 100 dB, 40 dB, 20 dB and 3 dB. These results are compared with the noise free situation.

Figure 7-10 presents for each predictor, trained with 20 channel observations, the time evolution of the NMSE at various SNR's.

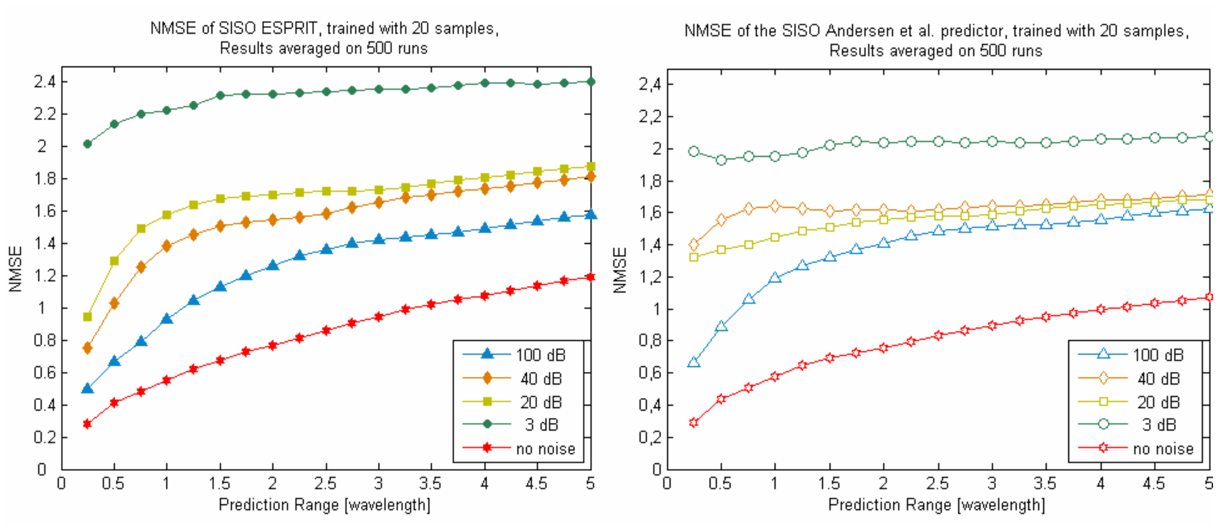


Figure 7-10: Time evolution of the NMSE for our predictors in noisy situations (E = 20)

With added noise, a difference appears between the two predictors. ESPRIT produces better results for the first time steps, but after a while, its NMSE overtakes the one of Andersen et al. And the higher the SNR is, the longer ESPRIT is better than Andersen et al. At the low SNR of 3 dB, the performances are really poor, especially for ESPRIT.

In a same way, **Figure 7-11** presents the time evolution of the NMSE for each predictor, but now trained with only 8 channel observations, still at various SNR's.

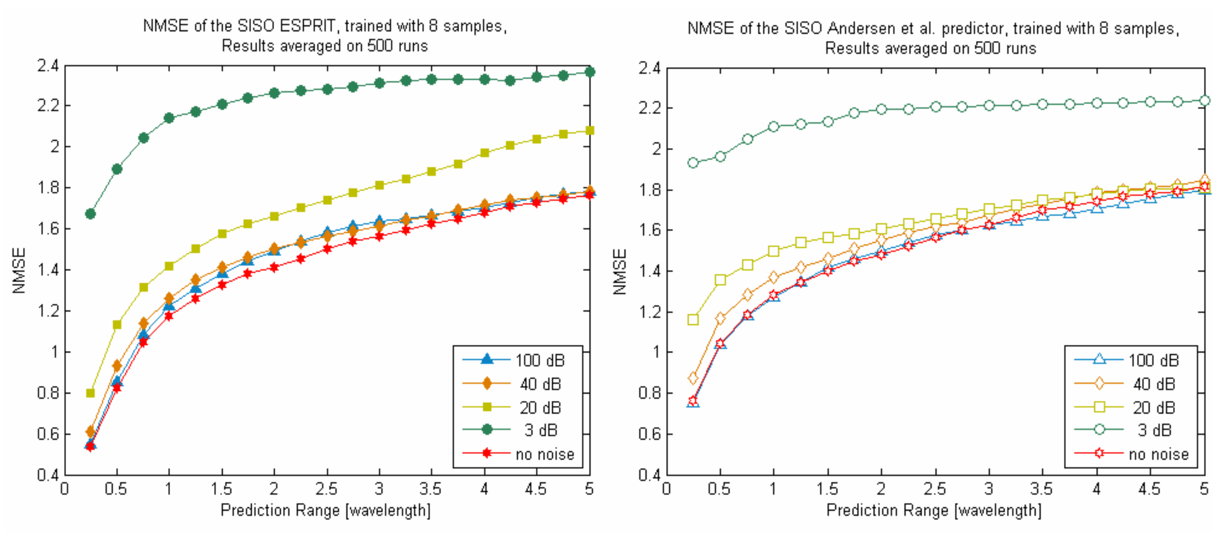


Figure 7-11: Time evolution of the NMSE for our predictors in noisy situations (E = 8)

With only 8 observations to train the predictors, the influence of the noise is very limited. The noise free curve is really close to the ones at SNR of 40 and 100 dB, and the curves at SNR = 20 dB are not so far away. The situation with SNR = 3 dB still produces poor results.

We also still have that ESPRIT produces lower NMSE at first time steps than the Andersen et al. technique and again, the higher the SNR is, and the longer ESPRIT is better than Andersen et al. Keep in mind that ESPRIT uses noise variance to set the value of N_S, which influences the number of signal poles implied in the computations. On the other side, the Andersen et al. technique only discards poles with modulus larger than 1.05, which is a default value not depending on the noise level. This is therefore an explanation for the better results of ESPRIT in noisy situations, at least for the first predicted values.

While in noise free conditions, we said that the more observations we use to train the predictors and the more accurate they are, when Gaussian noise has been added, using only 8 observations produce better results than with 20 observations. **Figure 7-12** shows that comparison with SNR = 20 dB.

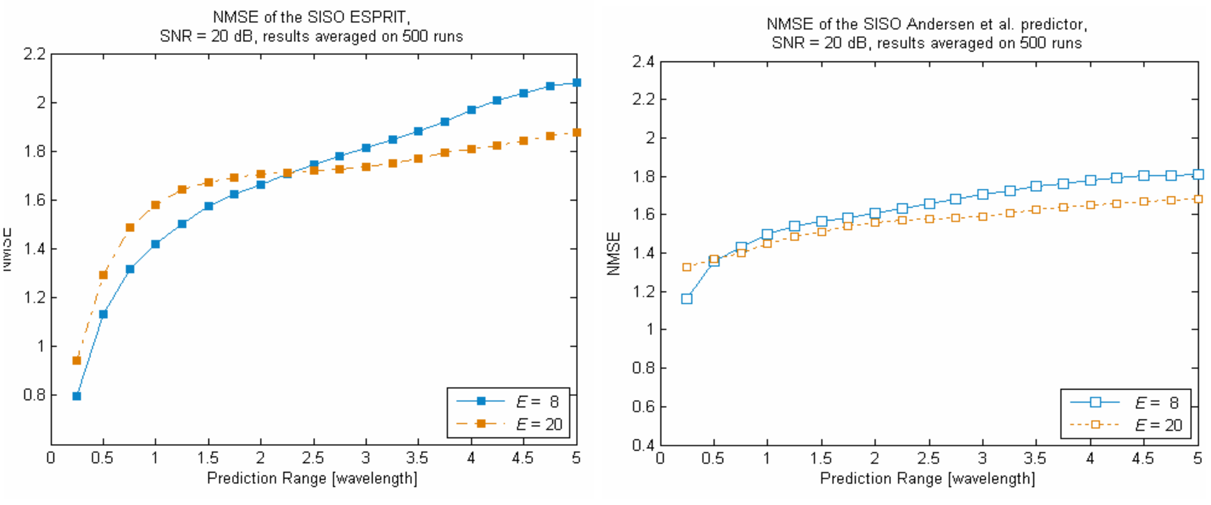


Figure 7-12 : Influence of the number of observations in noisy conditions (SNR = 20 dB)

Actually, it is only at SNR of 100 dB or in noise free conditions that predictors work better with more observations. This is maybe because increasing the number of observations increases more the influence of the noise than the amount of useful information.

7.5 SISO conclusions

Because noise free conditions are not practical, and even a SNR of 100 dB seems unrealistic, it seems more efficient to limit the number of channel observations used to train the predictors. Moreover, it makes the computations less demanding, and it requires shorter warm-up time.

About the choice of the technique, ESPRIT seems to perform better than the other, especially for the first time steps. It is maybe because it adapts its predictions according to the noise level. But on drawback it means that it requires estimations of that noise level.

Having covered the SISO case, let us move now to the MIMO case.

8 MIMO PREDICTORS

In this section, we will present innovative MIMO extensions of the ESPRIT and the Andersen et al. predictors.

The idea is now to find the poles for all the transmit/receive antenna pairs of the MIMO channel in a single operation. Of course, we can consider that a $n_T \times n_R$ MIMO is like $n_T \cdot n_R$ SISO channels, but if we run $n_T \cdot n_R$ times the SISO algorithm, we will not exploit the spatial correlation there is between these SISO channels.

Once we have identified the poles of the MIMO channel, we compute the amplitudes for each transmit/receive antenna pair like in the SISO case.

We will present the MIMO extension of ESPRIT then the MIMO extension of the Andersen et al. technique in the following subsections. To simplify the notations, we will consider a 2 x 2 MIMO, but these predictors also work with larger antenna arrays, as we will see in the following chapter.

8.1 The MIMO extension of ESPRIT

This subsection presents the MIMO extension of the original ESPRIT technique. This MIMO predictor will be called MECoM, for MIMO ESPRIT Based on Correlation Matrix. Remembering the summary algorithm of the SISO ESPRIT in Section 5.7, the first step of the algorithm is the writing of the time domain correlation matrix.

So here is a way to get it for our MIMO channel. First, we still have to fill a "Hankel" matrix with the channel observations. Beforehand, we will define MIMO channel vectors:

$$h_{MIMO}(t) = \left(h^{11}(t),\quad h^{12}(t),\quad h^{21}(t),\quad h^{22}(t)\right) \equiv \text{vectorized channel matrix } \mathbf{H}.$$

$$\mathcal{H} = \begin{pmatrix} h_{MIMO}(1) & h_{MIMO}(2) & \cdots & h_{MIMO}(m) \\ h_{MIMO}(2) & h_{MIMO}(3) & \cdots & h_{MIMO}(m+1) \\ \vdots & \vdots & \ddots & \vdots \\ h_{MIMO}(n) & h_{MIMO}(n+1) & \cdots & h_{MIMO}(m+n-1) \end{pmatrix}$$

$$= \begin{pmatrix} h^{11}(1) & h^{12}(1) & h^{21}(1) & h^{22}(1) & \cdots & h^{11}(m) & h^{12}(m) & h^{21}(m) & h^{22}(m) \\ h^{11}(2) & h^{12}(2) & h^{21}(2) & h^{22}(2) & \cdots & h^{11}(m+1) & h^{12}(m+1) & h^{21}(m+1) & h^{22}(m+1) \\ & & \vdots & & & & \vdots & & \\ h^{11}(n) & h^{12}(n) & h^{21}(n) & h^{22}(n) & \cdots & h^{11}(m+n-1) & h^{12}(m+n-1) & h^{21}(m+n-1) & h^{22}(m+n-1) \end{pmatrix}$$

The incidence of parameters *m* and *n* should be further discussed. The phenomena we described in the SISO case (Section 5.4) are still relevant in the MIMO case. For our computations, we chose *m* and *n* such that $\mathcal{H}$ is nearly a square matrix, with slightly more rows than columns.

Then, we compute an estimation of *R* by $\frac{\mathcal{H}\cdot\mathcal{H}^*}{m}$.

When we have obtained the correlation matrix of our MIMO channel, we use the same algorithm as in the SISO case. It means that we compute the *S* matrix from the SVD of *R* ($\rightarrow R = U.\Sigma.V$, and *S* is filled with the first N_s columns of *U*; N_s comes from the number of singular values of *R* larger than the variance of the noise).

With this *S* matrix, we can get the $\mathcal{D}$ matrix by solving $S_{UP}\cdot\mathcal{D} = S_{DOWN}$. The MIMO signal poles (the z_i) are then the eigenvalues of $\mathcal{D}$.

Once the poles have been identified, we derive the amplitudes. We can compute them from $Y^{m\,n} = Z\cdot a^{m\,n}$, where *Z* is a Vandermonde matrix filled with the MIMO poles, and $Y^{m\,n}$ is a vector containing the observations corresponding to the *m*th transmitting antenna and the *n*th receiving antenna.

$$\Rightarrow \quad a^{m\,n} = (Z^*Z)^{-1}\cdot Z^*\cdot Y^{m\,n}$$

Therefore, we have the poles of the MIMO channel (the z_i) and the amplitudes corresponding to these poles for each transmitting-receiving antenna pair (the $a^{m\,n}\ \ \forall\, m, n$), and we are able to do predictions:

$$\forall t = E, E+1, \ldots \quad \hat{h}^{m\,n}(t) = \sum_{k=1}^{N_s+1} a_k^{\,m\,n}\, z_k^{\,t}\ .$$

We will discuss later about the reliability of these predictions, and we will also show results of Matlab computations. But before this performance assessment, we will present our MIMO extension of the Andersen et al. predictor.

8.2 The MIMO extension of the Andersen et al. predictor

This section will now extend the Andersen et al. algorithm to MIMO scenarios. This MIMO predictor will be called MEHaM, for MIMO ESPRIT Based on Hankel Matrix. As it was detailed in Section 6.4, the Andersen et al. technique does not require a correlation matrix. Instead, one directly applies the analysis to a Hankel matrix containing the channel observations. In the MIMO case, the procedure is similar, with the following matrix:

$$\mathcal{H} = \begin{pmatrix} h^{11}(1) & h^{12}(1) & h^{21}(1) & h^{22}(1) & \cdots & h^{11}(m) & h^{12}(m) & h^{21}(m) & h^{22}(m) \\ h^{11}(2) & h^{12}(2) & h^{21}(2) & h^{22}(2) & \cdots & h^{11}(m+1) & h^{12}(m+1) & h^{21}(m+1) & h^{22}(m+1) \\ & & \vdots & & & & \vdots & & \\ h^{11}(n) & h^{12}(n) & h^{21}(n) & h^{22}(n) & \cdots & h^{11}(m+n-1) & h^{12}(m+n-1) & h^{21}(m+n-1) & h^{22}(m+n-1) \end{pmatrix}$$

Again, the choice of our parameters *m* and *n* should be discussed. They should be chosen with the same rule than in the MIMO version of ESPRIT. For our computations, we will use the same $\mathcal{H}$ matrices (nearly square, with slightly more rows than columns).

When we have generated the $\mathcal{H}$ matrix, we compute its SVD $\left(\rightarrow \mathcal{H} = U.\sum.V\right)$, and take the first columns of the U matrix (U_1) to compute the $\mathcal{D}$ matrix:

$$J_{UP} \cdot U_1 \cdot \mathcal{D} \approx J_{DOWN} \cdot U_1,$$

such as $J_{UP} \cdot U_1$ and $J_{DOWN} \cdot U_1$ are square matrices.

Again, the poles of the MIMO channel are the eigenvalues of $\mathcal{D}$ which are smaller than 1.05.

Then we have to compute the amplitudes for each transmitting-receiving antenna pair, like we did with MECoM:

$$a^{m\,n} = (Z * Z)^{-1} \cdot Z * \cdot Y^{m\,n}$$

Therefore, the predictions are given by: $\forall t = E, E+1, \ldots \quad \hat{h}^{m\,n}(t) = \sum_{k=1}^{N_s+1} a_k^{m\,n} \, z_k^{\,t}$.

8.3 Numerical complexity of these predictors

Before presenting computation results, we investigated the numerical complexity of the two predictors. They both require 3 SVD of matrices with size n_T x n_R, which can be done in $O\left(min\left\{n_T n_R^2 \, , n_T^2 n_R\right\}\right)$.

The other computations to get the predicting parameters can be done $O\left(n_T^3 n_R^3\right)$, but MEHaM requires two times less computations. Moreover, MEHaM does not require to estimate the noise variability, so the computational load is further reduced.

Once the signal poles and the amplitudes are known, predicting the channel state for the next time step can be done in $O\left(n_T n_R N_S\right)$.

In the next section, the performance of these two ESPRIT-based MIMO predictors are assessed with Matlab computations on SCME observations.

9 MIMO COMPUTATIONS

In this section, we will present results from Matlab computations of the MIMO extensions of both ESPRIT and Andersen et al. predictors, respectively MECoM and MEHaM. We will still rely on WINNER's SCME to provide us with a simulated MIMO channel, using its default values listed in Section 2. Matlab was run in its default resolution, which is 64 bits.

9.1 A typical computation

Figure 9-1 to **Figure 9-4** first illustrate the results of a typical execution of both predictors in noise free conditions, showing the difference of behaviour between them. Both collected *E* = 11 channel observations. Because the MIMO set-up is 2 x 2, there are four plots in **Figure 9-1**. Each of them shows the channel observations in red, and the predicted values in green.

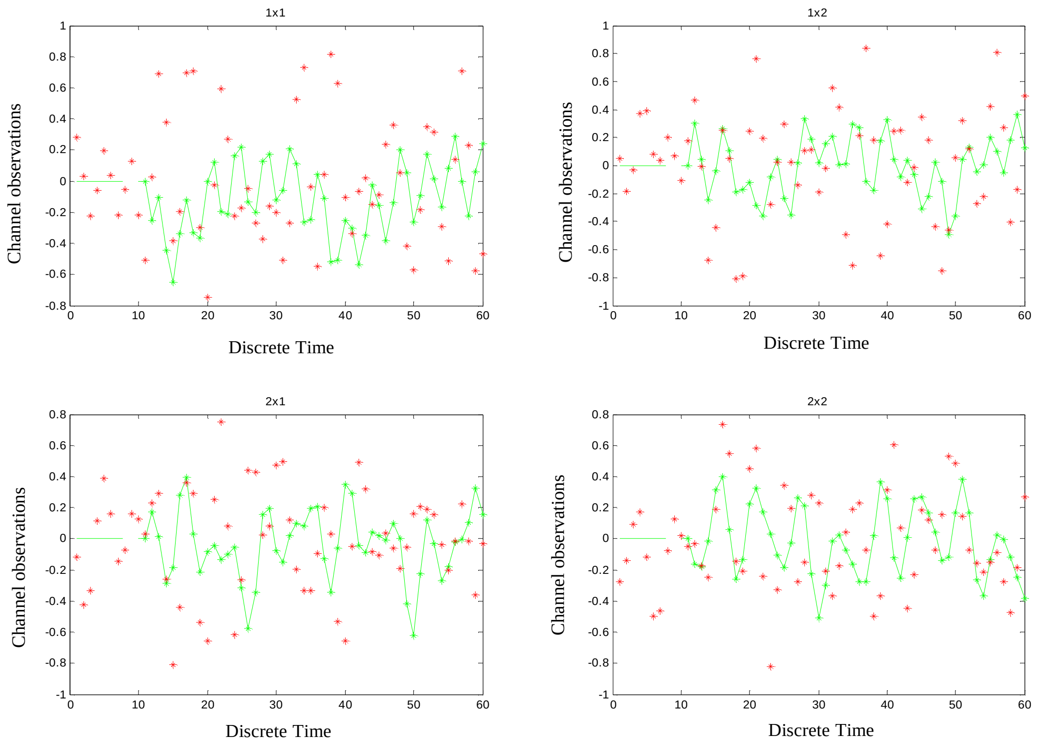


Figure 9-1 : A typical MECoM computation

Figure 9-2 presents the error curves for the MECoM algorithm. It has rather poor performance.

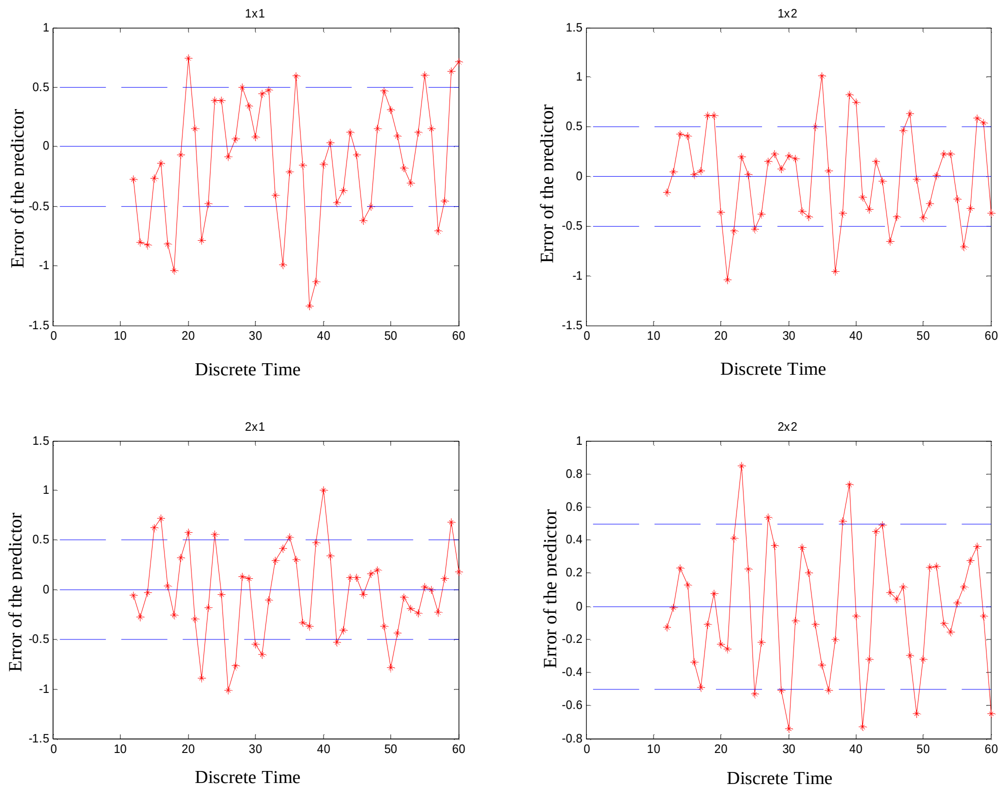


Figure 9-2 : Error of a typical MIMO ESPRIT computation

On the other hand, **Figure 9-3** shows a typical run on the MEHaM predictor.

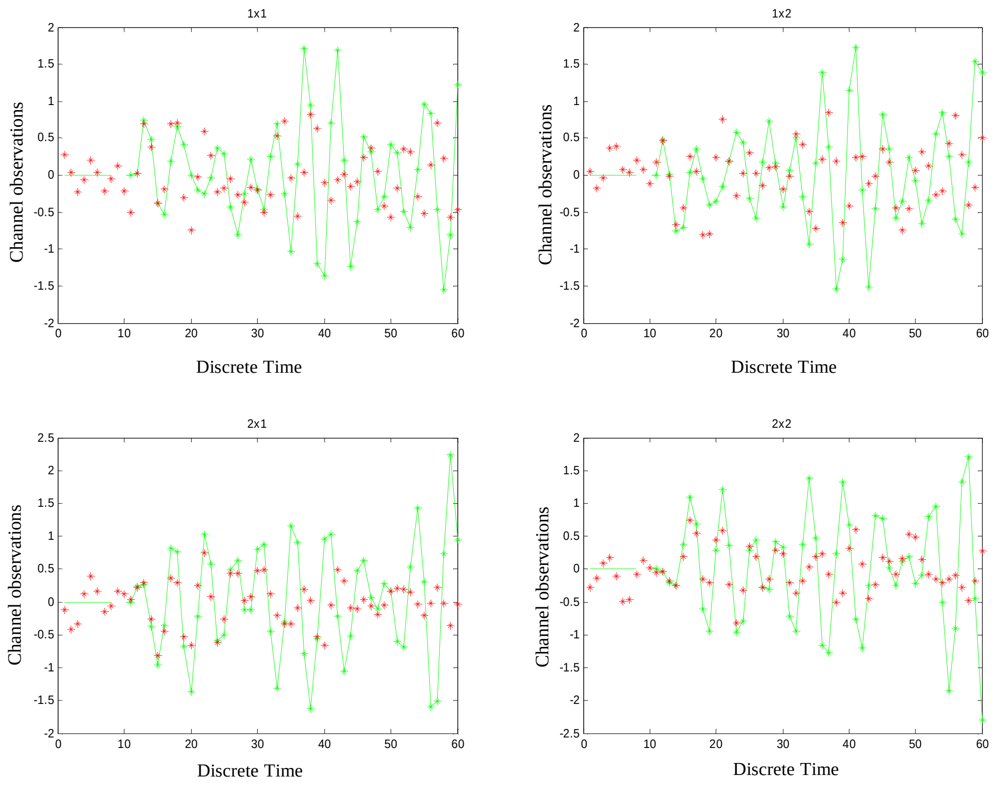


Figure 9-3 : A typical MIMO Andersen et al. predictor computation

The prediction error is shown in **Figure 9-4**.

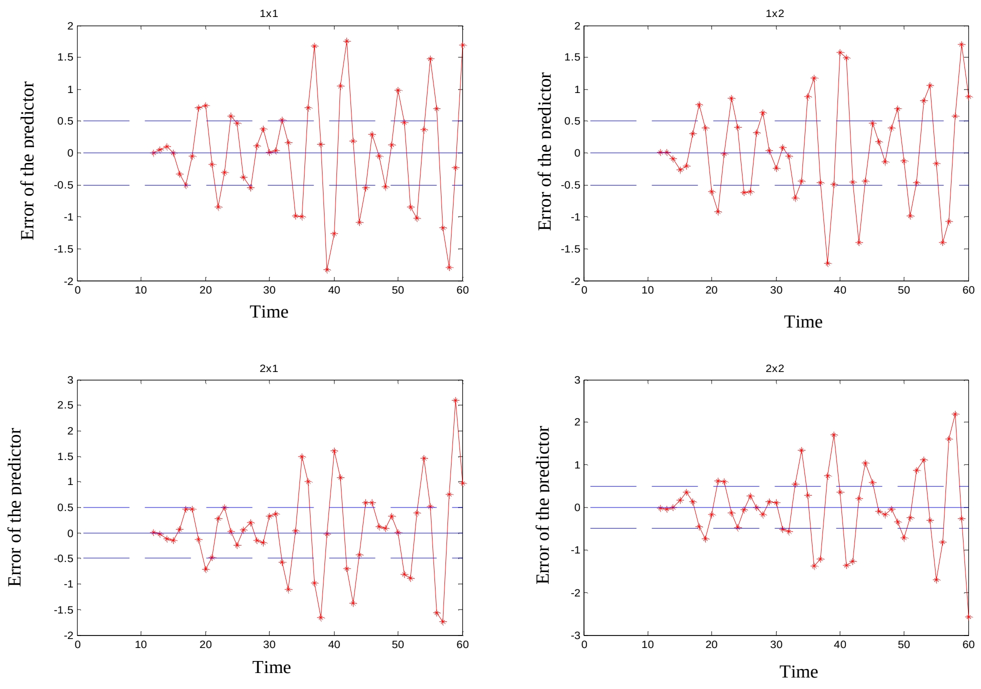


Figure 9-4 : Error of a typical MIMO Andersen et al. predictor computation

It appears that MEHaM performs better than MECoM, at least from this single typical run. Let us now consider the averaged performance of those predictors, to have more consistent figures.

9.2 NMSE in MIMO scenarios

As in the SISO case, we will consider the Normalized Mean Square Error. In MIMO scenarios, the NMSE for a given future time instant t_0 (with $t_0 > E$) is defined by

$$NMSE(t_0) = \sum_{t=E+1}^{t_0} \sum_{n=1}^{n_T} \sum_{m=1}^{n_R} \frac{1}{Px} \frac{\left| \left(h^{nm}(t) - \hat{h}^{nm}(t) \right)^2 \right|}{(t_0 - E) \cdot n_T \cdot n_R}$$

where Px, the mean power of the MIMO channel, is obtained by

$$Px = \sum_{n=1}^{n_T} \sum_{m=1}^{n_R} \sum_{t=E+1}^{E+K} \left| \left(h^{nm}(t) \right)^2 \right|$$

and K stands for the considered prediction horizon.

9.3 Performance of the predictors

This section will assess the performance of the MIMO predictors in noise free conditions. **Figure 9-5** shows the time evolution of the NMSE in a 2x2 scenario. Predictors where trained with 11 MIMO channel observations, which means a 9x8 Hankel matrix, or with 20 MIMO channel observations, which implies a 17x16 Hankel matrix. Averages where performed on 1,000 runs.

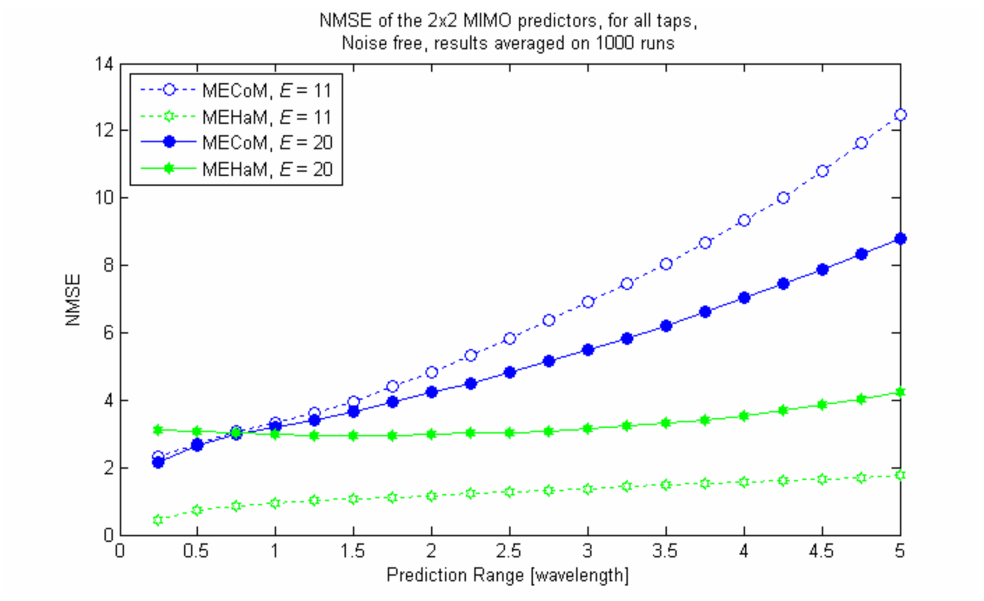


Figure 9-5 : Time evolution of the NMSE of our MIMO predictors in noise free conditions

MEHaM, trained with 11 MIMO channel observations produces the best results. Using more observations causes an increase of the NMSE to the value of 3. For MECoM, results are really poor – pay attention to the scale. But there it seems using 20 observations is better than using only 11 observations.

Let us add some Gaussian noise, to see how much the predictors are robust against noise.

9.4 Adding coloured noise

As in the SISO case, we have added Gaussian noise to the reference channel to model background noise and multiuser interference. That noise was white in time domain, but coloured in the space domain, to reflect spatial correlation of the MIMO channel.

We tested the two predictors with different noise levels. We used a Signal to Noise Ratio (SNR) of {3, 20, 40, 100} dB. In noise free conditions, singularity problems occur. This is the reason why we tested a SNR of 100 dB. Thanks to the noise, as weak as it can be, the Hankel matrix $\mathcal{H}$ has independent rows and columns. This solves the singularity issue.

Remember that in SISO case, when Gaussian noise was added, we found it is better to limit the number of channel observations used to train the predictor. We also had that ESPRIT performed better than the Andersen et al. technique, especially at low SNR's. In MIMO noise free conditions, it seems that only MEHaM, so the MIMO extension of the Andersen et al. technique, produces consistent results.

We will see now how the MIMO predictors behave when noise is added. We will present first results obtained with predictors trained with 11 MIMO channel observations, and then we will show what happens when using 20 or 5 observations instead.

9.4.1 Using 11 observations on 2x2 MIMO

We will start our computation with predictors built on $E = 11$ observations for each antenna pair. **Figure 9-6** first presents the results for MECoM first. We did 1,000 executions, and we tested four levels of noise: SNR = 100 dB, 40 dB, 20 dB and 3 dB.

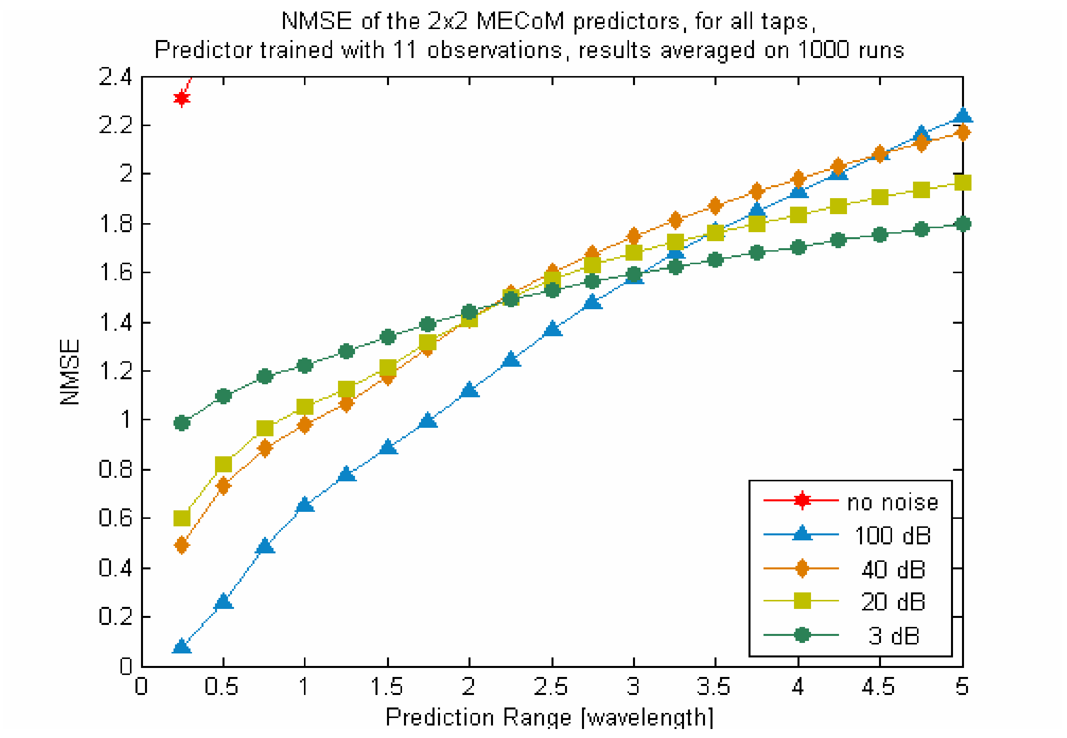


Figure 9-6 : Time evolution of the NMSE for MECoM in noisy situations ($E = 11$)

As you can see, the noise free curve is corrupted by singularity problems which disappear when adding a few noise, even at the high SNR of 100 dB. On **Figure 9-6**, the graphic has been truncated to present clearly the noisy curves, and therefore, only the first point of the noise free curve is visible.

High SNR obviously brings lower NMSE for the first values, but not for all. After 3 wavelengths for example, NMSE is lower with SNR = 3 dB than with SNR = 100 dB.

On the other hand, **Figure 9-7** shows the results for MEHaM, also averaged on 1,000 executions. The curves are more stationary. First NMSE values are larger than with MECoM, but they do not increase like the MECoM ones. As with MECoM, the lower the SNR is, the more constant the NMSE will be. After half a wavelength, NMSE are lower at SNR of 20 dB than 100 dB.

Let us point out that for both predictors, having a SNR or 20 dB or 40 dB barely lead to the same NMSE curves. One can also notice that MECoM with SNR = 20 dB lead to the same NMSE curve as MEHaM with SNR = 100 dB does.

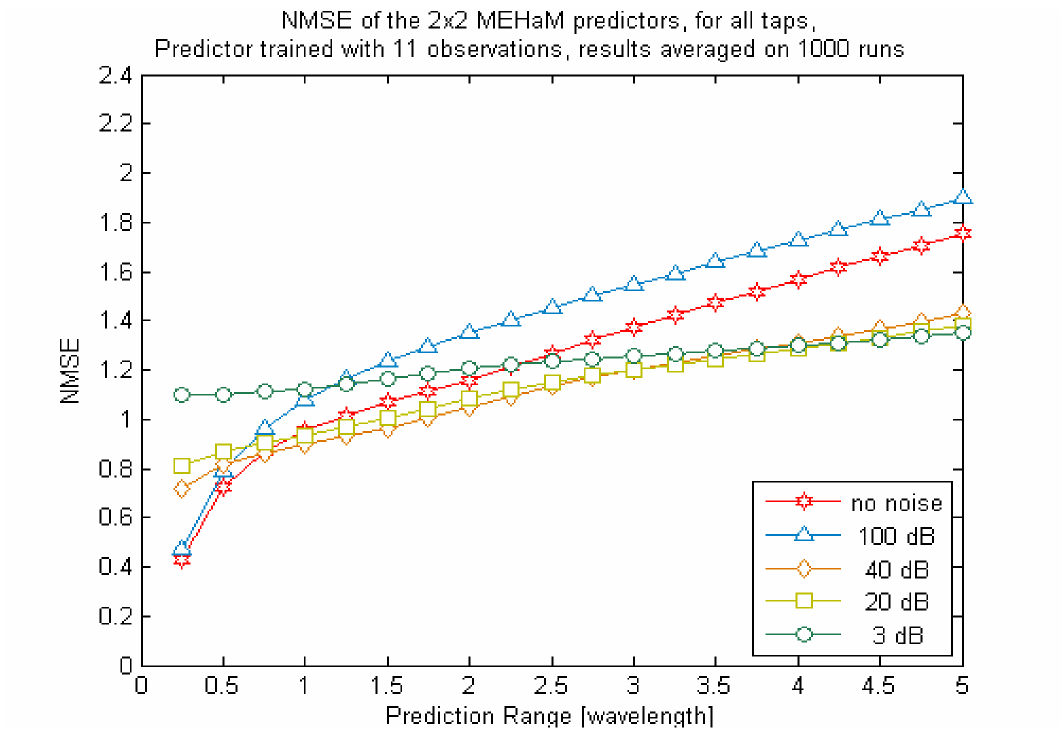


Figure 9-7 : Time evolution of the NMSE for MEHaM in noisy situations (*E* = 11)

If we want to compare the two predictors, at the realistic SNR's of 3 and 20 dB, as it is shown on **Figure 9-8**, we should say that MECoM produces better results for the first time steps, but then it is MEHaM which has the lowest NMSE curve. So we find the same conclusion as we get for the SISO case. Therefore, if we consider prediction horizon longer than half a wavelength, MEHaM seems to be a better predictor than MECoM.

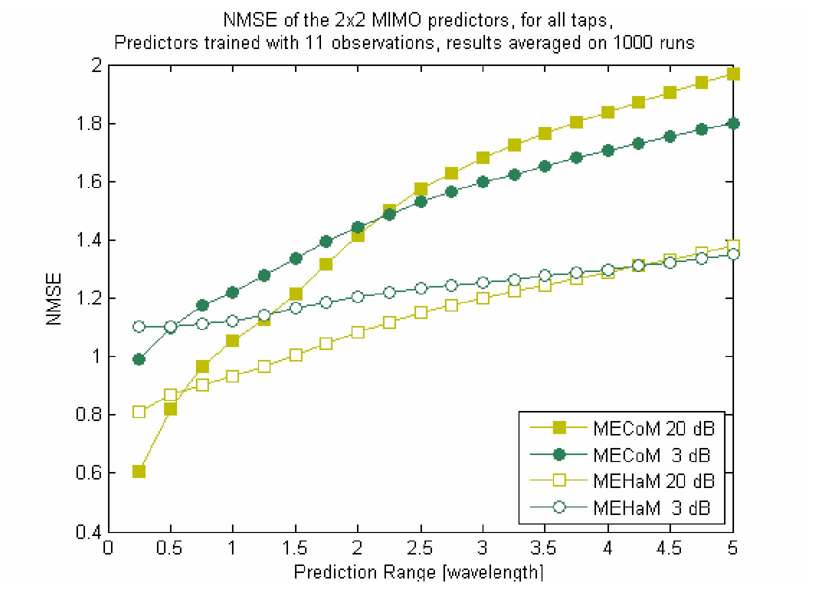


Figure 9-8 : Reliability of our two MIMO predictors (*E* = 11, SNR = 40 dB or 3 dB)

We will try to improve the predictions by increasing the number of channel observations we use to build the predictors.

9.4.2 Using 20 observations on 2x2 MIMO

So we will use now 20 observations of each of our 4 ($n_T \cdot n_R$) antenna pairs. So it means that initialisation of the predictor (warm-up period) will take more time and that the computation load will be heavier, due to larger matrices.

Figure 9-9 shows the results for MECoM.

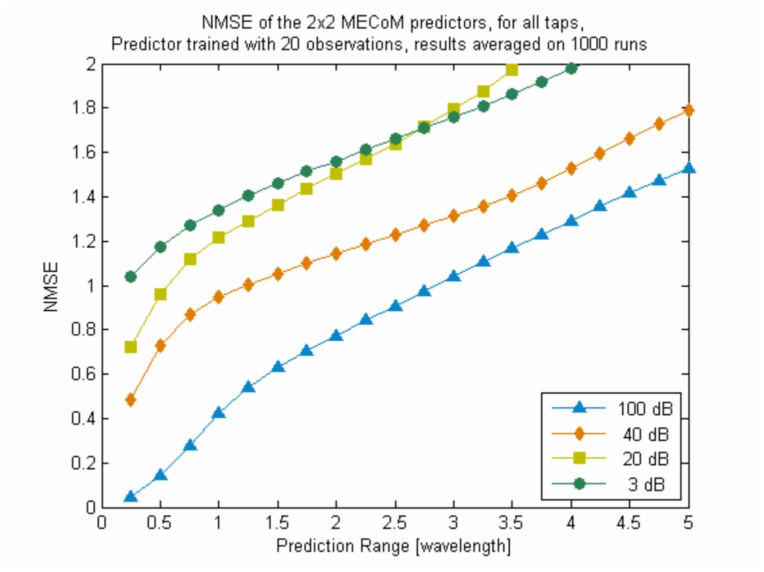


Figure 9-9 : Time evolution of the NMSE for MECoM in noisy situations (E = 20)

Except with SNR = 100 dB, the NMSE curves for MECoM are actually higher with E = 20 than with E = 11. And therefore, it is better to limit the value of E. Moreover, it makes the computations less demanding and makes the predictor faster to train.

The MEHaM's performance is shown in **Figure 9-10**. For MEHaM, it is a bit different. Using more observations lead to lower NMSE.

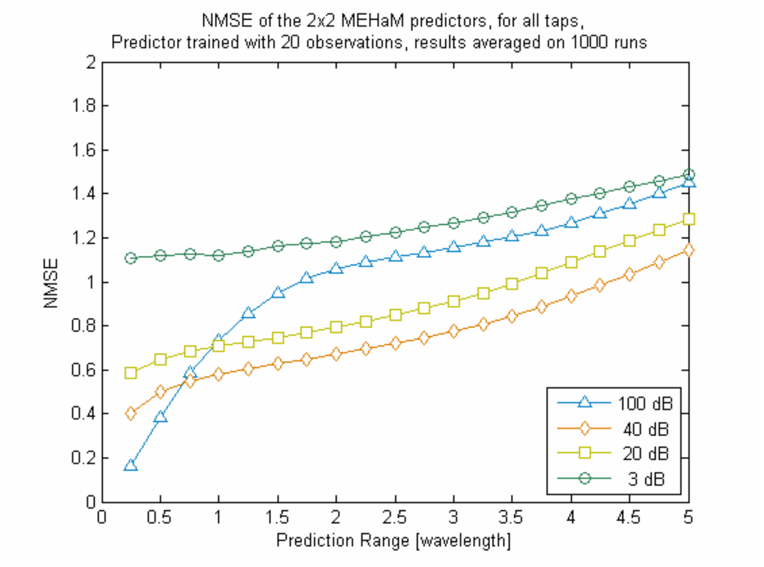


Figure 9-10 : Time evolution of the NMSE for MECoM in noisy situations (E = 20)

Because MECoM performs better with a limited number of observations, and because, we can not afford to wait and then to handle a large number of MIMO channel observations, we will now try to reduce the number of samples we need. With our setting, 11 channel observations require to wait 11 x 3.75 = 41.25 ms = 82.5 TTI, 20 observations need 75 ms = 150 TTI, but 5 observations can be collected within 18.75 ms = 37.5 TTI.

9.4.3 Using 5 observations on 2x2 MIMO

Because the Hankel matrix described in sections 8.1 and 8.2 should have more rows than columns, for a 2x2 MIMO channel, we need at least 5 MIMO channel observations.

It means that $$\mathcal{H} = \begin{pmatrix} h^{11}(1) & h^{12}(1) & h^{21}(1) & h^{22}(1) \\ h^{11}(2) & h^{12}(2) & h^{21}(2) & h^{22}(2) \\ h^{11}(3) & h^{12}(3) & h^{21}(3) & h^{22}(3) \\ h^{11}(4) & h^{12}(4) & h^{21}(4) & h^{22}(4) \\ h^{11}(5) & h^{12}(5) & h^{21}(5) & h^{22}(5) \end{pmatrix}.$$

Figure 9-11 presents the performance of the two MIMO predictors, trained with only $E = 5$ observations. Frequencies are still obtained from 1,000 executions.

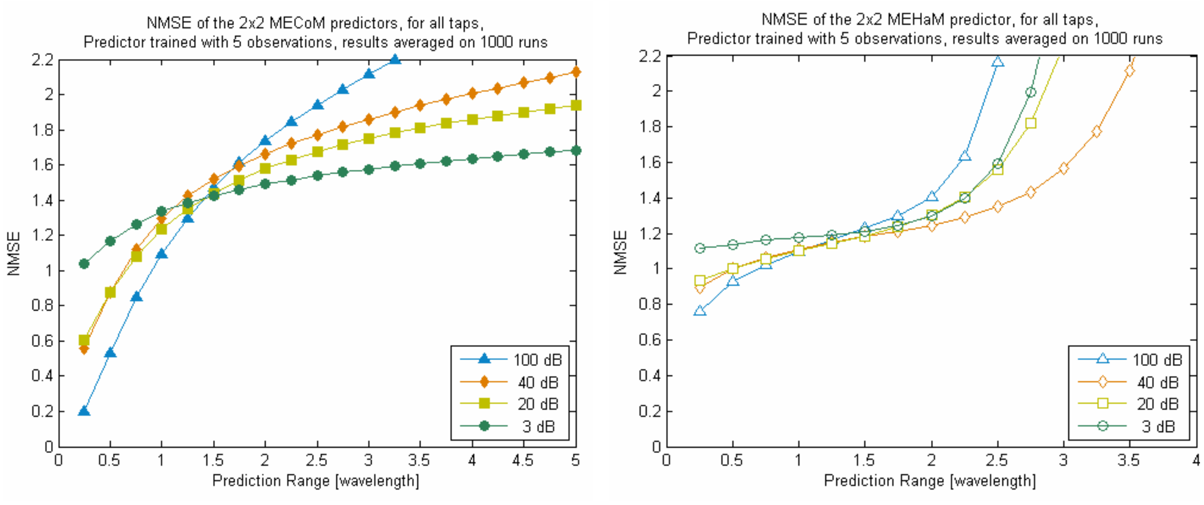


Figure 9-11 : Time evolution of the NMSE for MECoM and MEHaM in noisy situations (E = 5)

The MECoM performances are not so depredated when using only 5 MIMO channel observations, and are even equivalent for the first predictions. As far as MEHaM is concern, except for the unrealistic 100 dB curve, the NMSE are equivalent for E = 11 and E = 5 at the beginning. But after 2 wavelengths, the NMSE are exponentially increasing. We will now try to improve the MEHaM technique.

9.4.4 Improving MEHaM

One of the main difference between the two MIMO predictors is that MECoM uses the noise level (which has to be estimated) to set the N_S value to decide how many columns will be taken into account for computing the signal poles, whereas MEHaM uses a default value, which until now was the maximum value. This subsection will present the influence of this parameter, and will help to decide of a better default value.

Figure 9-12 shows the influence of that number of columns on the time evolution of the NMSE for 2x2 MEHaM predictor, at several noise levels. MEHaM was only trained with 5 MIMO channel observations.

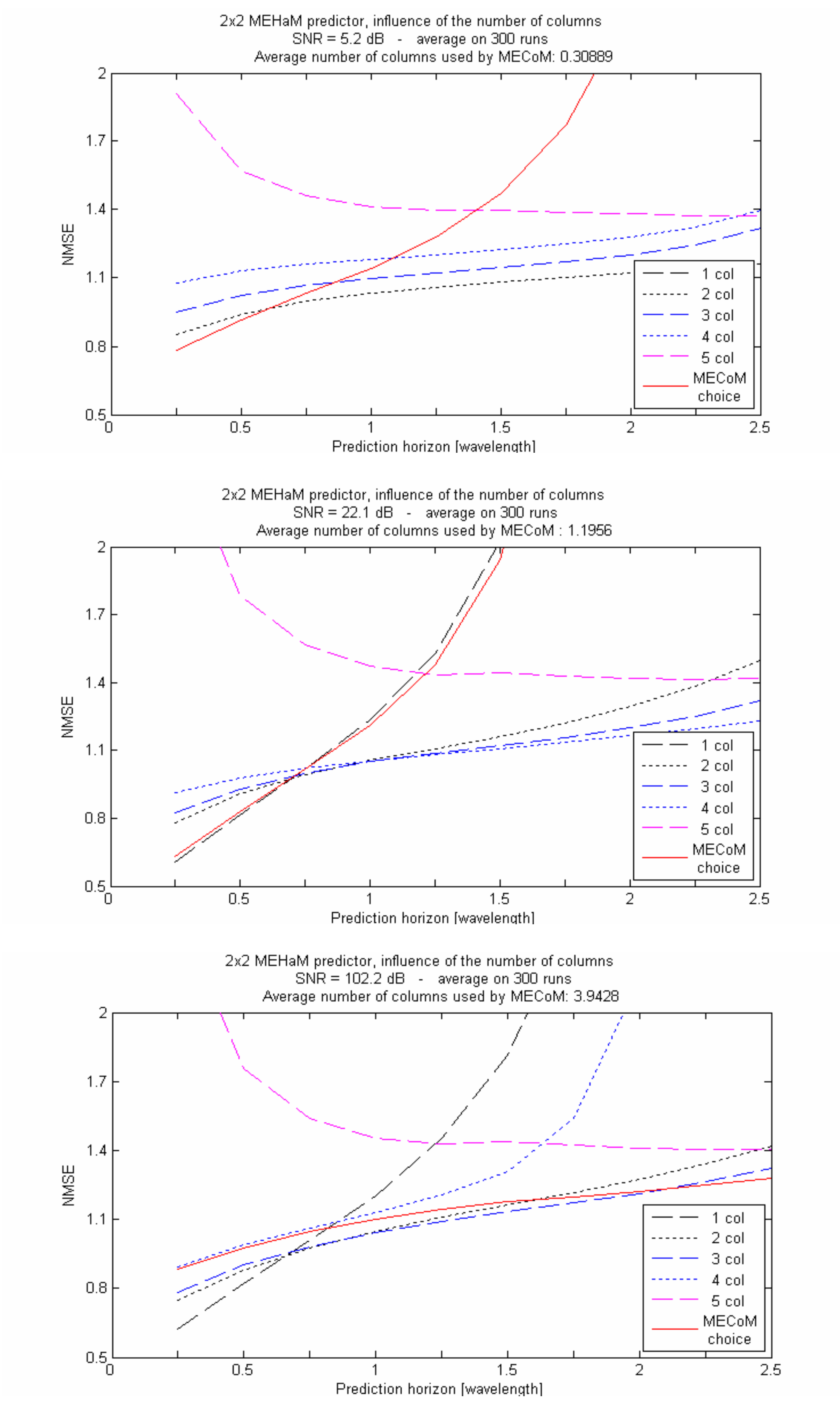


Figure 9-12 : Influence of the number of columns on the MEHaM predictor (5 obs, SNR = 3, 20, 100 dB)

As one can see on **Figure 9-12**, the optimal number of columns to use is the smallest. With only one column, MEHaM produces its lowest NMSE, at least for first predicted values. It can fit the MECoM curves at low SNR and is more efficient than the other predictor at the high SNR of 100 dB.

So from now on, we will only use the first eigenvector of the Hankel matrix to compute signal poles. It implies MEHaM will only consider one signal pole, and therefore only one corresponding amplitude. So the MEHaM computations become highly simplified.

Here are now some results comparing MECoM and the enhanced MEHaM technique.

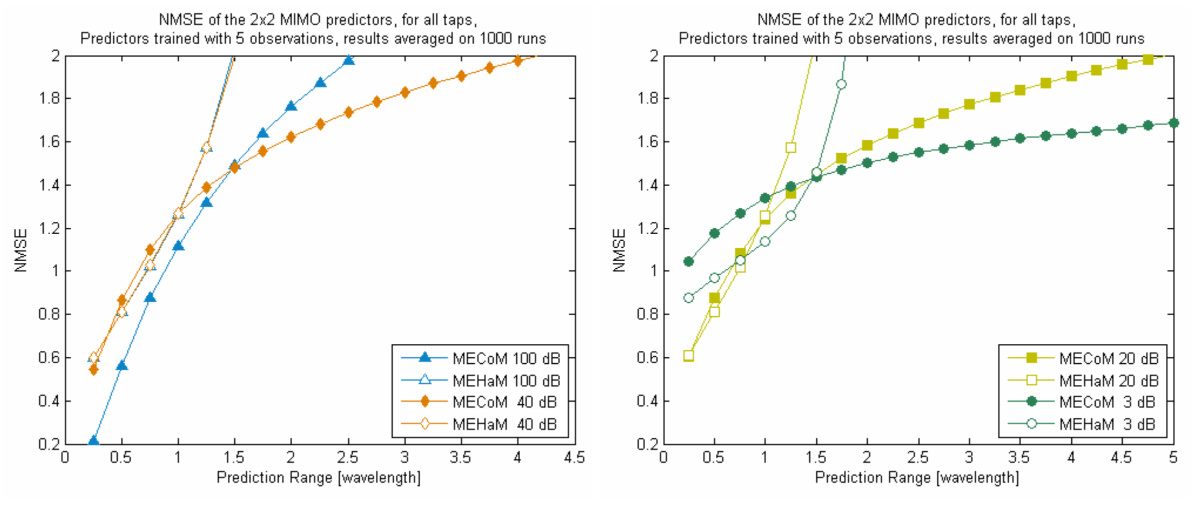


Figure 9-13 : Comparison between MECoM and MEHaM, influence of the limited number of columns on MEHaM, at several noise levels (*E*=5)

First, one may notice that MEHaM produce the same NMSE curves for SNR = {20, 40, 100} dB.

Then except at the unrealistic noise level of 100 dB, MEHaM, using only the first eigenvector, produce the same NMSE as MECoM on the first wavelength, and is even more efficient than MECoM at the low SNR of 3 dB. It is a main achievement point because using only 1 eigenvector really simplifies the computations, with no performance degradation regarding MECoM. And moreover, MEHaM does not need any knowledge about noise variability. On drawback, prediction error is dramatically increasing after a wavelength, whereas MECoM's error does not increase so much. A refresh of the predictor should be considered.

Whatever, from now on, when we will use the MEHaM predictor, we will always restrict the technique to consider only the first eigenvector of the Hankel matrix of 2x2 MIMO channels. We will now look at the performance of the predictors in 4x4 MIMO scenarios.

9.5 Testing 4x4 MIMO scenarios

As it was mentioned before, the predictors are not restricted to 2x2 MIMO systems. This subsection will therefore present some 4x4 MIMO results. We will begin with noise free computation, then we will add some Gaussian noise.

9.5.1 Noise free conditions on 4x4 MIMO

Because a 4x4 MIMO channel embeds 16 antenna pairs, we need at least 17 channel observations to maintain the number of rows of the Hankel matrix larger than its number of columns. Therefore we tested the predictors when trained with 17 observations, and with 33 for the comparison. **Figure 9-14** shows these results.

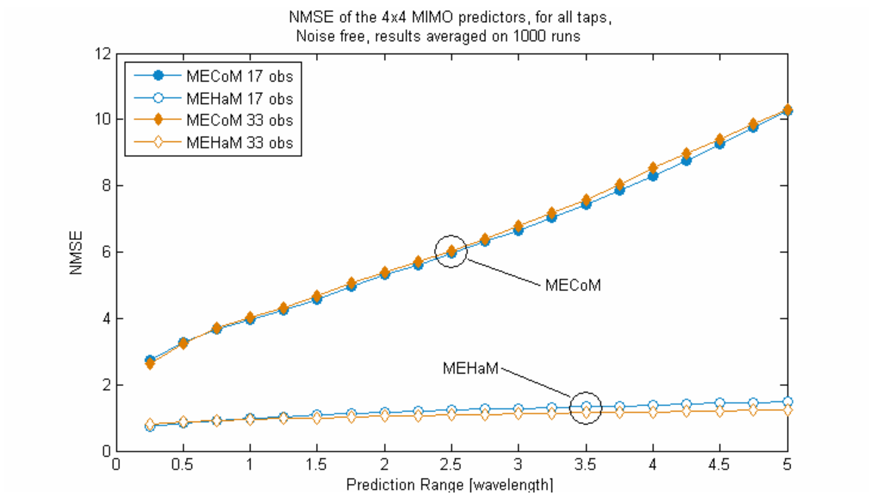


Figure 9-14 : Time evolution of the NMSE for the 4x4 MIMO predictors, in noise free conditions (*E* = 17 and 33)

We still have some problems with MECoM in noise free conditions. As we will see in the following subsection, with added noise, performance will be improved. One can note that the number of observations used to train the predictors only have limited effect.

9.5.2 Adding coloured noise

Before trying to enhance the MEHaM for 4x4 MIMO channels as it has been done for 2x2 MIMO channels in 9.4.4 , we will present some comparative results of the two predictors. So all eigenvectors have been considered to set up the MEHaM predictor. **Figure 9-15** shows these results, with predictors trained with 17 MIMO channel observations.

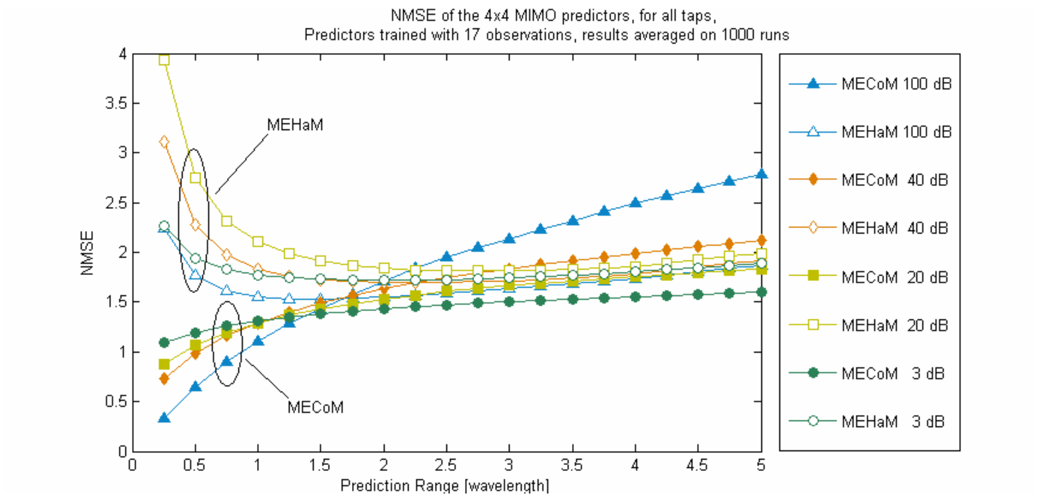


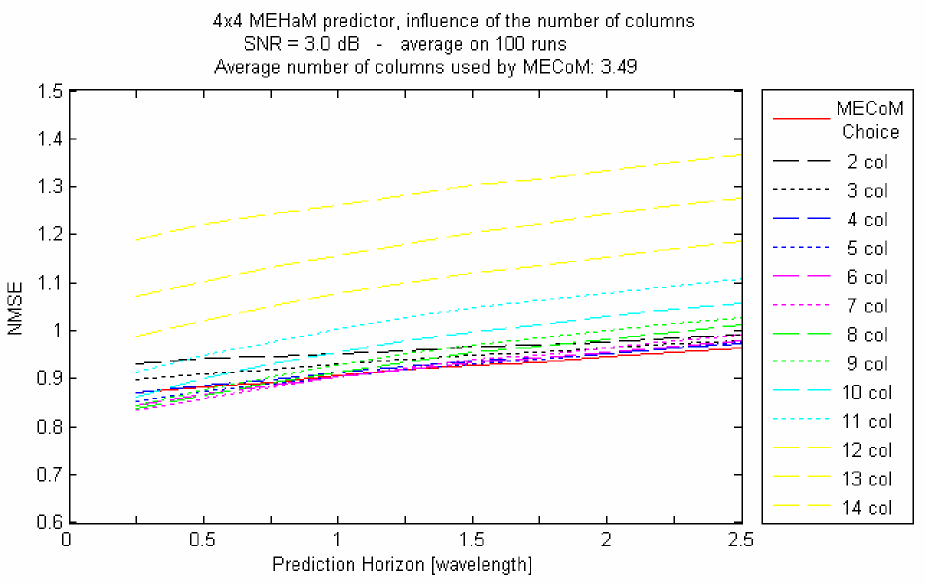
Figure 9-15 : Time evolution of the NMSE for the 4x4 MIMO predictors, no limitation on the number of columns for MEHaM, with additive Gaussian noise (*E* = 17)

As one can see, adding noise, even at the high SNR of 100 dB really improves the prediction quality of MECoM. But on drawback, first predicted values of MEHaM contain much instability. This is maybe because all the eigenvectors have been considered. As in the 2x2 MIMO scenarios, we should introduce a limitation.

The following subsection will help to decide the choice of the eigenvectors.

9.5.3 Improving MEHaM in 4x4 scenarios

As we did for 2x2 MIMO scenarios in section 9.4.4, we will now try each possible value of N_S for MEHaM in 4x4 MIMO scenarios. **Figure 9-16** and **Figure 9-17** show the results of these computations. NMSE attained by MEHaM using only 1 column or more than 14 are so bad that they are not plotted on the figures, to keep a narrow scale.



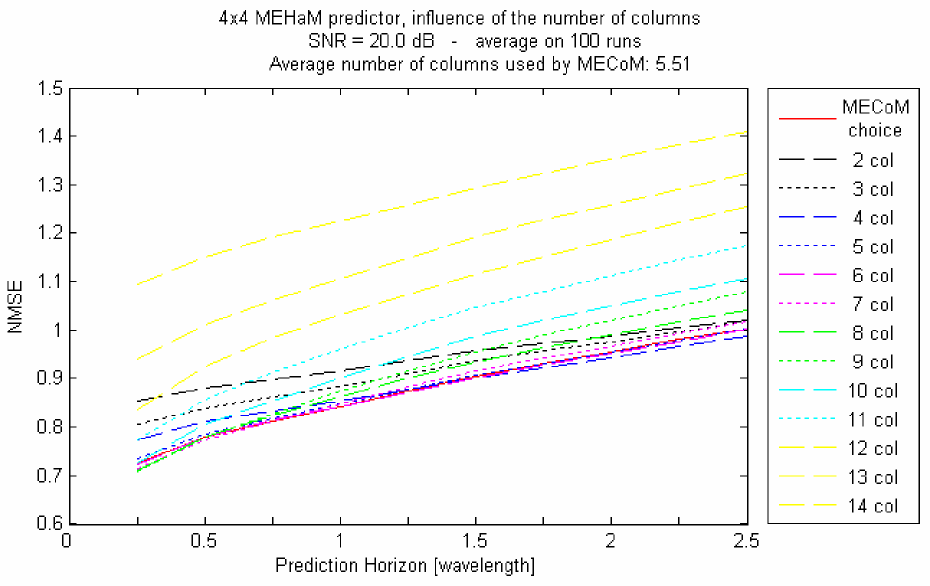
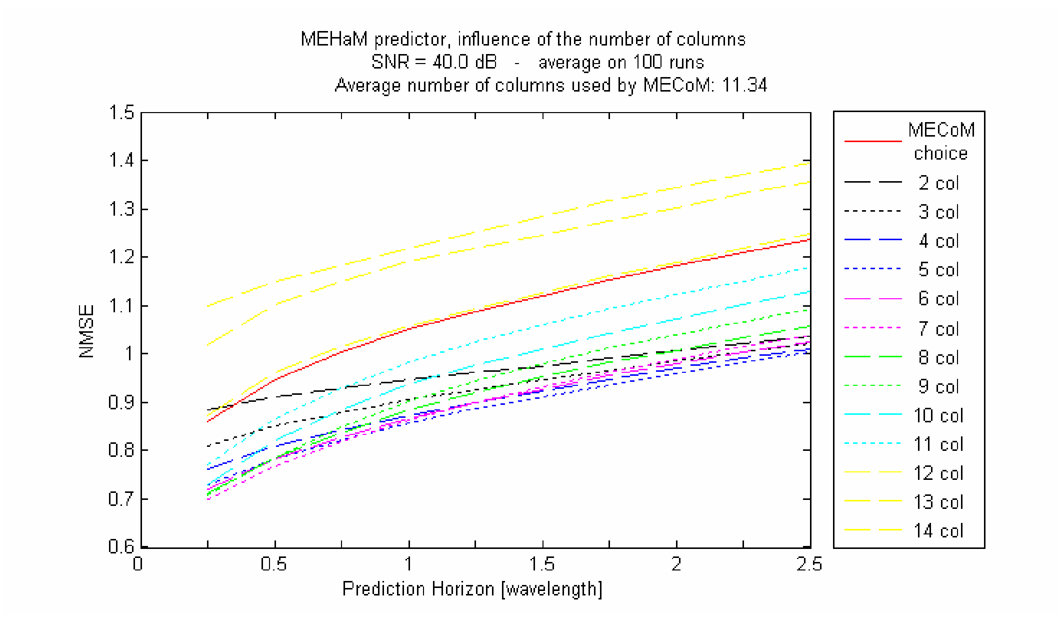


Figure 9-16 : Influence of the number of columns on the 4x4 MEHaM predictor (17 obs, SNR = 3, 20dB)



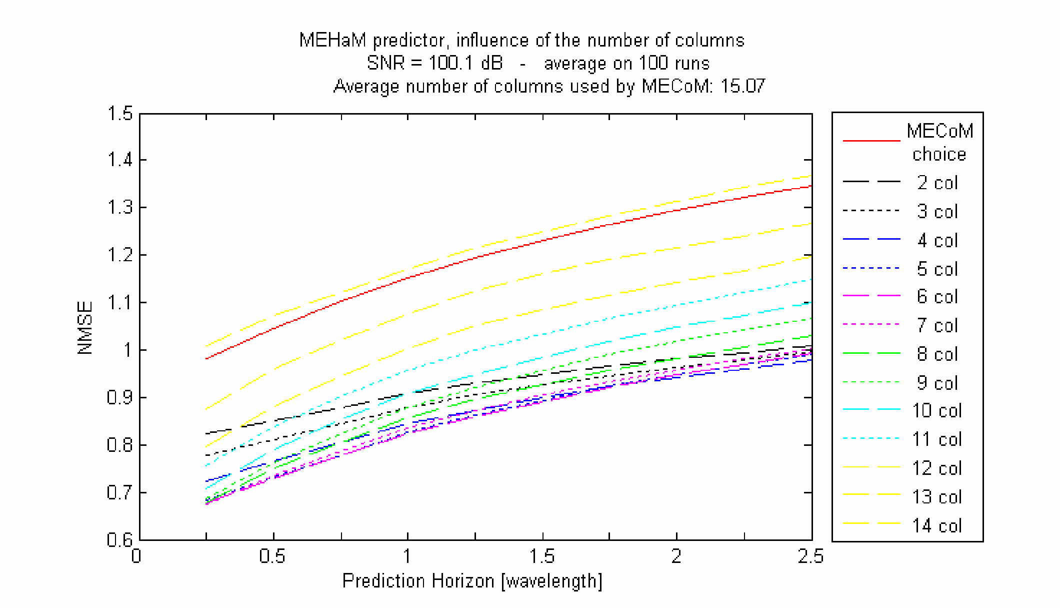


Figure 9-17 : Influence of the number of columns on the 4x4 MEHaM predictor (17 obs, SNR = 40, 100dB)

The red curves show the performances of MECoM, using the noise variance to decide the number of columns. As one can see on **Figure 9-16**, by choosing a value in between 4 and 7, MEHaM can reach the performance of MECoM. But with a higher SNR, as it is shown on **Figure 9-17**, by choosing an efficient value, MEHaM can behave really better.

To limit the size of handled matrices and the computational load of the predictor, using 5 columns seems us to be the best trade off. From now on, in 4x4 MIMO scenarios, we will use MEHaM trained from 17 MIMO channel samples, and poles will only be computed from 8 eigenvectors of the Hankel matrix of the channel.

Here are now on **Figure 9-18** some comparative results of the 4x4 MIMO predictors.

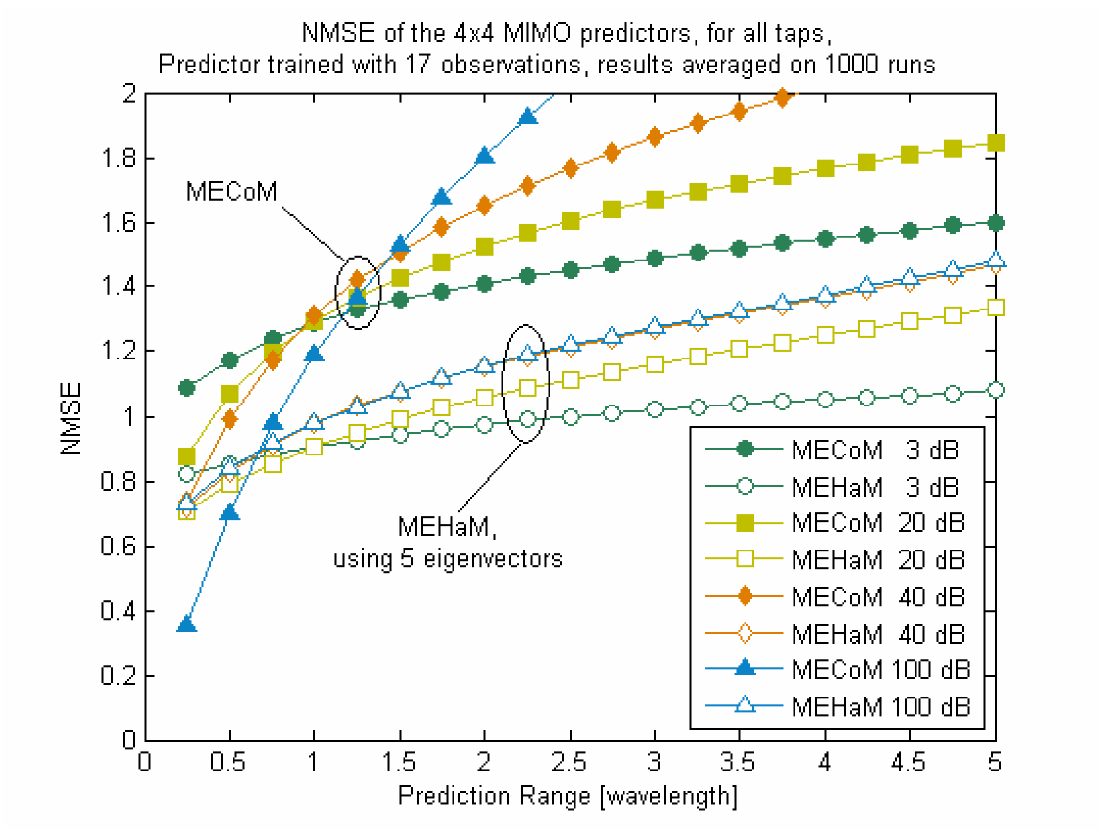


Figure 9-18 : Comparison between MECoM and MEHaM in 4x4 MIMO scenario, influence of the limited number of columns on MEHaM, at several noise levels (*E*=17)

The limitation we added on the number of eigenvectors used in the poles' computation really improved the performances of the predictors. The instability of the first predictions on **Figure 9-15** completely disappeared, and MEHaM produces better results than MECoM (except in the unrealistic case of SNR = 100 dB), in an easier way, which does not require noise estimation.

9.6 Conclusions about MIMO computations

During all our MIMO tests, we found first that the number of channel observations used to train the predictors has only limited effect. For that reason, choosing the smallest number is the more efficient, because it deals with smaller matrices and warm-up time is shortened.

We have also shown that the noise free conditions generate troubles of singularity and so on, but adding a slight noise, with a SNR of 100 dB, prevent these troubles. Then in noisy conditions, we showed that MECoM was producing better results than MEHaM. Actually, MECoM is considering the noise level to decide how many eigenvectors it will use, whereas MEHaM was first using all of them. We therefore restricted MEHaM to use only the first one in 2x2 scenarios, and only the 5 first ones in 4x4 scenarios.

With that enhancement of MEHaM, we found the MEHaM as good as MECoM, or sometimes better. Moreover, MEHaM does not require noise estimation.

Figure 9-19 illustrate the final results of MECoM and MEHaM, at the realistic SNR's of 3 and 20 dB.

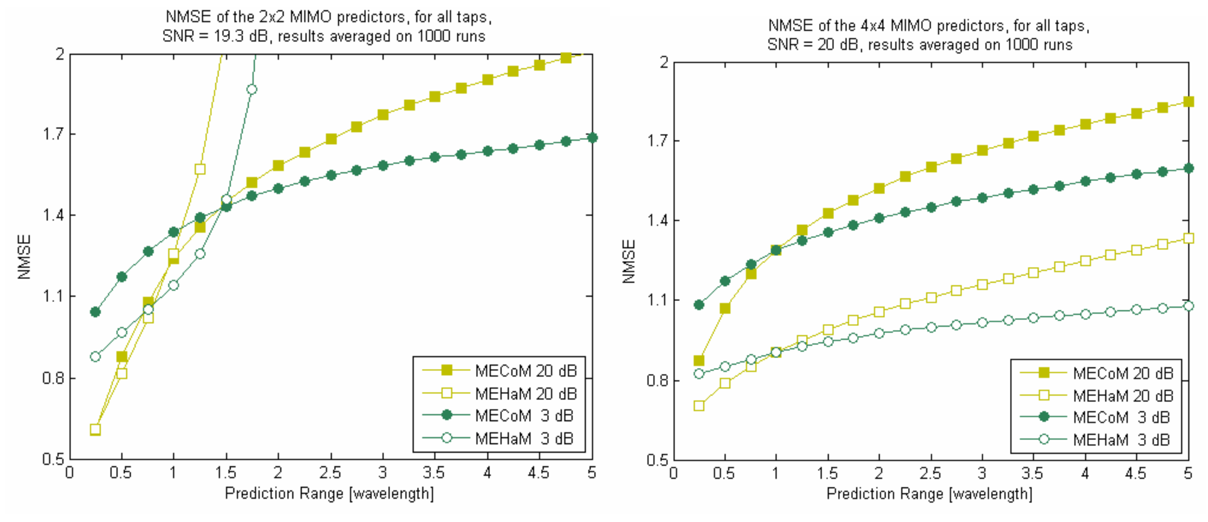


Figure 9-19 : Reliability of MIMO ESPRIT (SNR = 20 dB and 3 dB)

The 2x2 MEHaM has its NMSE growing dramatically fast compared to MECoM's. But on the first wavelength, results remain good. And the computational load of MEHaM is lighter than the MECoM one. A refresh step should be considered.

On 4x4 scenarios, NMSE curves are flatter. This is probably because more information is taken into account for 4x4 scenarios. (16 embedded SISO, instead of 4 for the 2x2 case). Moreover, MEHaM curves are largely lower to the MECoM ones.

We will now try to find how our predictors behave when feedback channel has been quantized on a few bits.

10 WORKING WITH QUANTIZED VALUES

From the beginning, we always used to 64 bits default resolution of Matlab. If a computer can perform computations on 64 bits, this is not the case of small handsets. Moreover, when considering the feedback load, we can not afford transferring all coefficients at full number resolution. We should preserve the bandwidth for the user's applications.

This section will therefore investigate the influence of quantization of the reference channel on the predictors' performance.

10.1 Quantization law

We will try to reduce the feedback information to only 3 or 6 bits for each part of the complex MIMO coefficients. So to feedback MIMO channel information about one time step, we need 2 $n_T\, n_R\;\; N_{\text{bits}}$ bits, where $N_{\text{bits}} = \{3\ ,\ 6\}$.

Therefore, to train a 2x2 MIMO predictor using 3 bits, we need 120 bits, and using 6 bits, we need 240 bits. And for a 4x4 MIMO predictor, using 3 bits requires 816 bits of feedback, and using 6 bits requires 1632 bits.

Figure 10-1 presents the quantization law we used. It is symmetrical, since we round real values to the nearest quantized ones.

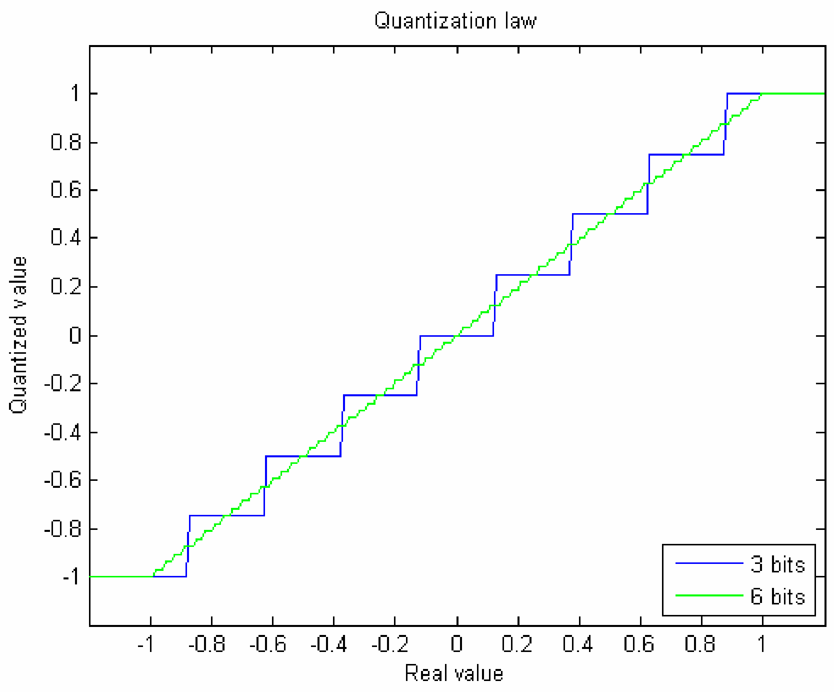


Figure 10-1 : Quantization law, using 3 or 6 bits

We will now see how the predictor behaves when trained with limited feedback quantization.

10.2 Influence on 2x2 MIMO predictors

This section show performance of the two predictors when trained with low bit quantization feedback. These results are compared with full feedback results.

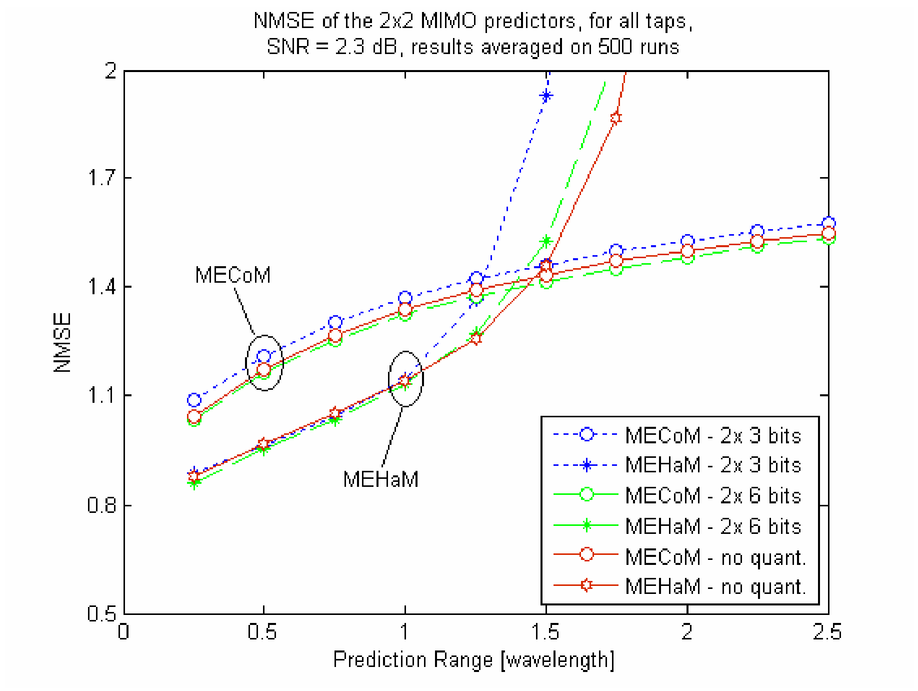


Figure 10-2 : Influence of quantization on 2x2 MIMO channel, SNR = 3 dB

Figure 10-2 presents these results with SNR = 3 dB. One can see that quantization has barely no influence at that low SNR.

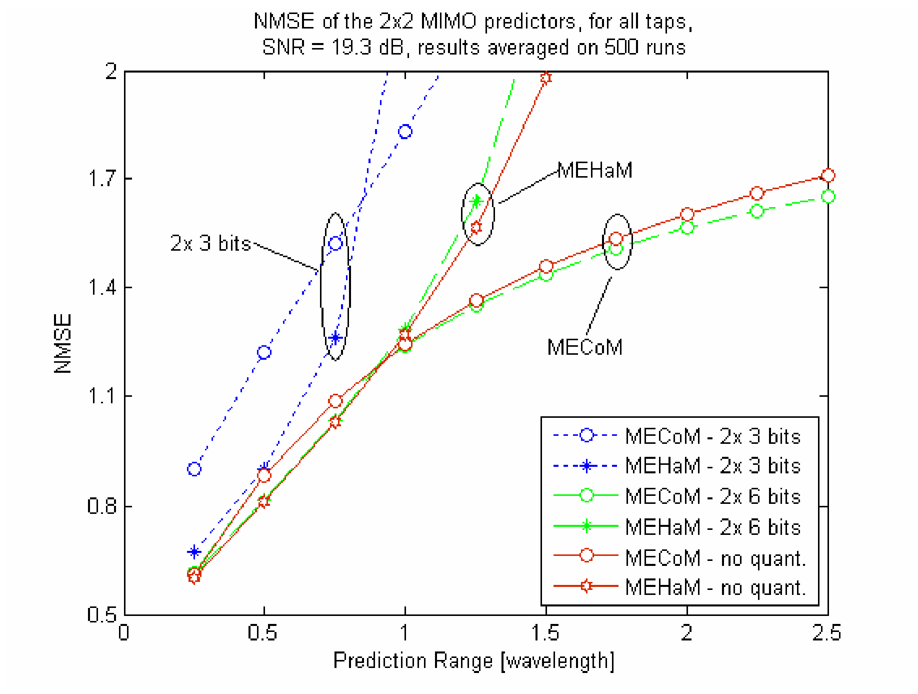


Figure 10-3 : Influence of quantization on 2x2 MIMO channel, SNR = 20 dB

Figure 10-3 shows the same kind of results, but with a SNR of 20 dB. With a so weak noise level, quantizing feedback with only 2x 3 bits is insufficient. But using 2x 6 bits produces the same results as using the full resolution.

The problem of using only 2x 3 bits is even more critical at the high SNR of 100 dB, because using to few bits leads to singular matrices when computing the poles with MECoM, and therefore put a stop to their computation. This is the reason why there is no curve using 2x 3 bits for MECoM on **Figure 10-4**. The MEHaM performances are a bit degraded by the 2x 3 bits quantization. Besides, that figure shows quantization on 2x 6 bits has still no influence on MEHaM, but decrease the MECoM's performance.

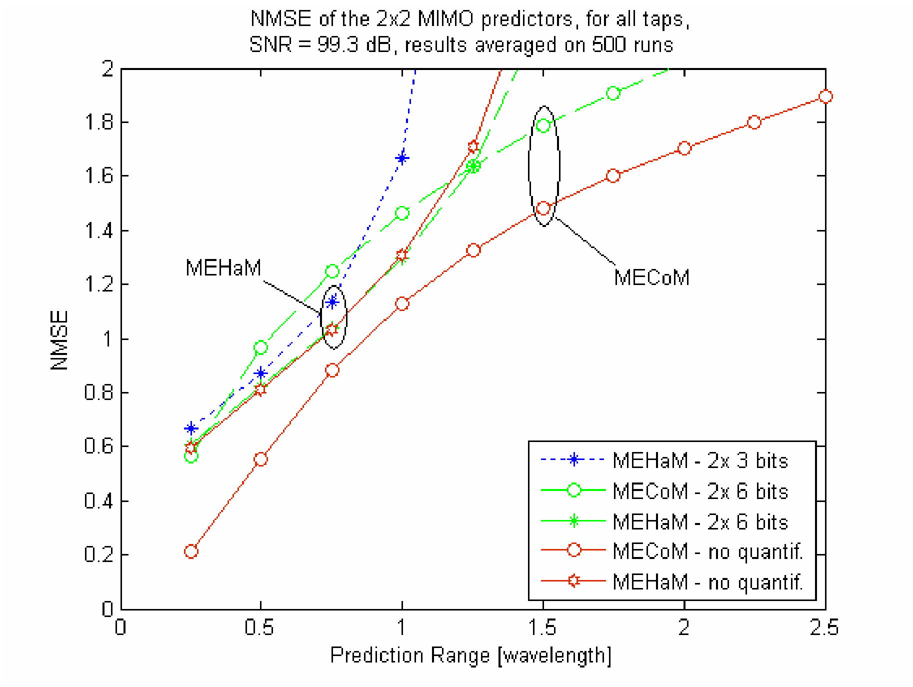


Figure 10-4 : Influence of quantization on 2x2 MIMO channel, SNR = 100 dB

Therefore, if we neglect the unrealistic 100 dB curve of MECoM, we can state that we can quantize our feedback information with 2x 6 bits per complex MIMO coefficient without losing quality. It means that we require 240 bits of feedback to train the predictors, for being able to perform predictions.

10.3 Influence on 4x4 MIMO predictors

In a same way, we will now present how 4x4 MIMO predictors behave with limited amount of feedback. **Figure 10-5** present these results with a SNR of 3 dB, and then, SNR's of 20 and 100 dB are considered.

At 3 dB, the shape of the MECoM curves is the same for both 2x2 and 4x4 scenarios, but MEHaM ones are different: there is less instability in 4x4 MIMO MEHaM predictions. MEHaM produces far better predictions than MECoM. And as in 2x2 scenarios, quantize on 2x 3 bits, 2x 6 bits, or consider full resolution leads to the same results, for each predictor. At that low SNR, they are both really robust against quantization.

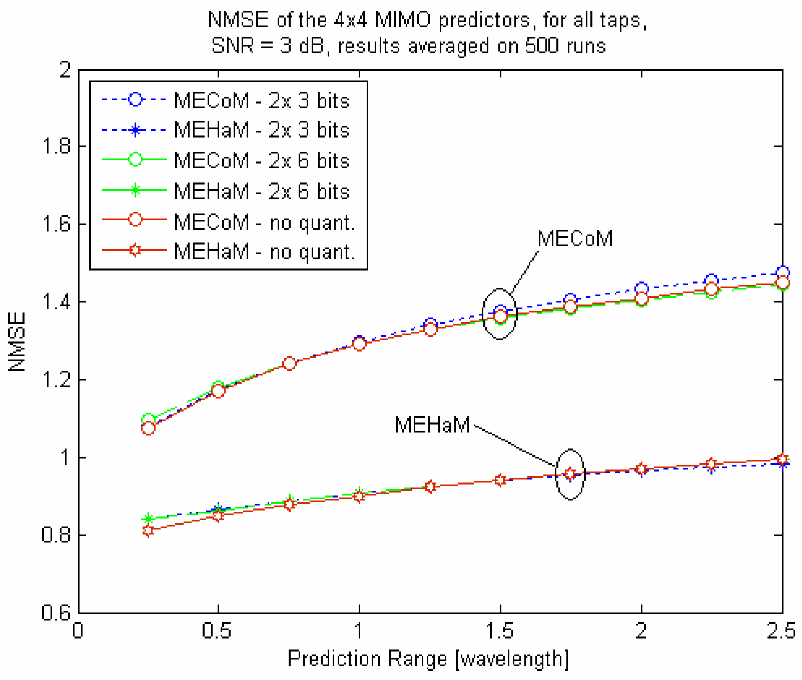


Figure 10-5 : Influence of quantization on 4x4 MIMO channel, SNR = 3 dB

When decreasing the noise to get a SNR of 20 dB, a difference appears with quantization on only 2x 3 bits: MEHaM predictions are a bit better, but MECoM ones become poor. Moreover, as in 2x2 scenarios, we get that quantized curves with 2x 6 bits can match really well curves obtained with full resolution.

These results are presented in **Figure 10-6**

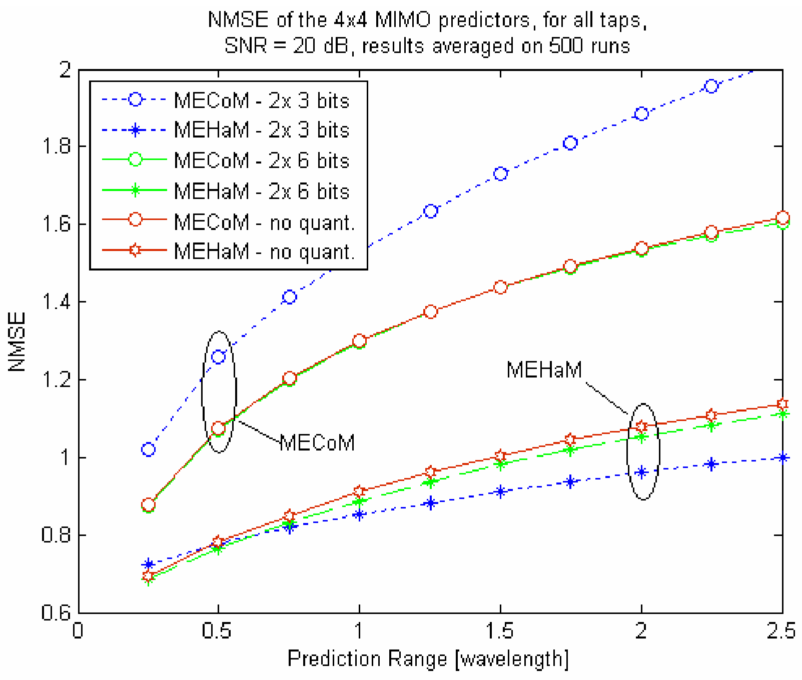


Figure 10-6 : Influence of quantization on 4x4 MIMO channel, SNR = 20 dB

Finally we move to the 100 dB simulations. As in the 2x2 case, singularity troubles happen with MECoM when the noise is too low, and its effect disappears in the rounding operation of the quantization step. Using 2x 6 bits prevent these troubles. This is the reason why **Figure 10-7** does not show 2x 3 bits results for MECoM.

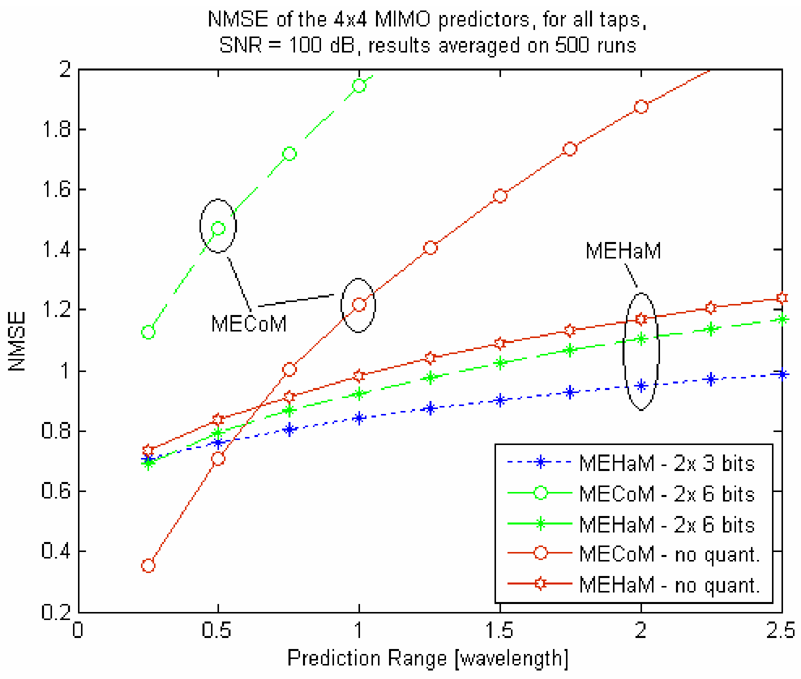


Figure 10-7 : Influence of quantization on 4x4 MIMO channel, SNR = 100 dB

At that high SNR, MEHaM is still not really affected by the quantization. On long term predictions, using only 2x 3 bits even produces better results. As far as MECoM is concerned, when there is no quantization, it is better than MEHaM for half a wavelength, but when there is quantization, MEHaM is always the best.

10.4 Summary results

From all these simulations, one can state that quantize the feedback using 2x 6 bits does not decrease the accuracy of predictions. The NMSE curves are the same when quantized on 2x 6 bits and when not quantized, for each predictor and for every SNR. Using fewer bits for quantization sometimes also leads to good results, but sometimes leads to instability at high SNR's, and even to singularity troubles with MECoM.

11 AN INOVATIVE MIMO CHANNEL PREDICTOR

The two predictors have globally the same working scheme. They need a matrix containing the channel information, then they look for signal poles, and their corresponding amplitude. But they use different techniques to perform the same thing. From all the tests presented before, we are now able to merge the two MIMO predictors into a single one, using best of both predictors.

11.1 Working scheme of the predictors

Here is first the comparative algorithm of the two predictors. It highlights the differences between them.

MECoM	MEHaM
Using correlation matrix, estimated by a normalized cross product of a Hankel matrix	Using straightly the Hankel matrix
Choosing the number of eigenvectors of the correlation taken into account with respect to the noise level (which has therefore to be estimated)	Choosing the number of eigenvectors taken into account from past simulations. That value depends only on the number of antennas.
Computing the $\mathcal{D}$ matrix from the eigenvectors	
The signal poles are the normalized eigenvalues of $\mathcal{D}$	The signal poles are the eigenvalues of $\mathcal{D}$ smaller than 1.05
Computing the corresponding eigenvalues, by solving a linear system for each antenna pair	

The following subsections will discuss the choice between MECoM's and MEHaM's strategies.

11.2 Choice of the matrix containing channel information

First of all, there is the choice of the matrix containing the channel observations. MECoM uses the correlation matrix of the channel, estimated by the cross-product of a Hankel matrix. In section 6.2, we demonstrated that using the correlation matrix or the straightly the Hankel matrix leads to the same eigenvectors.

Therefore, it is faster and less power demanding to use the Hankel matrix, to avoid the cross product of a complex ($N_T N_R$ +1) x ($N_T N_R$) matrix.

We have also shown we can limit the number of MIMO channel observations to the minimum, with the only imperative to keep more rows than columns in the Hankel matrix. So it means that the Hankel matrix should contain ($N_T N_R$ +1) MIMO channel samples.

Here is how the Hankel matrix should look like:

$$\underset{(N_T \cdot N_R+1 \times N_T \cdot N_R)}{\mathcal{H}} = \begin{bmatrix} h^{11}(1) & \cdots & h^{1N_R}(1) & \cdots & h^{N_T N_R-1}(1) & \cdots & h^{N_T N_R}(1) \\ h^{11}(2) & \cdots & h^{1N_R}(2) & \cdots & h^{N_T N_R-1}(2) & \cdots & h^{N_T N_R}(2) \\ \vdots & & \vdots & & \vdots & & \vdots \\ h^{11}(N_T \cdot N_R+1) & \cdots & h^{1N_R}(N_T \cdot N_R+1) & \cdots & h^{N_T N_R-1}(N_T \cdot N_R+1) & \cdots & h^{N_T N_R}(N_T \cdot N_R+1) \end{bmatrix}$$

Then we compute its SVD and take the eigenvectors to compute the signal poles. There are two options to decide how many eigenvectors should we use. Then we can filter and/or normalize these poles. The next subsection discusses these points.

11.3 Computation of the poles

The number of eigenvectors involved in the computations for the poles should actually reflect the dimension of the signal subspace. The MECoM strategy comes from the original ESPRIT technique. It estimates the dimension of the signal subspace by the number of eigenvalues of the correlation matrix which are larger than the variance of the noise. So the noise has to be estimated.

The strategy we adopted with MEHaM is to perform Monte-Carlos simulations to decide what fixed values are optimal. It has been shown that using only the first eigenvector produces good results for 2x2 MIMO scenarios, and using the first five eigenvectors is optimal for 4x4 MIMO scenarios.

Then we can normalize the poles, to set their modulus to 1. It means only the arguments have to be recorded, but they have to be computed. The original ESPRIT technique was normalizing the poles, and so the MECoM technique too. Andersen et al. proposed instead normalization to discard poles with modulus larger than 1.05, and so MEHaM does.

So there are three choices to make:

1. Use a dynamical amount of eigenvectors or use static values;
2. Normalize the poles or not;
3. Discard the largest poles or not.

We ran several simulations to decide the optimal choices to do. **Figure 11-1** shows these results. Predictors were based on Hankel matrices as it has been discussed in section 11.2. As one can see, using a fixed number of columns produces slightly better results than the variable value during more than a wavelength. Excepted when there is normalization and filtering of the poles, but these curves are really bad and these solutions will no be chosen.

When there is no filtering of the poles, normalization has no influence, and it produces the best results.

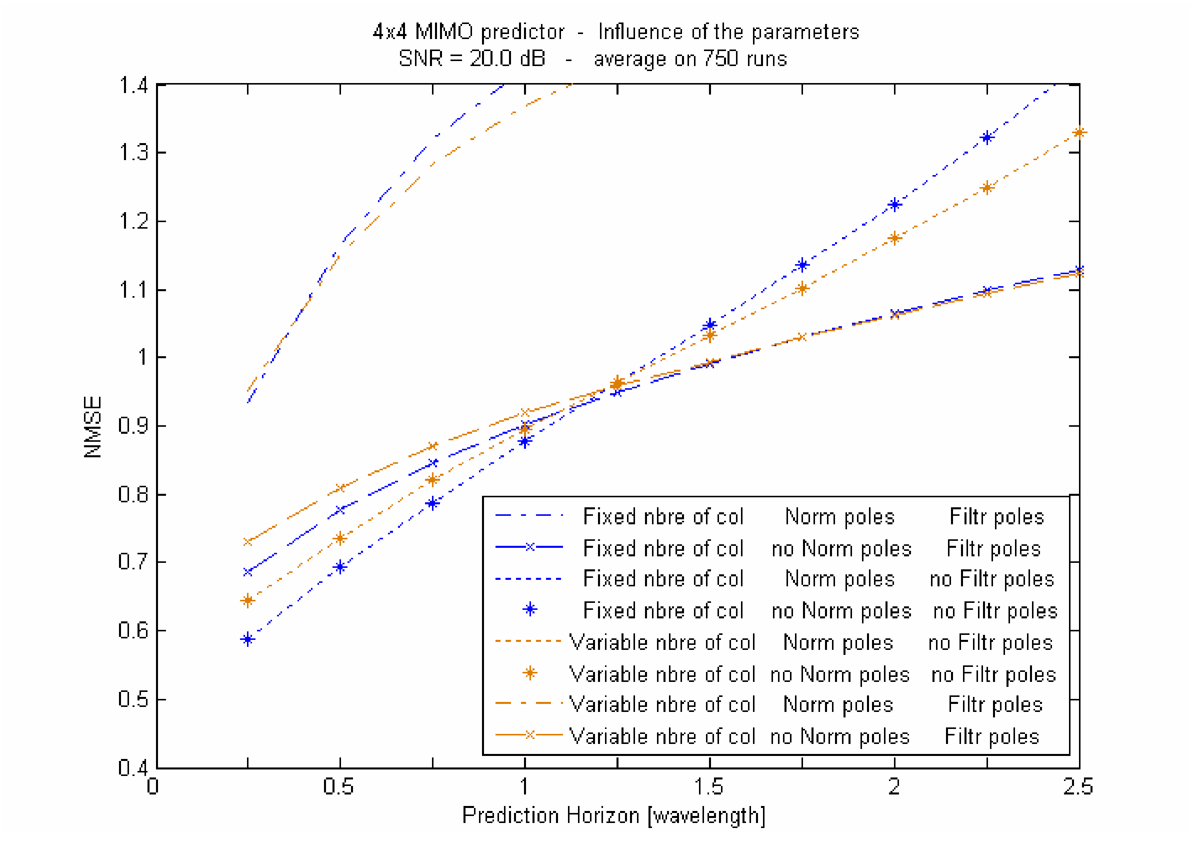


Figure 11-1 : Computation of the poles, influence of the parameters (4x4 MIMO, SNR = 20 dB, 750 runs)

In conclusion, it is first better to use fixed amount of eigenvectors. Results are slightly better and moreover, it does not need noise variability estimation. Then it is more efficient to consider all the eigenvalues, without any filtering. Finally, if there is no filtering, the normalization step has no influence. We can then ignore it to avoid additional computations, or we can perform it to simplify the poles in case of their transmission.

For the rest of this document, we will not consider poles' normalization anymore. The rest of the MECoM and MEHaM technique, i.e. the amplitude computations for each antenna pair, and the prediction scheme are the same, and so the new MIMO predictor will do.

Before presenting the performances of this new MIMO predictor, we will give its summary algorithm.

11.4 Summary Algorithm

Here is the summary of this new MIMO predictor.

1. Compute the SVD of the $\mathcal{H}$ matrix, where $\mathcal{H}$ is defined as in section 11.2. So $\mathcal{H} = U \cdot \Sigma \cdot V$.

2. Take the first $(N_s + 1)$ columns of the U matrix to build the U_1 matrix, where $(N_s + 1)$ worth 1 in case of 2x2 MIMO, or 5 when 4x4 MIMO.

3. Compute the $\mathcal{D}$ matrix such as $J_{UP} \cdot U_1 \cdot \mathcal{D} \approx J_{DOWN} \cdot U_1$.

4. Therefore, the poles are the eigenvalues of $\mathcal{D}$.

5. Then compute the amplitudes: $a = (Z^* Z)^{-1} \cdot Z^* \cdot Y$

Finally, the predictions are given by: $\forall t = E, E+1, \ldots \quad \hat{h}^{mn}(t) = \sum_{k=1}^{N_s+1} a_k^{mn} z_k^{t}$.

We will now end this section by presenting the numerical complexity and the performance of the new MIMO predictor.

11.5 Numerical complexity of the new MIMO predictor

Because the new predictor is based on the two others, it also requires 3 SVD of matrices with size n_T x n_R, which can be done in $O(\min\{n_T n_R^2, n_T^2 n_R\})$.The other computations to get the predicting parameters can be done in $O(n_T^3 n_R^3)$. The noise variability is not required, and the choice of the number of eigenvectors is drawn from static values, which makes the new predictor faster.

Once the signal poles and the amplitudes are known, predicting the channel state for the next time step can still be done in $O(n_T n_R N_S)$.

11.6 Performances of the new MIMO predictor

This section will present the effect of using a limited amount of bits to quantize the feedback. We tested from 2x 3 bits to 2x 6 bits per complex MIMO coefficient, at several noise levels. The quantization law we used is the one described in section 10.1. **Figure 11-2**, **Figure 11-3** and **Figure 11-4** compare these results to MIMO channel predictions based on full number resolution feedback.

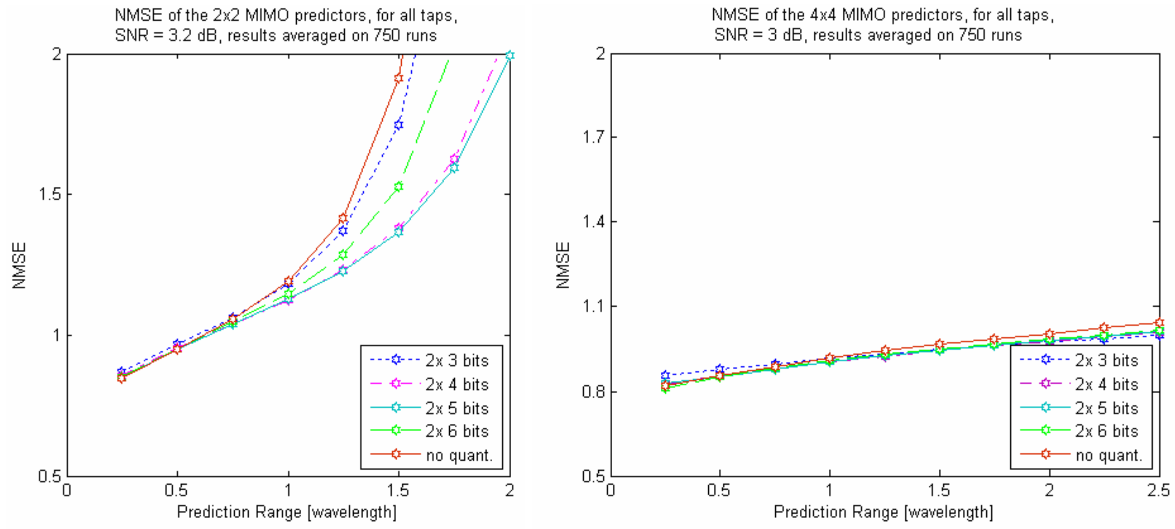


Figure 11-2 : Effects of quantization on 2x2 and 4x4 MIMO channels with SNR = 3 dB

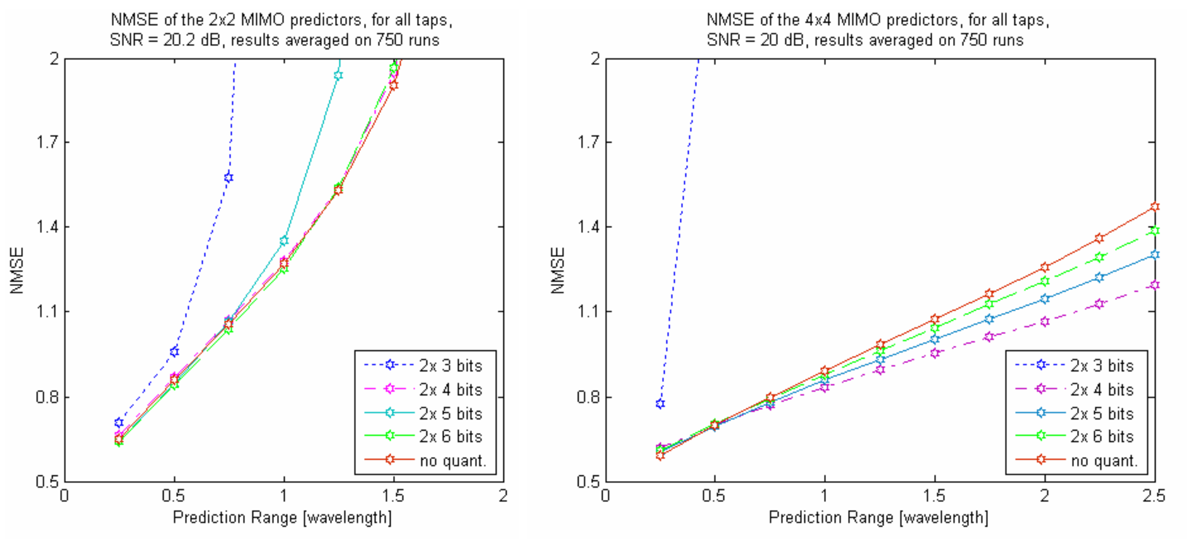


Figure 11-3 : Effects of quantization on 2x2 and 4x4 MIMO channels with SNR = 20 dB

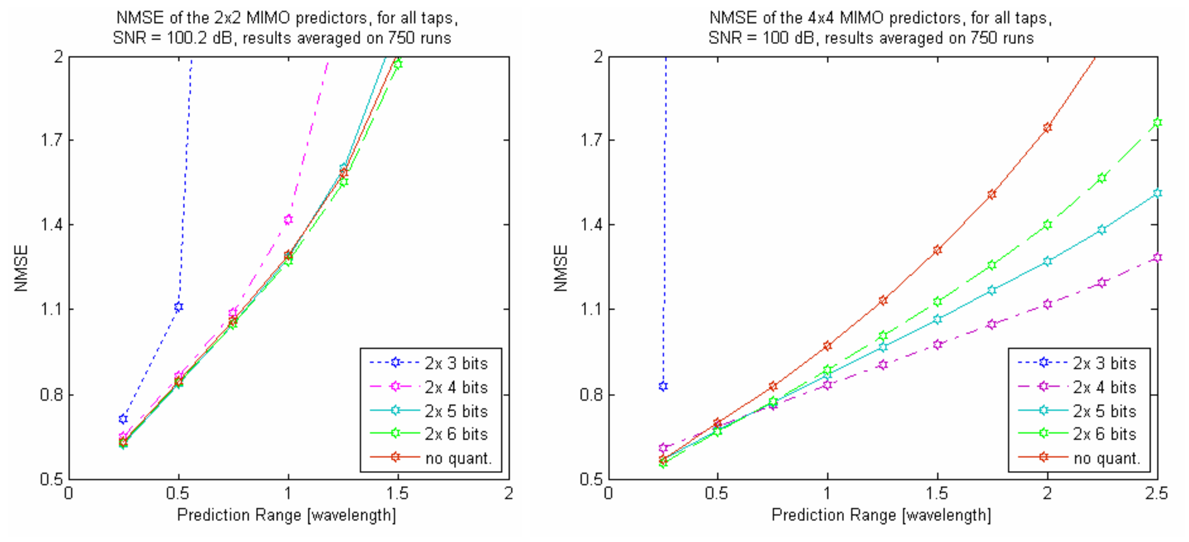


Figure 11-4 : Effects of quantization on 2x2 and 4x4 MIMO channels with SNR = 100 dB

Except at the poor SNR of 3dB, where all quantization levels lead to the same results, using only 2x 3 bits produces really poor results. Actually, using too few bits causes instability of the predictions, especially in 4x4 scenarios. Moreover, with only 2x 3 bits, the higher the SNR is, and the poorer the results are. This may come from the fact that using too few bits causes badly scaled matrices. And the more there is noise, the more the channel is shuffled, and then the less matrices are badly scaled.

On the contrary, when using more bits, predictions are better with SNR at 20 or 100 dB than at 3 dB, which seems logical. In addition, curves at 20 dB and 100 dB are the same. This is may be because at high SNR, the effect of the noise is hidden by the rounding operation of the quantization.

Another thing to point out is that using 2x 4 bits to quantize each complex MIMO coefficient leads barely to the same results as working with 2x 5 and 2x 6 bits, which are close to full number resolution results. So we can afford quantization step, even using only 2x 4 bits per MIMO coefficient, to keep light the feedback load.

One can also drawn attention to the fact that long-term predictions are really better for 4x4 than 2x2 MIMO scenarios. Do not forget that 4x4 predictions are based on much more channel observations (17 time samples of 16 MIMO coefficients) than 2x2 predictions (5 time samples of 4 MIMO coefficients). This can explain why there is more instability embedded in 2x2 predictions than 4x4 ones.

Finally, we should also mention that this new MIMO predictor produces slightly the same results as MEHaM, since it is mainly based on its strategies. Results of MEHaM are just a bit better on 4x4 scenarios, only after 1.5 wavelengths. It comes from the only difference between the two techniques: with MEHaM, we discard the biggest poles (with modulus larger than 1.05), and with the new predictor we do not. Therefore, the long term predictions of MEHaM are more preserved form instability than the ones of the new predictor.

We have also considered another quantization law, using floating point technique. We tested it with 2x 6 bits for each complex MIMO coefficient. It means for each real value, 1 bit is for the sign, 2 for the exponent, and 3 for the mantissa. **Figure 11-5** presents this quantization law. The main advantage of this law is that it is more accurate around zero, and therefore it avoids rounding to zero and the ranks of the matrices are maintained.

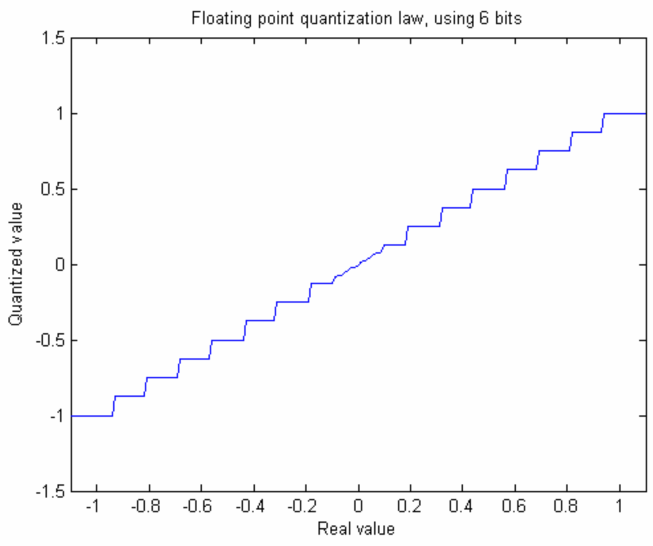


Figure 11-5 : Floating point quantization law, using 6 bits

Figure 11-6, **Figure 11-7** and **Figure 11-8** compare the two quantization laws. The floating point was used with 2x 6 bits, and the fixed point with 2x 4 bits since we have shown it was sufficient. As one can see, results are globally the same, and using more bits with the floating point quantization does not really improve predictions. Therefore it is more efficient to consider fixed point values, with only 2x 4 bits per MIMO channel coefficient.

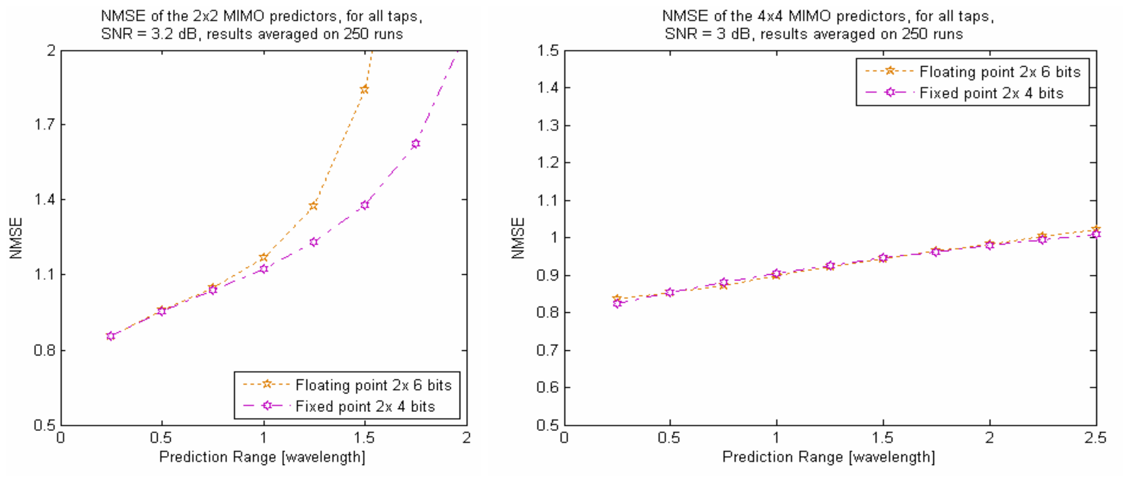


Figure 11-6 : Effects of floating point quantization on 2x2 and 4x4 MIMO channels with SNR = 3 dB

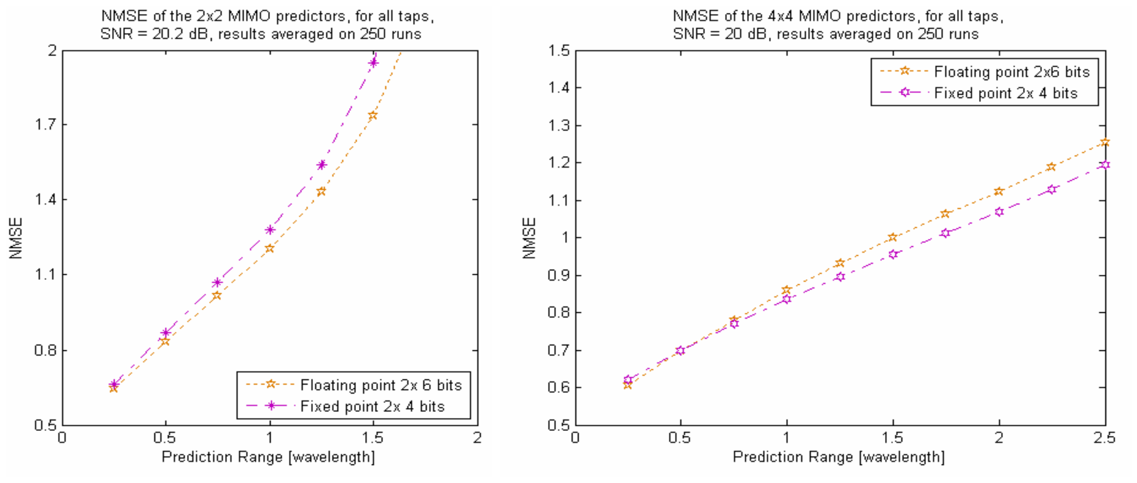


Figure 11-7 : Effects of floating point quantization on 2x2 and 4x4 MIMO channels with SNR = 20 dB

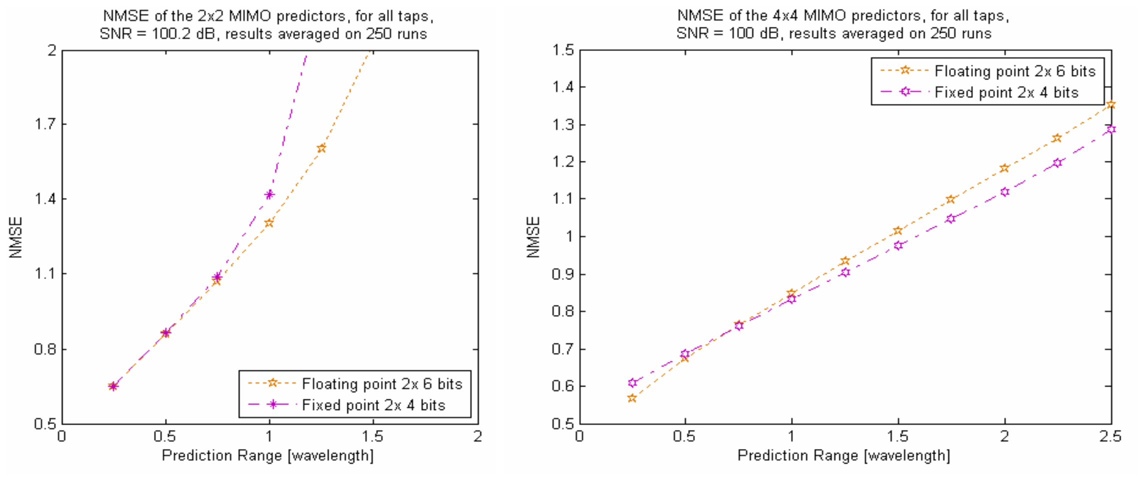


Figure 11-8 : Effects of floating point quantization on 2x2 and 4x4 MIMO channels with SNR = 100 dB

We will now discuss another topic. Helped by the new MIMO predictor, we will try to predict the rank evolution of MIMO channel matrices.

12 PREDICTIONS ON THE RANK OF THE MIMO CHANNEL MATRIX

This chapter will tackle another issue. We will use our new MIMO predictor to predict the time evolution of the rank of the MIMO channel. We will aim at predicting the complex MIMO coefficients of the channel, quantizing them using the fixed point quantization law presented in Section 10.1, computing the rank of that predicted matrix, and comparing it with the matrix of the reference MIMO channel, provided by SCME.

Actually, the following figures will show, for each possible rank, at a given time step t_0, the fraction of channel matrices keeping the same rank value from the first time step up to the t_0^{th}.

12.1 2x2 MIMO scenarios

This first subsection will consider 2x2 MIMO scenarios. We will next move to 4x4 ones. **Figure 12-1** presents these results with SNR = 20 dB, when reference and predicted matrices have been quantized on 2x 6 bits, as it has been defined in Section 10.1.

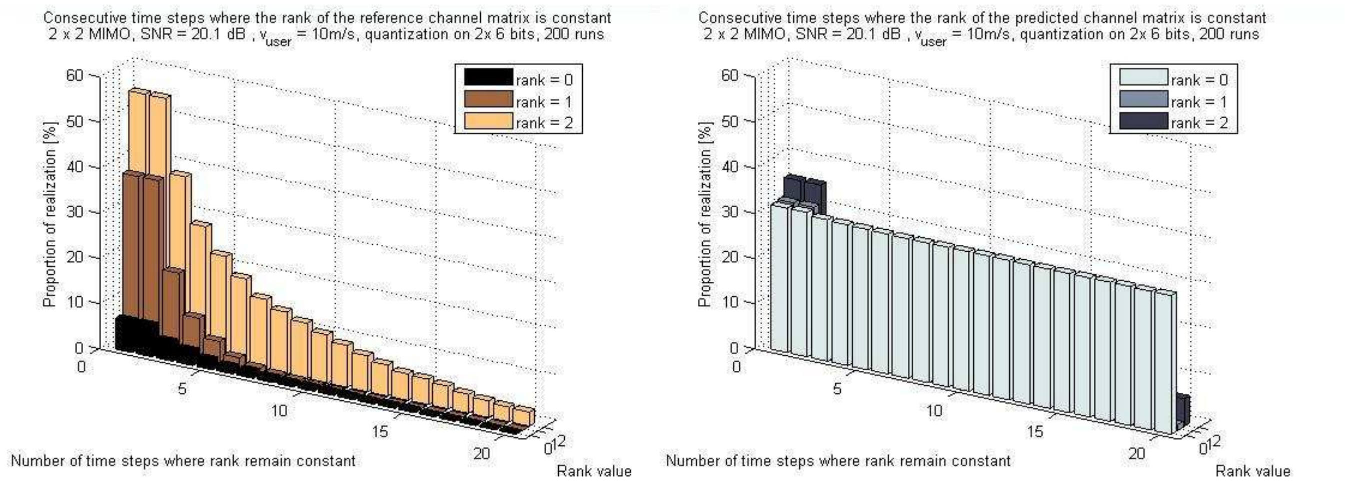


Figure 12-1 : Rank prediction of 2x2 MIMO channel (SNR = 20 dB, quantization on 2x 6 bits)

Then **Figure 12-2** present the same kind of results, but with quantization on 2x 3 bits.

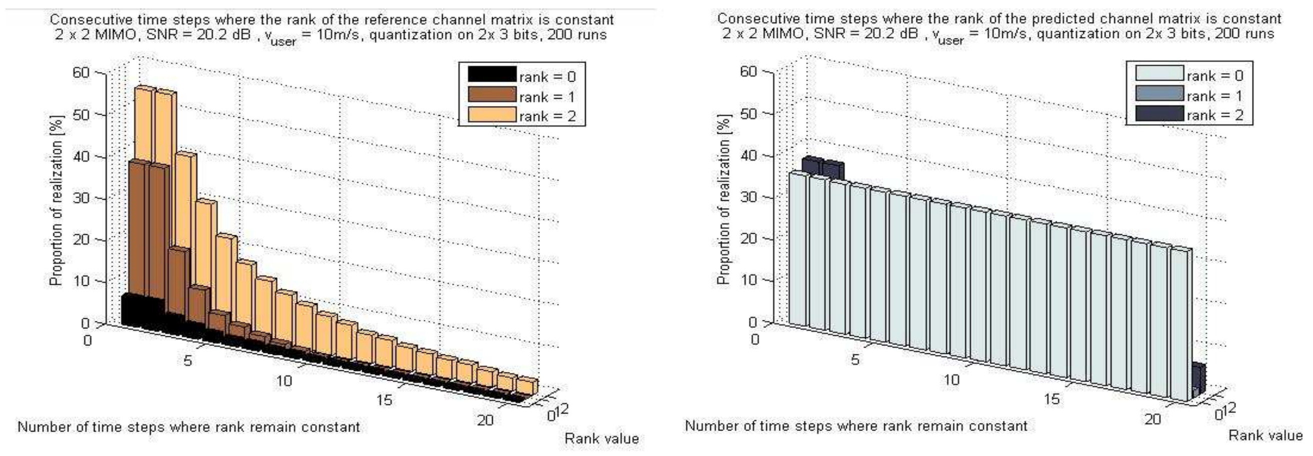


Figure 12-2 : Rank prediction of 2x2 MIMO channel (SNR = 20 dB, quantization on 2x 3 bits)

There is no great difference between 2x 3 bits and 2x 6 bits. But the issue is that predicted matrices often have zero rank. It arises when the rounding operation of the quantization leads to a zero matrix. It's a case of deep fade. The problem is that there are a lot of realizations, about one third, where the predicted channel has a zero rank. With our quantization laws, when using 3 bits, everything in between -0.25 and 0.25 is rounded to zero, and when using 6 bits, it is everything in between -0.03125 and 0.3125 which is rounded to zero. Therefore we tested using 2x 32 bits, to see what happens. This result is shown in **Figure 12-3**.

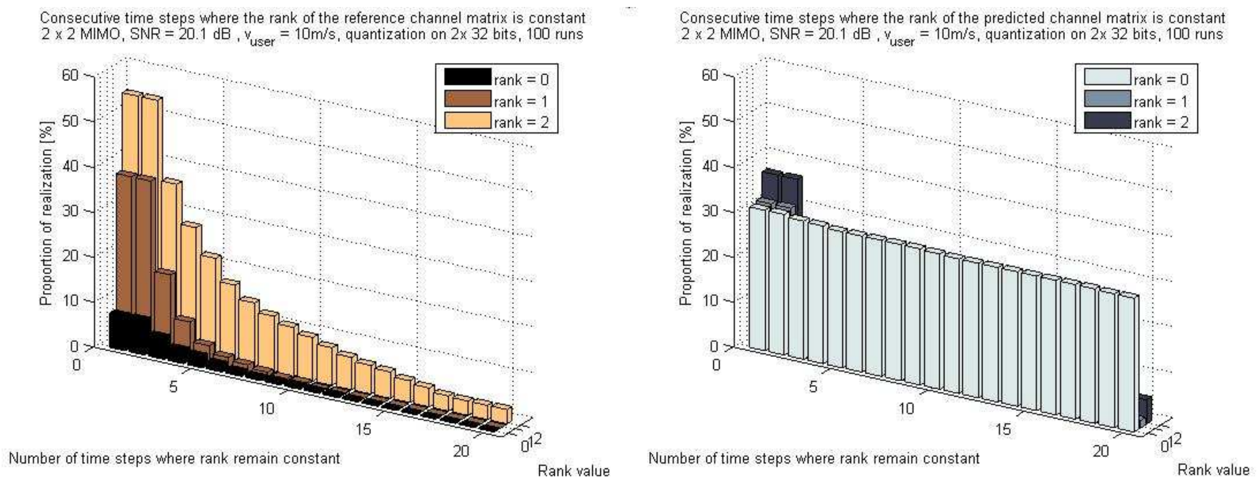


Figure 12-3 : Rank prediction of 2x2 MIMO channel (SNR = 20 dB, quantization on 2x 32 bits)

So increasing the number resolution up to 2x 32 bits does not really increase accuracy of rank prediction. There is still a big difference between the reference channel and the predicted one. The predicted matrices have smaller rank value in average, and have more often zero rank than reference channel matrices.

Let see what happens with no quantization, as it is plotted in **Figure 12-4**.

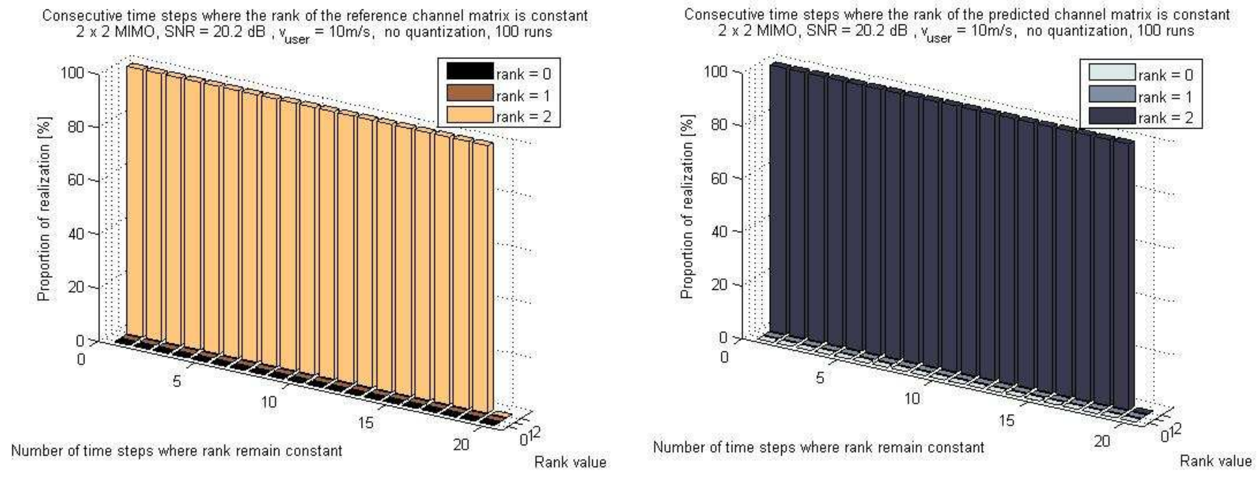


Figure 12-4 : Rank prediction of 2x2 MIMO channel (SNR = 20 dB, no quantization)

So when the channel matrices are not quantized, the rank always remains full. This statement is maybe a consequence of the SCME scenario. But do not forget that we work on computer with finite precision. We used the default Matlab resolution, which is actually 64 bits, so it should bring the same results as the 2x 32 bits. The difference arises because Matlab uses floating point and our quantization laws only consider fixed point values…

The quantization brings in some issues. First, we only consider values in between -1 and 1, for the real and the imaginary parts. But sometimes these values are larger. So if all values are rounded to (1 + i), the rank of the matrix is decreased. Secondly, there is the problem of the zero rank. Actually, quantization on 2x 6 bits means that only 6 bits are used to represent a real value in between -1 and 1. Therefore, every coefficient too close to zero is rounded to zero. Some of our predictions, in case of deep fades, are in the magnitude order of 10^{-2} and lower, and so this may cause a zero rank. The considered tap also has an influence, as it is shown in **Figure 12-5**. If we only consider the first one, which is the most powerful, we can limit the proportion of zero rank.

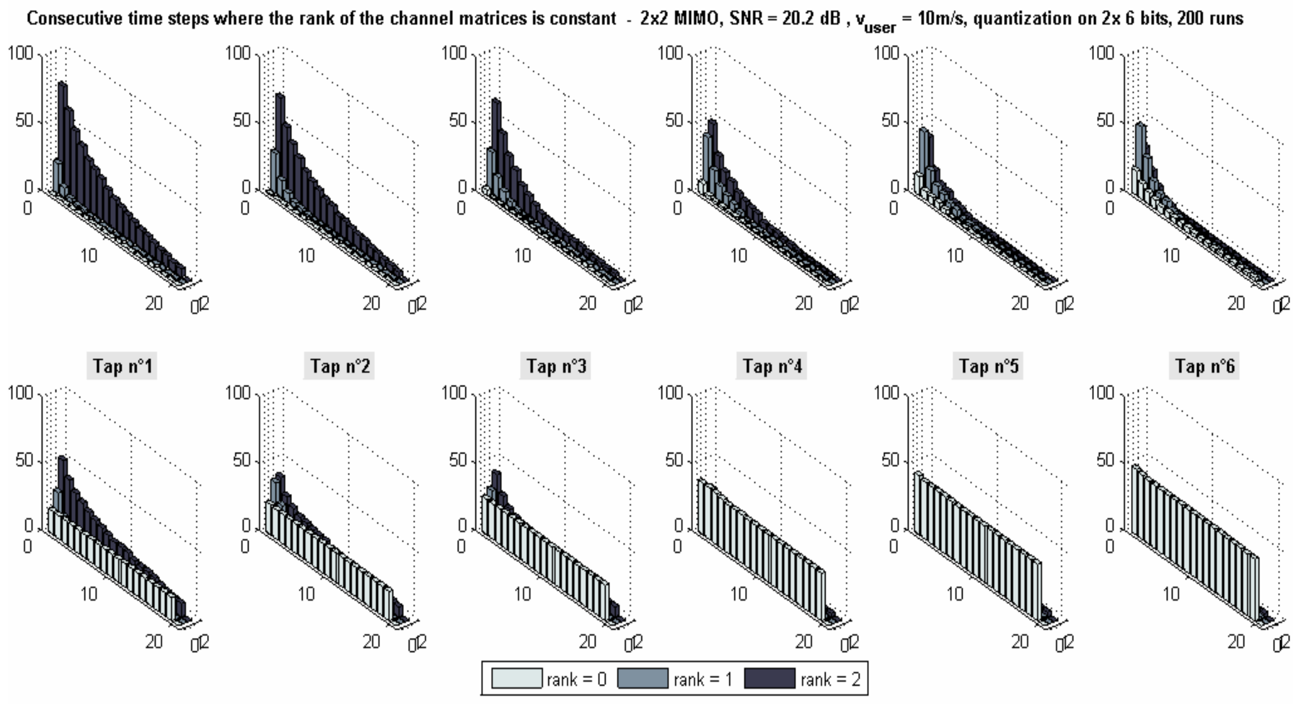


Figure 12-5 : Rank prediction of 2x2 MIMO channel, with each tap considered separately (SNR = 20 dB, quantization on 2x 6 bits)

Therefore it should be better to pick a more suitable quantization law. It would be better to work with a floating point one, and to use a non-uniform representation. On the other hand, detecting cases of deep fade, when the rank is null, could help avoiding the waste of resources.

We will see now how it works when considering 4x4 MIMO channels.

12.2 4x4 MIMO scenarios

Here are now the same kinds of results, but considering 4x4 MIMO channels. **Figure 12-6** present results with quantization on 2x 6 bits of the MIMO channel matrices, whereas results of **Figure 12-7** are obtained with quantization on only 2x 3 bits.

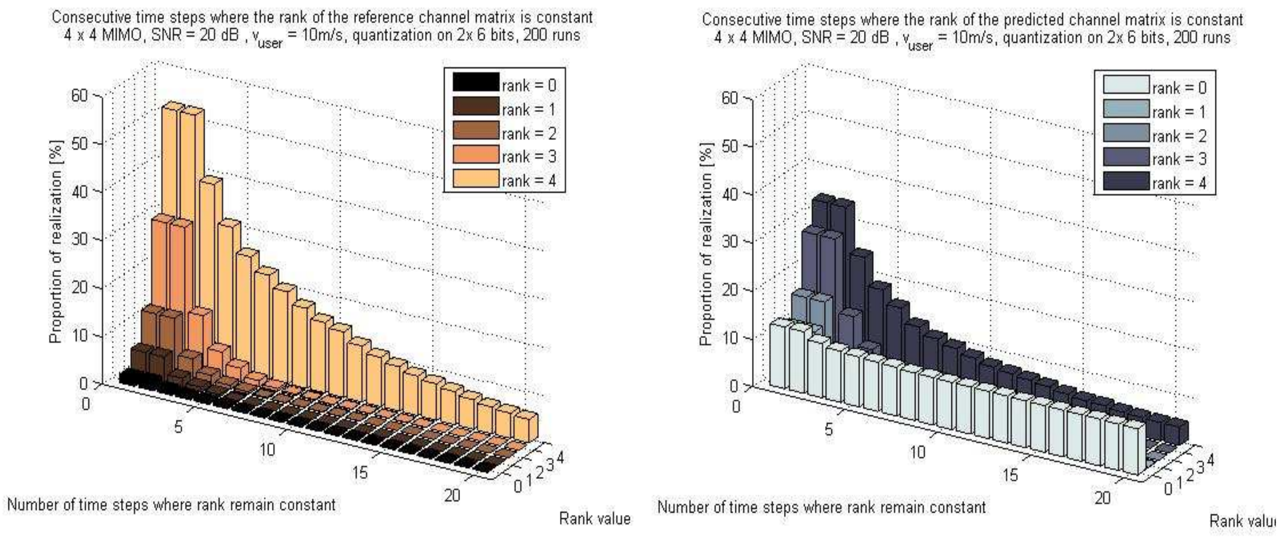


Figure 12-6 : Rank prediction of 4x4 MIMO channel (SNR = 20 dB, quantization on 2x 6 bits)

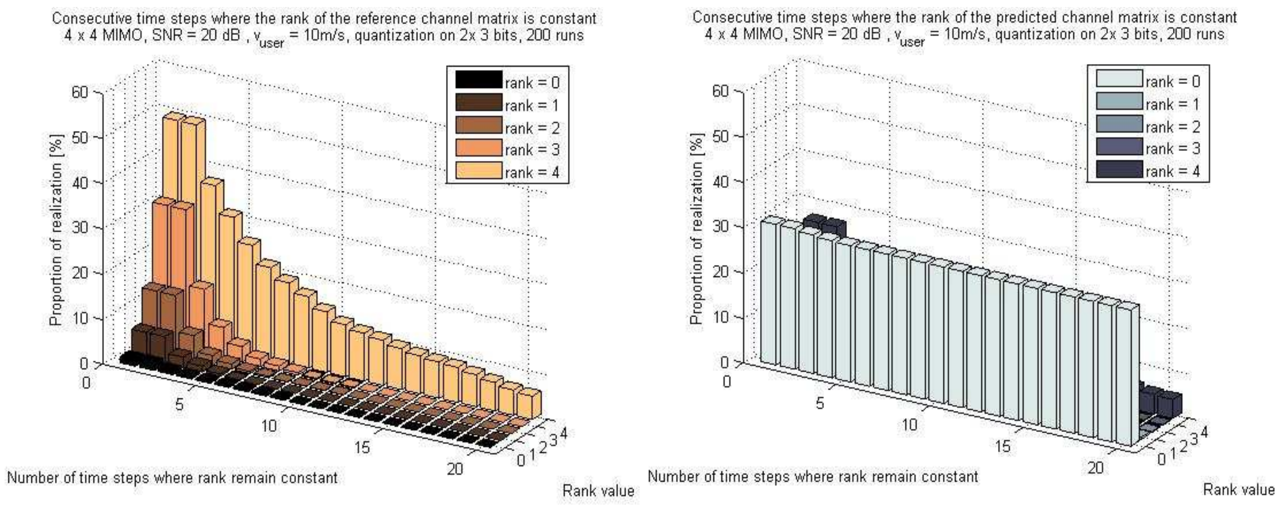


Figure 12-7 : Rank prediction of 4x4 MIMO channel (SNR = 20 dB, quantization on 2x 3 bits)

The results obtained in 4x4 cases are similar to the ones of 2x2 scenarios. The difference is that there are more possible values, and therefore, the values are more distributed. The trouble of zero matrices happens less often. It is because these MIMO channel matrices has size 4x4, it means that 16 coefficients have to be null to get a zero rank, instead of only 4 coefficients in 2x2 case.

Before considering another quantization law, here is in **Figure 12-8**, the time evolution of the ranks for each tap separately. We still have larger ranks on first taps, because there is much power.

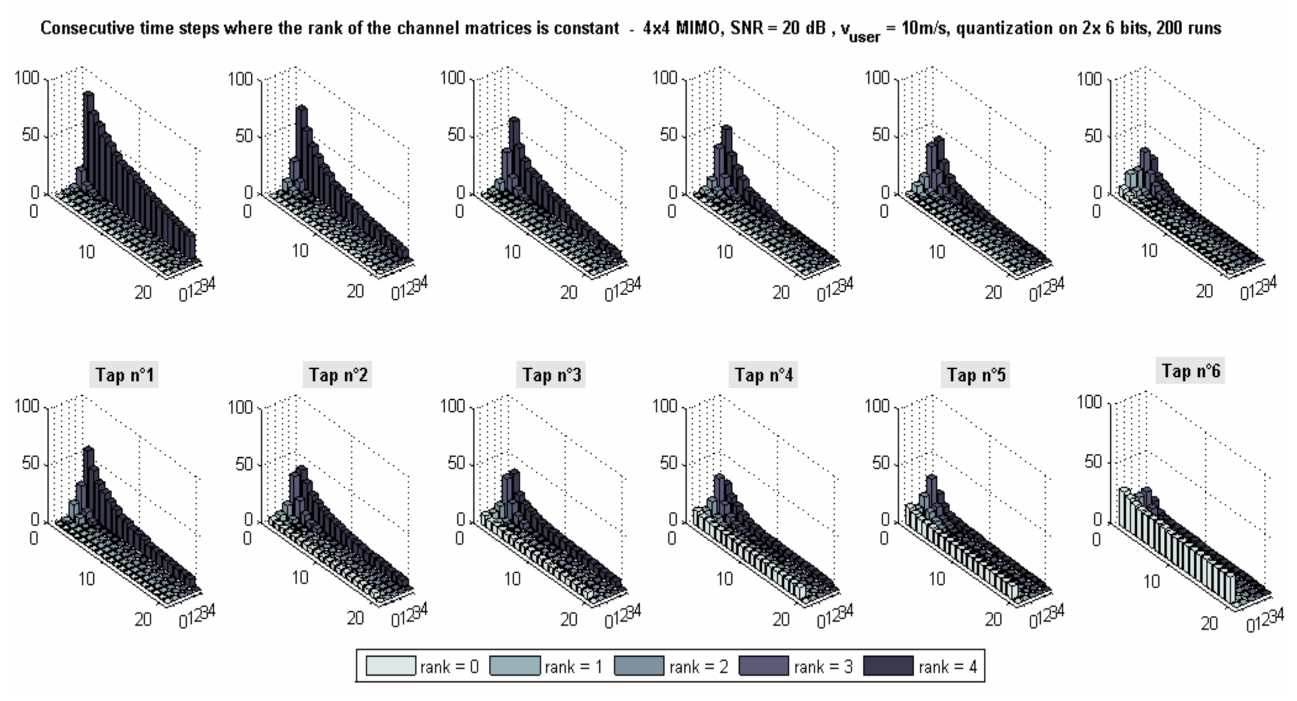


Figure 12-8 : Rank prediction of 4x4 MIMO channel, with each tap considered separately (SNR = 20 dB, quantization on 2x 6 bits)

12.3 Using another quantization law

The trouble of zero rank we experience comes from the quantization step. Small MIMO channel coefficients are rounded to zero. And even when considering quantization on 2x 32 bits, we still have the same problem.

So an idea would be to use floating point to quantize channel matrices before computing their rank. Using floating point would bring extra accuracy near zero. We have already tried floating point quantization when considering channel prediction, and it was useless. Having zero or near zero values leads to the same channel predictions. But when considering rank prediction, having zero or non zero values is different.

We tested different amount of bits for this floating point quantization: from 3 to 6 bits per real value. Actually, when considering 6 bits, 2 are for the exponent, and the last 4 are for the mantissa. And when considering 3 to 5 bits, only one bit is for the exponent, and the others are for the mantissa. To have a larger dynamic with only one-bit exponents, we decided to read this bit either as 10^0 or 10^{-2}. So our quantization law is not uniform.

Figure 12-9 illustrate the quantization law. Because of the special meaning of the single exponent bit, all the curves look the same near zero, in the magnitude order of 10^{-2}. It also explains why there is no difference between 5 and 6 bits when away from zero, because both use 4 bits for their mantissa. Finally, near zero, all laws can achieve values in the magnitude order of 10^{-2}, so there is no visible difference.

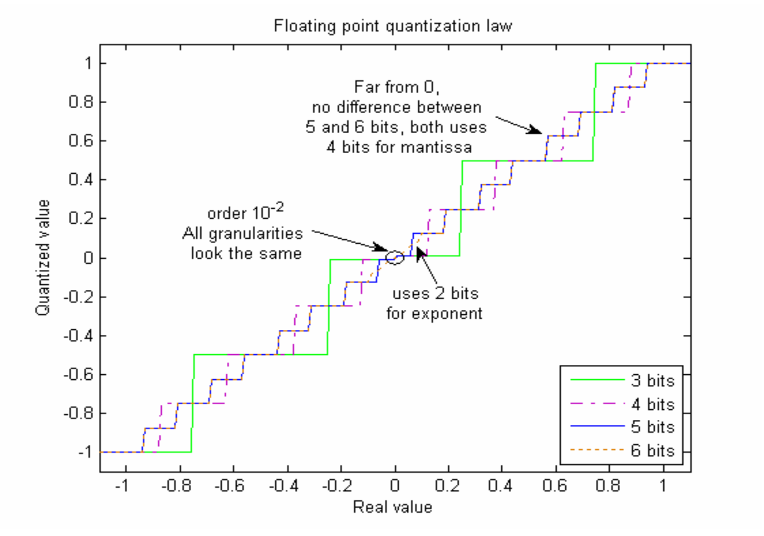


Figure 12-9 : Floating point quantization law, using 3 to 6 bits

Here are now some tests of rank prediction, using these floating point quantization laws. The SNR is 20 dB, and 2x2 MIMO channels are considered. It will underline the influence of the number of allocated bits.

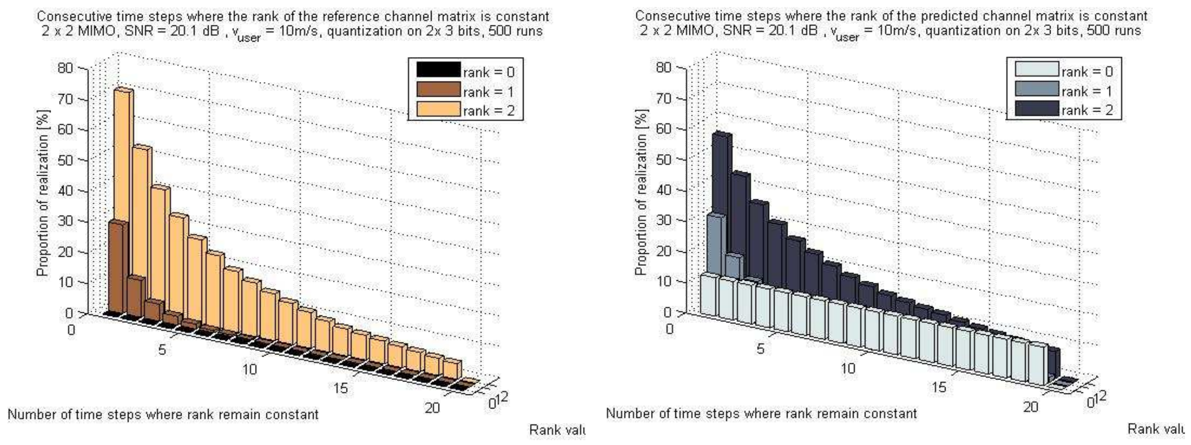


Figure 12-10 : Prediction rank of 2x2 MIMO channel, floating point quantization on 2x 3 bits (SNR = 20 dB)

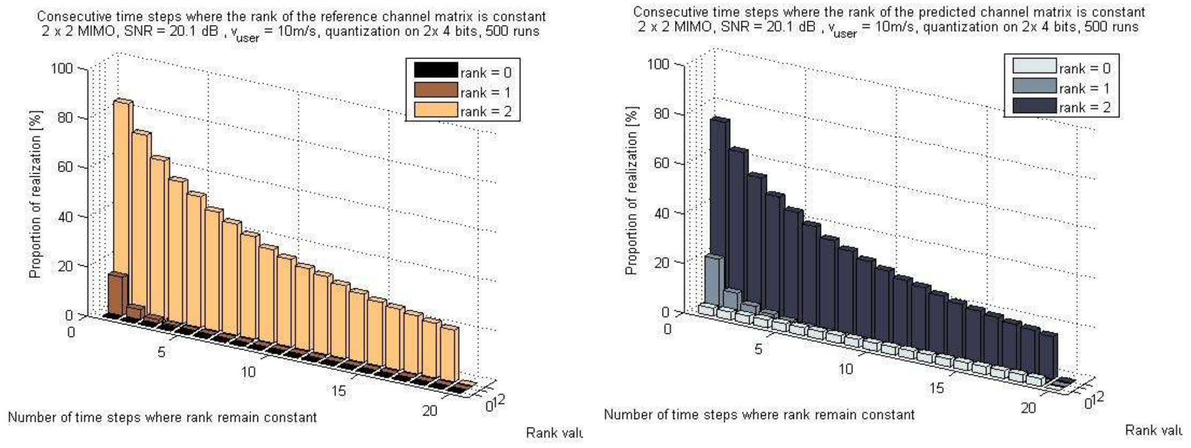
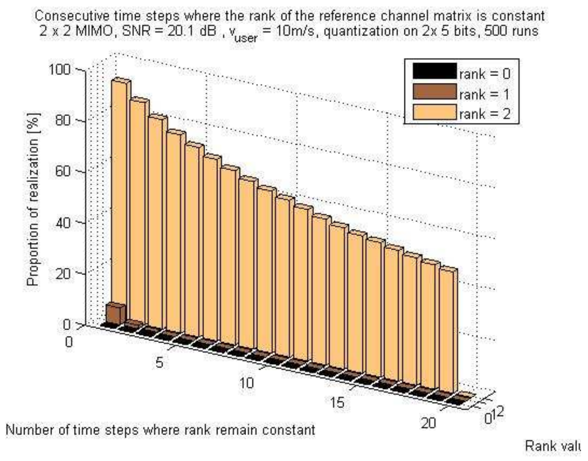


Figure 12-11 : Prediction rank of 2x2 MIMO channel, floating point quantization on 2x 4 bits (SNR = 20 dB)



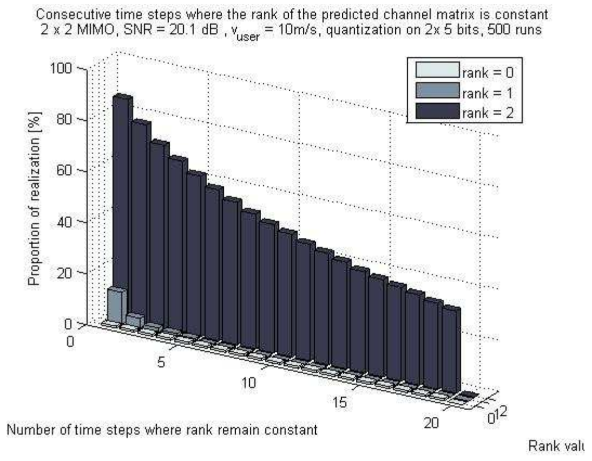
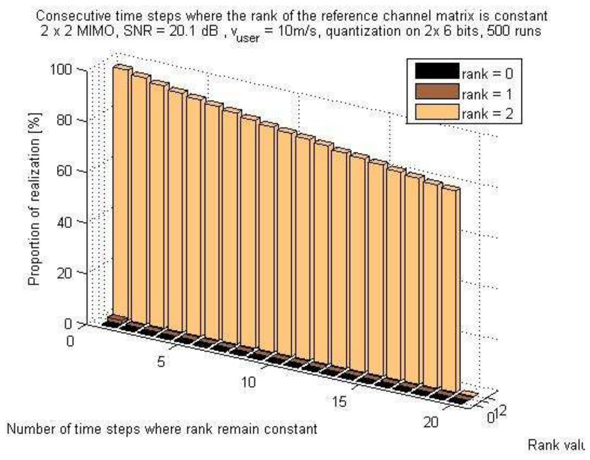


Figure 12-12 : Prediction rank of 2x2 MIMO channel, floating point quantization on 2x 5 bits (SNR = 20 dB)



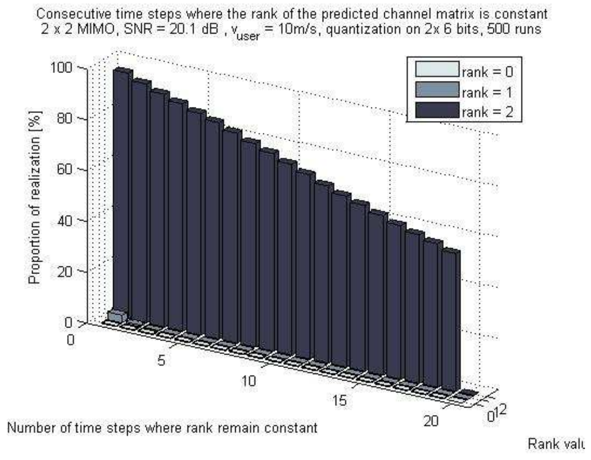


Figure 12-13 : Prediction rank of 2x2 MIMO channel, floating point quantization on 2x 6 bits (SNR = 20 dB)

As one can see, the amount of bits we use influence both reference channel and predicted channel. Using a lot of bits leads to high ranks, whereas using only a few causes low ranks. As mentioned earlier, when there is no quantization, all the ranks are full.

Another thing to point out is that predictions always underestimate the rank, and predicted ranks change faster than reference ranks. But keep in mind that after a while, predictors become outdated, with a too large NMSE.

From **Figure 12-10**, one can draw that using only 2x 3 bits is not sufficient. There is a lot of zero rank predicted, but none in the reference channel. Using 2x 4 bits leads to the best results, and the 2x 5 results are rather good too. When using 2x 6 bits, ranks are full at the beginning, but predicted ranks change faster than reference ranks. Therefore, 2x 4 bits seems to be the most efficient choice, at least for 2x2 MIMO scenarios. We will now move to 4x4 scenarios.

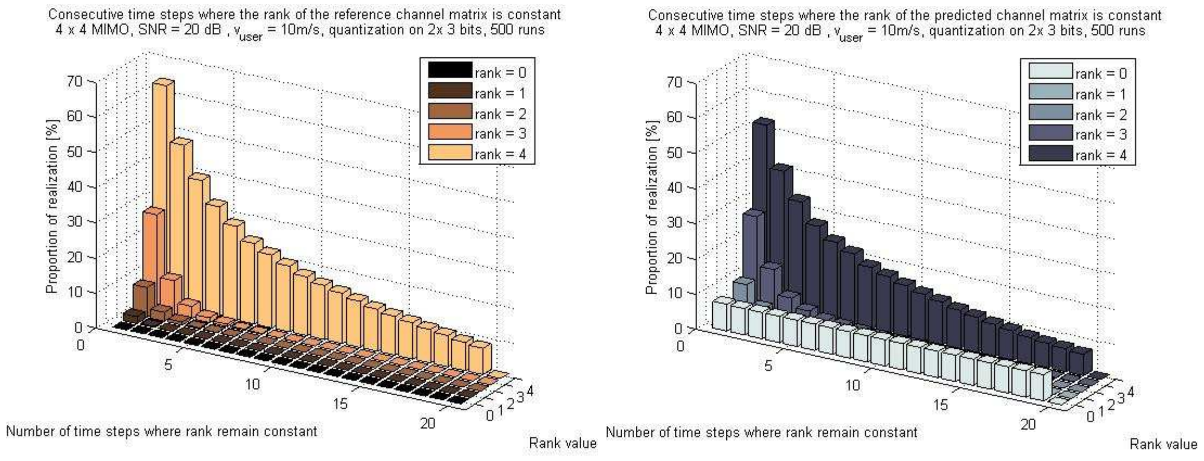


Figure 12-14 : Prediction rank of 4x4 MIMO channel, floating point quantization on 2x 3 bits (SNR = 20 dB)

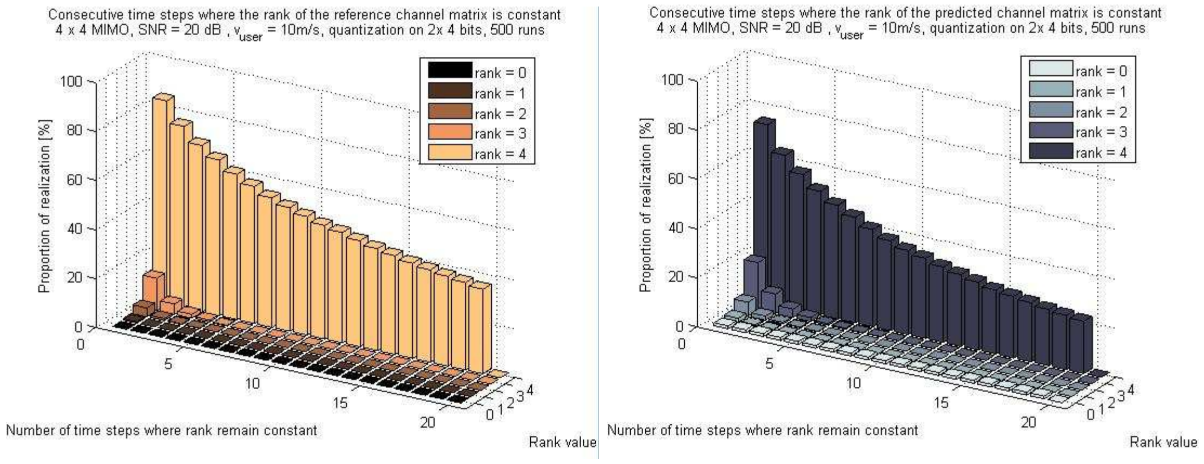


Figure 12-15 : Prediction rank of 4x4 MIMO channel, floating point quantization on 2x 4 bits (SNR = 20 dB)

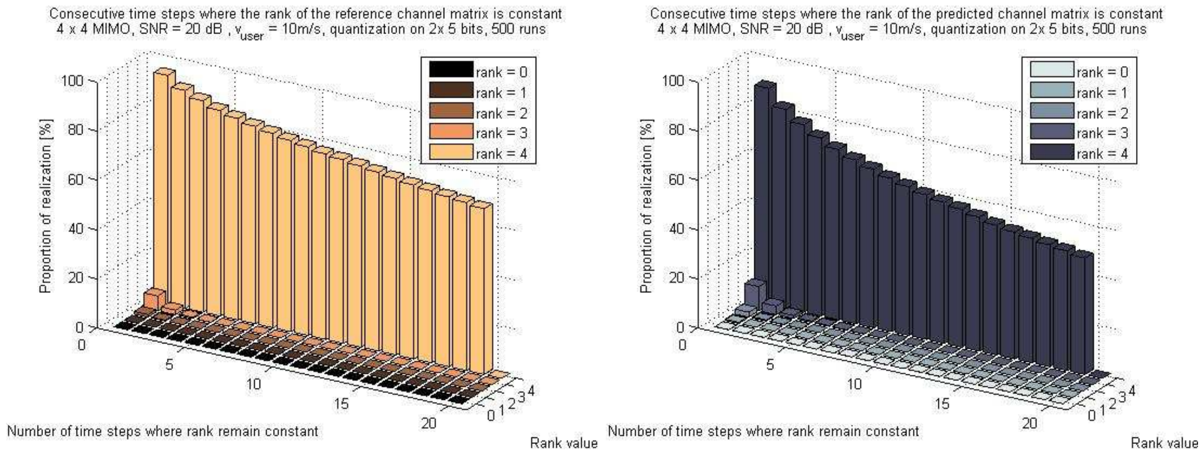


Figure 12-16 : Prediction rank of 4x4 MIMO channel, floating point quantization on 2x 5 bits (SNR = 20 dB)

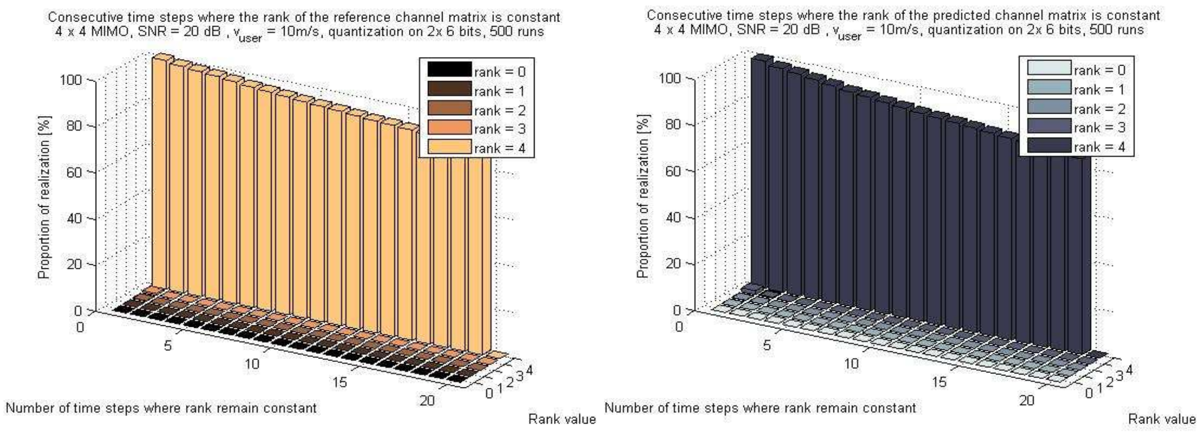


Figure 12-17 : Prediction rank of 4x4 MIMO channel, floating point quantization on 2x 6 bits (SNR = 20 dB)

In 4x4 scenarios, it is still 2x 4 bits which produces the best results, and 2x 5 bits is pretty good too. When considering 2x 6 bits, the rank predictions are really good, but only give full rank. Here are now some simulations considering 2x 4 bits for quantization, but with a SNR of 100 dB.

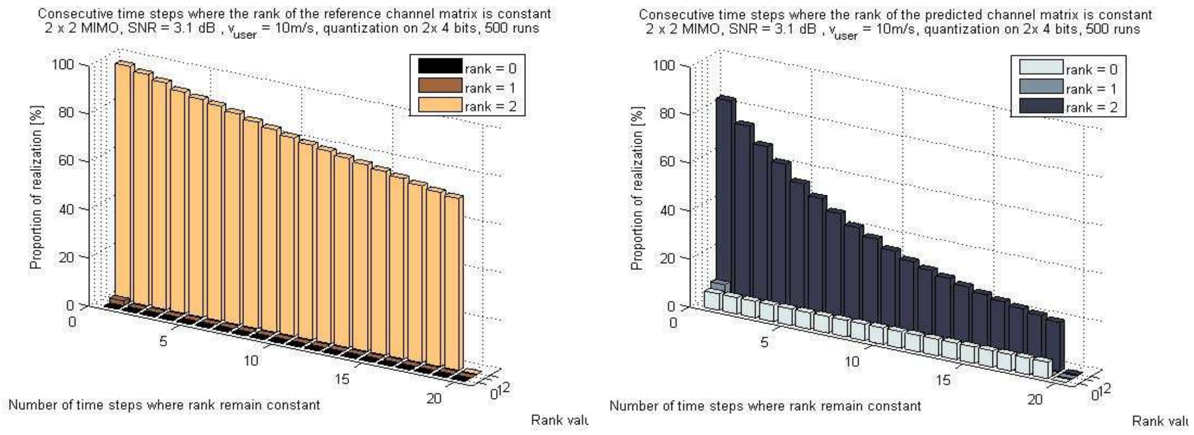


Figure 12-18 : Prediction rank of 2x2 MIMO channel, floating point quantization on 2x 4 bits (SNR = 3 dB)

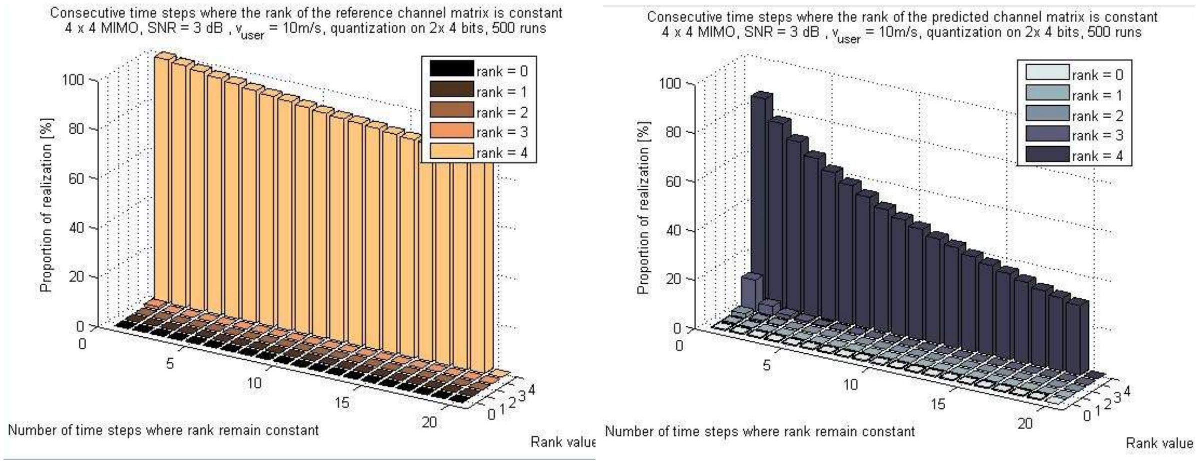


Figure 12-19 : Prediction rank of 4x4 MIMO channel, floating point quantization on 2x 4 bits (SNR = 3dB)

At the low SNR of 3 dB, predictions are not as good as at SNR of 20 dB. Predicted ranks change faster than reference ranks. Though there is no joined figure explaining it, using more or less bits than 2x 4 for quantization does not improve rank prediction.

Then let us have a look at results when SNR = 100 dB.

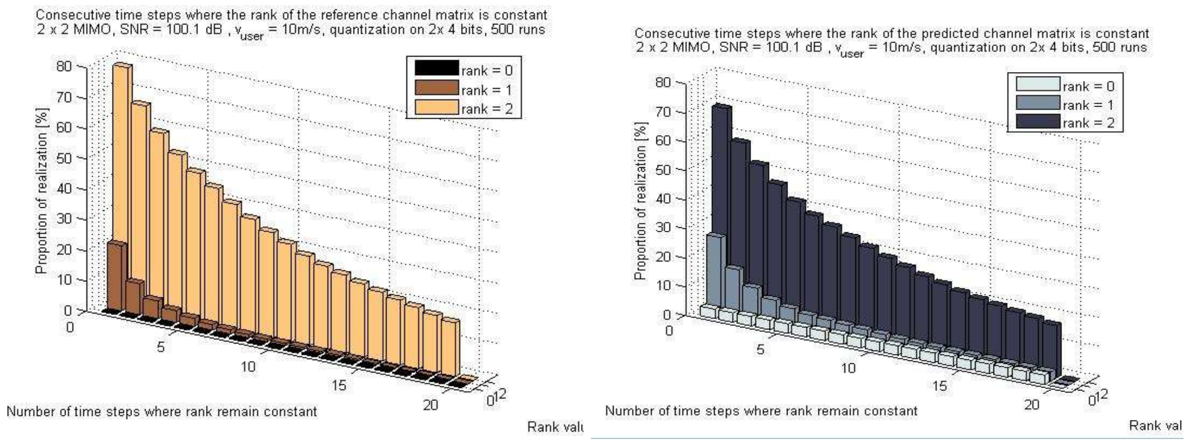


Figure 12-20 : Prediction rank of 2x2 MIMO channel, floating point quantization on 2x 4 bits (SNR = 100 dB)

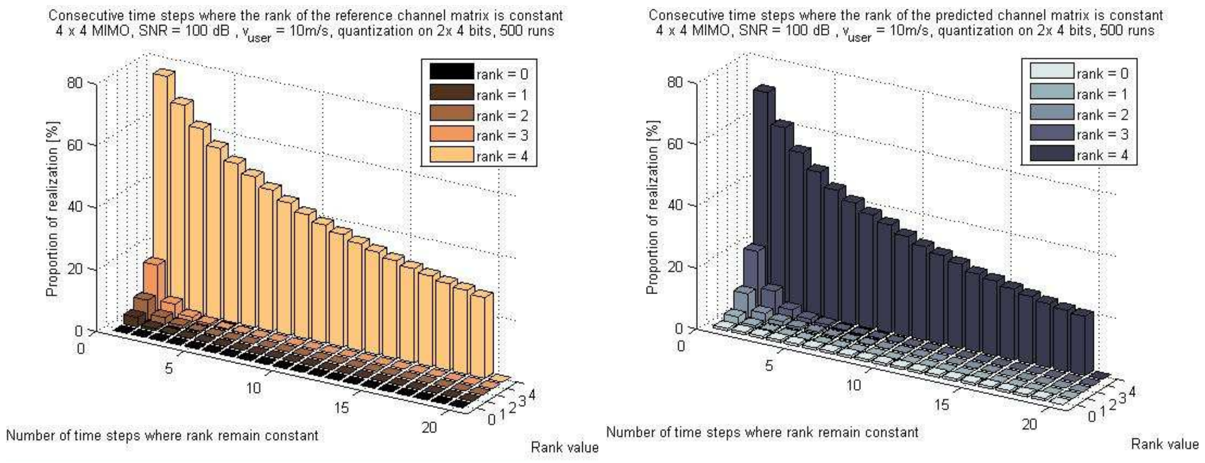


Figure 12-21 : Prediction rank of 4x4 MIMO channel, floating point quantization on 2x 4 bits (SNR = 100 dB)

Results are good at this high SNR, but predictions always underestimate slightly the reference rank. Still now, choosing 2x 4 bits for quantization seems to be the optimal trade off.

12.4 Conclusions on rank prediction

When considering rank prediction, the quantization is very important. If there is no quantization of the MIMO channel matrices, the rank always remain full. When quantized, these matrices often have zero rank troubles, especially the predicted ones. It comes from the rounding operation of the quantization: all small values are rounded to zero, which may cause a zero matrix.

This is the reason why we used a floating point quantization law. Because of the exponent, we can have enhanced precision near zero, with only a low amount of bits. We have shown that using 2x 6 bits per MIMO channel coefficient leads to nearly constant full rank, as the unquantized scenarios. Then we found that using only 2x 4 bits with our non uniform quantization law seems to be the optimal trade-off to get the best rank prediction.

13 REFERENCES

[3GPP 07] 3GPP TS 36.211 V8.0.0 (2007-09).; 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Physical Channels and Modulation (Release 8).

[And 99] J.B. Andersen , J. Jensen, S.H. Jensen and F. Frederiksen, “ Prediction of Future Fading Based on Past Measurements”, Vehicular Technology Conference (IEEE VTC Spring), 1999.

[Arr 02] A. Arredondo, K. Dandekar and G. Xu, “Vector Channel Modeling and Prediction for the Improvement of Downlink Received Power”, IEEE Transactions on Communication, Vol. 50, No. 7, July 2002.

[Bar 03a] S. Barbarossa, G. Scutari, S. Vergari, G. Paccapeli, “Robust Estimation and prediction methods for Time-Variant SISO channels”, IST ROMANTIK D421, June 2003

[Bar 03b] S. Barbarossa, G. Paccapeli, G. Pecchini, G. Scutari, “Robust Estimation and prediction methods for Time-Variant MIMO Systems”, IST ROMANTIK D422, July 2003

[Bau 05] D.S. Baum, J. Hansen and J. Salo, “An Interim Channel Model for Beyond-3G Systems”, Vehicular Technology Conference (IEEE VTC Spring), 2005.

[Che 03] Z. Chen, J. Andrews and B. Evans, “Short Range Wireless Channel Prediction Using Local Information”, 37th Asilomar Conference on Signals Systems Computers, Nov 2003.

[SCM 05] J. Salo, G. Del Galdo, J. Salmi, P. Kyösti, M. Milojevic, D. Laselva and C. Schneider. (2005, Jan) MATLAB Implementation of the 3GPP Spatial Channel Model (3GPP TR 25.996) [Online] Available: http://www.tkk.fi/Units/Radio/scm/

[SD 7.3] SURFACE Deliverable D7.3, *First results from link and system level simulations.*

[SURF] 6th Framework Programme, Information Society Technologies, Self Configurable Air Interface (SURFACE), IST-4-027187-STP, [online] Available: http://www.ist-surface.org/

[Sto 05] P. Stoica and R. Moses, “Spectral Analysis of Signals”, Pearson Prentice Hall, 2005.

[WIN] 6th Framework Programme, Information Society Technologies, Wireless World Initiative New Radio (WINNER), IST-2003-507591, [online] Available: https://www.ist-winner.org/

[WIN 6.13.7] IST-2003-507581 WINNER II, ”D6.13.7 - WINNER II Test Scenarios and Calibration Cases Issue 2“, Deliverable v1.0.0, December 2006

14 CONCLUSIONS

Our objectives in this deliverable were on the one hand to present a wireless MIMO channel model, and on the other hand to provide a scheme able to perform predictions of the channel state. The channel model we worked on is a wrapper to the WINNER's SCME channel model. Its parameters have been tuned to meet SURFACE requirements discussed in deliverable D7.3 [SD 7.3].

About channel prediction, after having investigated several channel models, mostly SISO models, we found two predictors suited for our working environment (energy and CPU-limited devices), except that they are SISO's.

We implemented in Matlab these two SISO predictors and tested them with the WINNER's SCME channel simulator. We analyzed them, the choice of some of their parameters, and also their robustness against noise.

From these computations, it seemed that the two predictors produce the same kind of results, even if ESPRIT is a bit better for the first predicted values. We have also shown that in noisy conditions, using only a limited amount of channel observations produces better results; even if in noise free conditions it was more efficient to use a maximum of channel observations.

We then extended these two SISO predictors into MIMO channel predictors, which is actually the scope of this document. The two new MIMO predictors are named MECoM, which is the extension of the original ESPRIT technique, and MEHaM, based on the Andersen et al. techniques. Both predictors are not really dependent on the number channel observation used to initialize them, so we found that only 5 MIMO channel observations are sufficient to train a 2x2 MIMO predictor, and 17 are required to train a 4x4 one.

With some improvement, MEHaM was found to be the best predictor. Moreover, it is also the simpler, because it does not require noise estimation when computing the signal poles.

Afterwards, we investigated the robustness of the MIMO predictors to limited feedback load, by working with quantized values. We have shown that going from the full number resolution of MATLAB to 2x 6 bits quantized values has barely no influence on the predictors' performance.

From all these simulations, we have merged the two MIMO predictors into a single on, taking best of both. Actually, the new predictor is mainly based on MEHaM, but without discarding the largest poles. We have shown that we can limit the feedback load by quantization on 2x 4 bits per complex coefficient without decreasing prediction performances. We have also tested the difference between floating point and fixed point quantization laws, but it seems floating point has no interest in this case.

Finally, we have investigated the topic of rank prediction. The proposed scheme is to use the new MIMO predictor to predict future MIMO channel matrices, to quantize them and then to compute their rank. Without the quantization step, all ranks were full, which is not an interesting case. Our tests revealed that fixed point quantization produces poor results. The predicted rank was often zero, which means the predicted MIMO channel matrix, after quantization, was actually a matrix full of zeros. That problem actually occurs at the rounding operation of the quantization step. Near zero values are rounded to zero, and so the rank decreases.

This is the reason why we then have tried a non uniform floating point quantization technique, to get extra precision near zero. We have shown than using only 2x 4 bits per complex MIMO coefficient, with our non uniform quantization law, produces rather good results, even if they slightly underestimates reference MIMO channel rank. Rank predictions are only a bit poor when SNR is too low, at 3 dB in our simulations.

Hence, for future work, following topics should be considered:

- Working on refreshment scheme of the MIMO channel predictor. The performances of the predictor are known. Then from the requirements of the air interface, a threshold should be set-up to decide whether a prediction is reliable or not. Investigations on the capacity to guess the reliability horizon of a given predictor should be done, in the aim to automatically trigger refreshments of the predictor when it is outdated. It means as late as we can, to save computational power of simple user equipments like cell phones, but not too late to avoid using an outdated predictor.

- Improving the predictors, trying to optimize the algorithm, and to reduce the computational load, in the aim of practical implementation.

- Working on more efficient techniques of rank prediction. The rank predictor should be simplified. If we are only interested by the rank of a matrix, it is not optimal to spend time and power to compute all its coefficients.

15 DOCUMENT REVISION HISTORY

Date	Revision	Author	Comments
November 28, 2006	0.1	Joël Vanderpypen	Creation ; description of SISO Andersen et al. Predictor
December 8, 2006	0.2	Joël Vanderpypen	Addition of SISO ESPRIT predictor
December 15, 2006	0.3	Joël Vanderpypen	Addition of state-of-the-art review of Section 3
December 20, 2006	0.4	Joël Vanderpypen	Editing following comments from Laurent Schumacher; addition of SISO results
December 22, 2006	0.5	Joël Vanderpypen	Editing following comments from Laurent Schumacher; addition of MIMO results
January 12, 2007	0.51	Joël Vanderpypen	Further editing
January 15, 2007	0.52	Laurent Schumacher	Further editing
January 17, 2007	0.53	Joël Vanderpypen	Further editing ; addition of the pole plot
January 18, 2007	0.54	Laurent Schumacher	Further editing
January 19, 2007	0.55	Joël Vanderpypen	Further editing
January 22, 2007	1.0	Joël Vanderpypen	First release
August, 10, 2007	1.1	Joël Vanderpypen	Editing Executive summary, introduction, sections 6, 8 and conclusion; Addition of sections 9, 10 and 11.
November, 15, 2007	1.2	Joël Vanderpypen	Further editing, addition of the SURFACE channel model section
November, 30, 2007	1.3	Joël Vanderpypen	Further editing
January, 18, 2007	2.0	Joël Vanderpypen	Final release