[✓ **Public**]

Confidential

Project:	**CHAMELEON RF**
Project Number:	FP6/2004/IST/4-027378
Work Package:	WP3
Task:	T3.1
Deliverable:	D3.1a

Title:	Report on Parametric Model Order Reduction Techniques
Author(s):	L.M. Silveira, J. Fernández, P. Flores, G. Ciuprina D. Ioan, D. Niculae, A. Stefanescu, R. Janssen, W. Schilders, N.v.d. Meijs, K-J.v.d. Kolk, E. Seebacher, W. Pflanzl
Affiliation(s):	INESC-ID, PUB, NXP, TUD, AMS
Date:	01-May-2007

Table of Contents

1 Abstract

This document presents an extensive study of Parametric Model Order Reduction techniques and summarizes the activities carried out in Task 3.1 of the CHAMELEON RF project. These activities include a study of the variability and process uncertainty that leads to parameter dependent models, the research towards an efficient and accurate representation of such models, the validation of this chosen representation, and the study and report on the techniques for reducing those models without loss of parametric dependence. A framework for carrying out the tests required was specified and developed, including standards and functions for the representation, processing, reduction and comparison of the various procedures. All these data structures, procedures and functions were included in the Parametric ROM Workbench framework, a comprehensive extension of the previously available ROM Workbench. This parametric framework was also further extended in order to cover the reduction of interconnected models, using techniques consistent with the ones presented in D3.2.

2 Introduction

In today's fabrication processes, pushing into the nanometric regimes, certain variability effects and the resulting degrees of uncertainty, previously assumed small are now gaining a protagonist role. Process-induced variations are becoming more relevant as feature sizes decrease, pushing the limits of lithographic and deposition technologies. Such variations, which are hard to predict and control, may have a first-order effect on the functionality of designs or the accuracy of the resulting description models. Proper accounting for such variations is therefore necessary for reliable analysis, and needs to be taken into account in the modelling phase. The parameter variability can no longer be disregarded during modelling, simulation and verification, and their effects must be taken into account in early design stages so that side-effects, including potential delays in the design cycle, can be minimized. Within WP3, one of the major goals is to take into account these effects and to provide techniques and tools for the reduction of the large passive system descriptions generated in the extraction steps. These procedures must be able to handle these variation effects.

According to the Description of Work (DOW), the main objectives of Work Package 3 are:

1 To develop robust and efficient methods for order reduction of parameterised system descriptions generated in WP2, which can take into account variability induced by lithography or process variations as well as changing operating conditions.

2 To investigate existing order reduction techniques and to develop new algorithms which are applicable to massively coupled systems. Such methods must enforce required physical properties and guarantee the generation of realizable simulation models.

The second item has already been addressed and the results were previously presented in deliverable D3.2 [29]. The first item's objectives are the main motivation for task 3.1 and this deliverable. In this task we study the potential compression of the parameterised models generated in WP2. Parameterised model order reduction (pMOR) is an area in its infancy and many of the techniques available so far are trivial extensions of the standard ROM techniques which have limited applicability, or generalized approaches which are computationally very demanding. In this task, emphasis will be given to the reduction with respect to the parameters. With these techniques it is expected that (reduced) parameterized models can be created to be used in later simulations, so that designers can be made aware of the effects that these small fluctuations will have on the behavior of their RF ICs during the manufacturing process. These tools will also enable significant simulation speed up when sweeps or small-scale modifications over certain design parameters are applied, providing the simulators with faster and more accurate capabilities.

We will investigate whether advantage can be taken of the knowledge about the specific parameters in the reduction process, both in terms of compactness as well as efficiency. Enforcing the relevant physical properties on the reduced models is critical for the robustness of subsequent analysis. For standard interconnect structures there are well known techniques that can guarantee passivity of the model. However, when parametric descriptions are considered it represents an open field.

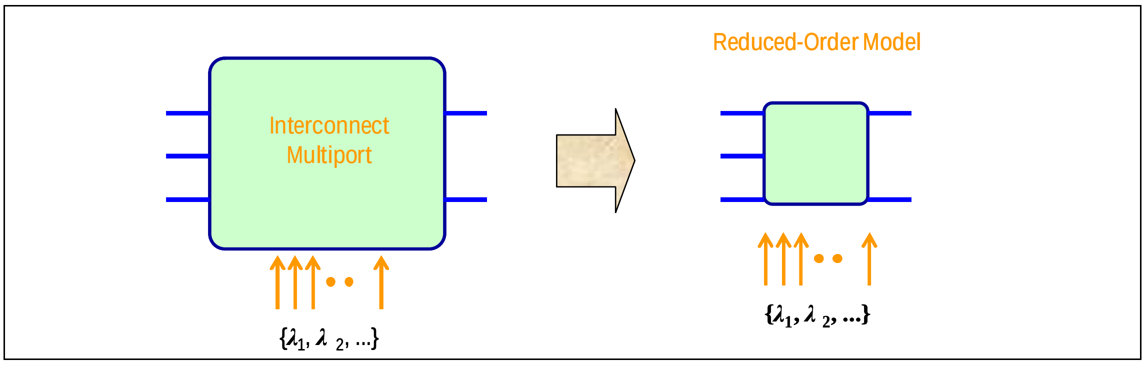

Interconnect Multiport
$\{\lambda_1, \lambda_2, \ldots\}$
Reduced-Order Model
$\{\lambda_1, \lambda_2, \ldots\}$

Figure 1. Parametric Model Order Reduction

One should be aware that the effects of the parametric uncertainty may not only affect one single system, but also the neighbouring devices. Therefore, to treat these devices in isolation could lead to gross inaccuracies and would represent a lack of consistency with previous work. Therefore, in this deliverable, the techniques reported will be combined with the ones previously presented in D3.2, and a deeper study of the complete picture, involving interconnection of systems, will be carried out. Such models, which may represent the parasitic interactions or EM effects, are generated in WP2 and consist of specific instances of the generic environment models envisioned in WP1 for the passives and the parasitic couplings. The descriptions may account for all possible interactions between the various design elements, leading to circuit-interconnect structures with large number of couplings. The descriptions are also parameterised in order to account for possible variations induced by lithographic or process variability. The large number of couplings considered together with the addition of parameters as model inputs is expected to lead to very large models, whose complexity must be reduced in order to enable efficient simulation.

3 Variability

The first step in the study of parametric Model Order Reduction is, of course, to understand the causes and the effects of the variability itself, in order to be able to cope with them in a more efficient way. For this reason the first stages of the research were dedicated to a review of existing literature related to the topic [22]-[25].

The main causes of the variability are related to the uncertainties in the fabrication process, whose effects are more relevant in the physical and electrical characteristics of the IC layout. Although extremely controlled, the fabrication process is always prone to small fluctuations of the environment, leading to small physical and electrical differences between the separate ICs and the original design. These small errors in production are generically a consequence of one of two different scenarios: a methodological deviation, that may be compensated and minimized, and a purely random effect, impossible of predict, but that may be modelled following some statistical studies.

The literature reviewed showed a glaring lack of real data and only pointed to certain general parameters, some of them related to either the physical part of the circuit (width, thickness, etc…) or either electrical characteristics (dielectric constant, conductivity, etc…). This is understood given in some cases the extreme sensitivity of the information. However, nowadays integrated circuits are composed of up to 9 layers, including different materials, doping profiles, etc… This yields a considerable amount of possible variability sources.

Information existing or generated within the project by the partners was used as starting point for the validation of the models to be used inside CHAMELEON RF. Tests were performed taking into account the nature of the parameter and its variation range. Summarized results are presented in Section 4.3, and more detailed experiments are described in Section 9. These results corroborate the type of formulation proposed in Section 4 for representing parameterized systems.

4 Requirements for Reduction of Parametric Compact Models

In order to be able to account for the effects of variability at the electrical level, during simulation, it is necessary that variability-aware models be available for such analysis. Such models must be produced earlier, possibly at lower levels of abstraction. Electromagnetic simulation environments, for instance, should account for the uncertainty caused by the process variation when generating models of structure. Also, the systems obtained after an extraction step must also be variability-aware presenting a dependence of some form with respect to the varying parameters. However, it is not clear at all what is the best option when representing such parametric system. Furthermore, there is no standard and no measure rank of the benefits and drawbacks of different representations. On the other hand, to set a base representation is the first and one of the most important steps when a common framework is meant to be built, and the results of the rest of the procedures depend to a great extent on this base representation.

From the perspective of WP3, which is mostly concerned with the reduction of the system representation size, the chosen representation should be compatible with most of the existing techniques for (parametric) Model Order Reduction. However, at the same time it must also be easily generated with the technologies and tools available from WP1-2 work (the pursued software must be efficient along all the stages, from extraction to reduction). An important characteristic of the chosen representation should be the explicit representation of the parameters, i.e. the possibility of accessing the parameter values and modifying them without doing additional costly extractions. Another crucial requirement is that the reduction techniques must be able to maintain the chosen representation after the reduction, i.e. the algorithms must receive and return systems represented in the same manner, where the parametric dependence is given in an explicit way. This ensures that the same model can be used for different values of the same parameter without the need of repeating the reduction, guarantees the possibility of multiple reduction steps, enables the combination of several algorithms and, to summarize, allows us to treat the results as potential inputs to the software to be developed. Due to this, the possibility of using and maintaining the chosen representation will be taken as a measure of the quality of the algorithms.

Another important issue is that these parameterized models will represent passive devices to be included in RF blocks, and as was already said, the coupling effects and the relation between them can no longer be ignored. This was the main reason for the efforts undertaken in Task 3.2 (for further details see deliverable D3.2b). Therefore, the techniques developed for Interconnected Systems must be extended for handling these parametric systems, such that a complete technique will be obtained that is usable within the objectives of CHAMELEON RF WP3. This means that the representation must be general enough for being efficiently included as a part of an interconnected set of compact models.

The first stages of the research were devoted to defining a clear framework in cooperation with the consortium members. A lot of literature on the topic was surveyed, and most of the potentially useful algorithms were studied in order to find some common representation.

4.1 Parametric Systems

In order to consider the effects of process variability the systems of relevance within the context of WP3 are parametric, and the response of such systems depends on the values of the parameter set. To include the parametric systems inside an efficient simulation flow, the parametric dependence must be explicit, i.e. it must be possible to access the parameter values and modify them inside the same representation, while avoiding, if possible, re-computing the parametric systems, i.e. to perform another extraction.

The process uncertainty usually directly affects the geometrical or electrical properties of the layout, and therefore, most of these variations can be represented as modifications of the values of the system matrices inside a state space descriptor (1). Therefore, the access ports of the inputs and outputs are not usually affected by these variations (this of course depends on how the system is built), and in the case when they are in fact affected, these variation can be shifted to the inner states.

In the general case, the system can be represented with the parametric formulation

$$\begin{aligned} C(l)\cdot\dot{x}(t,l) + G(l)\cdot x(t,l) &= B\cdot u(t) \\ y(t,l) &= L\cdot x(t,l) + D\cdot u(t) \end{aligned} \qquad \textbf{(1)}$$

where $C(l)\in\Re^{nxn}$, $G(l)\in\Re^{nxn}$, $B\in\Re^{nxm}$, $L\in\Re^{pxn}$, $D\in\Re^{pxm}$ are the state space matrices, $u(t)\in\Re^{m}(t)$ is the vector of inputs, $y(t)\in\Re^{p}(t)$ is the vector of outputs and $x(t,l)\in\Re^{n}(t)$ is the vector of inner states. The elements of the matrices C and G, as well as the states of the system $x(t,l)$, depend on a set of P parameters $l=[l_1, l_2..., l_P]$ which include the above mentioned uncertainty.

Similarly, this yields a parameterized frequency based representation of the transfer function of the type

$$H(s,l) = L\cdot\left(s\cdot C(l) + G(l)\right)^{-1}\cdot B \qquad \textbf{(2)}$$

However, the parameter dependence of the elements (entries) of these matrices may be very complicated and not obvious (this parametric dependence can follow some extremely complicated functions that vary from element to element).

Most of the currently used MOR techniques rely upon the projection of the state space matrices onto a suitable subspace in order to obtain a set of reduced matrices that capture the most relevant behavior of the system [4]-[9]. The drawback of this type of reduction is that the inner structure of the system is lost (this phenomena has a close relation with the methods and motivations in D3.2, but at a different level), which means that if the entries of the matrices have a complicated implicit parameter dependence, after reduction these dependences are completely mixed and lost, which makes usage of the model difficult in the manner previously envisioned. This situation is not desirable for later usage inside a simulation environment as the parameters may not be directly accessible and the representation may only be valid for certain parameter values. At this point it is clear that an explicit dependence of the system with respect to the parameters is highlydesirable in order to simplify usage in simulation flows.

4.2 Representation of Parametric Systems

Most of the existing parametric frameworks are based on the study of perturbation effects, so affine models are built in most of the cases. In fact, the surveyed literature pointed towards a Taylor Series expansion of the matrices C and G with respect to the parameters (more precisely a multivariate polynomial expansion) as a common means to capture such parametric behaviour. Since, as previously mentioned, most of the algorithms are based on projection techniques (these will be presented in the next sections) this approach seems to lead to a suitable representation in terms of the goals of the current Task. The projection based methodology performs a congruence transformation on the system matrices that has been proved to be very advantageous in certain settings, and, in addition, in this case it also preserves the Taylor Series representation.

Other important advantage of this representation are the flexibility it offers for building the system and simplicity of storage as well as the possibility of increasing the accuracy by increasing the expansion order. These advantages are extendable not only inside the WP3 framework, but to the extraction stage as well, due to the possibility of easy generation (see deliverable D1.2b [28] for details).

Due to all these facts, the Taylor Series was proposed as the basis for the representation of the parametric Model Order Reduction inside WP3, as well as the interface with the other WP's. Such a representation was later validated by joint work on WP3 and other WP's and is now accepted as the base representation for parameterized passive systems.

Therefore, the general representation of the parametric effects in the state space descriptor may be given by a generalized Taylor Series (multi-variable) expansion of the matrices C and G

$$\begin{aligned} G(l_1,\dots l_P) &= \sum_{i_1=0}^{\infty}\dots\sum_{i_P=0}^{\infty} G_{i_1,i_2,\dots,i_P}\cdot l_1^{i_1}\dots l_P^{i_P} \\ C(l_1,\dots l_P) &= \sum_{i_1=0}^{\infty}\dots\sum_{i_P=0}^{\infty} C_{i_1,i_2,\dots,i_P}\cdot l_1^{i_1}\dots l_P^{i_P} \end{aligned} \qquad \textbf{(3)}$$

where $G_{i_1,i_2,\dots,i_P}$ and $C_{i_1,i_2,\dots,i_P}$ are the sensitivities respectively of the G and C matrices with respect to the modelled parameters. As an example, of significant relevance as we shall see further on, the first order representation is

$$\begin{aligned} G(l_1, l_2\dots, l_P) &= G_0 + \Delta l_1\cdot\frac{\partial}{\partial l_1}G + \Delta l_2\cdot\frac{\partial}{\partial l_2}G + \dots + \Delta l_P\cdot\frac{\partial}{\partial l_P}G \\ C(l_1, l_2\dots, l_P) &= C_0 + \Delta l_1\cdot\frac{\partial}{\partial l_1}C + \Delta l_2\cdot\frac{\partial}{\partial l_2}C + \dots + \Delta l_P\cdot\frac{\partial}{\partial l_P}C \end{aligned} \qquad \textbf{(4)}$$

where G_0 and C_0 are the nominal values for the matrices, i.e. the value of the matrices when the parameters take their nominal values, that is when there is a zero variation of the P parameters, $\Delta l_i = 0$. In the previous expression $\frac{\partial}{\partial l_i}G = G_i$ and $\frac{\partial}{\partial l_i}C = C_i$ are thesensitivities of *G* and *C* with respect to the i-*th* parameter

This formulation leads to an explicit parametric dependence for the state space descriptor of the type:

$$\begin{aligned}&\left((C_0+\Delta l_1\cdot C_1+\ldots+\Delta l_P\cdot C_P)\cdot s+(G_0+\Delta l_1\cdot G_1+\ldots+\Delta l_P\cdot G_P)\right)\cdot x(s,l)=B\cdot u(s)\\&y(s,l)=L\cdot x(s,l)+D\cdot u(s)\end{aligned} \tag{5}$$

Of course this representation is a first approximation of the parametric system in (1), but it reveals the advantages of the representation, namely that the accuracy can be controlled via the expansion order of the Taylor Series w.r.t. the parameters. Another additional advantage is that the application of projection techniques on this kind of systems leads to systems of similar characteristics, as shown in the next equation

$$\begin{aligned}&\left(V^T\cdot(C_0+\Delta l_1\cdot C_1+\ldots+\Delta l_P\cdot C_P)\cdot V\cdot s+V^T\cdot(G_0+\Delta l_1\cdot G_1+\ldots+\Delta l_P\cdot G_P)\cdot V\right)\cdot x(s,l)=V^T\cdot B\cdot u(s)\\&y(s,l)=L\cdot V\cdot x(s,l)+D\cdot u(s)\end{aligned} \tag{6}$$

and therefore

$$\begin{aligned}&\left((\hat{C}_0+\Delta l_1\cdot\hat{C}_1+\ldots+\Delta l_P\cdot\hat{C}_P)\cdot s+(\hat{G}_0+\Delta l_1\cdot\hat{G}_1+\ldots+\Delta l_P\cdot\hat{G}_P)\right)\cdot x(s,l)=\hat{B}\cdot u(s)\\&y(s,l)=\hat{L}\cdot x(s,l)+D\cdot u(s)\end{aligned} \tag{7}$$

where $\hat{C}_i=V^T\cdot C_i\cdot V$, $\hat{G}_i=V^T\cdot G_i\cdot V$, $\hat{B}=V^T\cdot B$ and $\hat{L}=L\cdot V$, are the projected state-space elements. Clearly, analysis of (5) and (7) immediately shows that the parameter dependence is preserved in the exact same manner, and that the reduced system has exactly the same representation, making it easy to reuse inside a chosen environment or as an input for additional reduction steps.

Therefore from a usage perspective, the Taylor Series formulation seems to be an appropriate representation basis when considering the objectives of Task 3.1, namely those related to parametric Model Order Reduction. On the other hand it is an approximation of the real system, and for this reason it requires validation to determine whether it has enough accuracy in the present context. Stated differently such validation is equivalent to determining what is the order needed of the expansion to retain enough accuracy under the foreseen parameter variation range. This validation is going to be the main objective of the next section. It is important to notice that the Taylor Series expansion may not be necessarily the approach best applied in order to obtain an explicit parametric dependant system as in (5). A more general *multivariate polynomial* expansion may be used in order to improve the accuracy over the whole range of variation. The difference is that, as a result of such a fitting procedure, the values for the *C* and *G* matrices under zero variation may differ form the values of the real nominal matrices (which is not the case for a real Taylor Series). Nevertheless, such a difference is expected to be very small, so in the document the term Taylor Series expansion will be used as it gives a more intuitive idea of the procedure.

4.3 Validation of the Pursued Representation

The Taylor series expansion, as mentioned above, provides an adequate and also an intuitive means of incorporating variability into the model descriptions. The essential idea is that the series can be truncated after a given number of terms. The number of terms then effectively determines the order of the approximation. For a zero-th order expansion, only a single term is retained, which is just a constant matrix, the nominal matrix, and this expansion essentially results in a model where variability is not accounted for. A first-order expansion consists of a constant matrix plus a series of terms containing the sensitivities, in a univariate form, as in (5). Higher order expansions are more complicated, and contain multivariate terms. From the perspective of reduction, however, handling of expansions up to any order is transparent, as we shall see later on.

Obviously, using first order expansions will lead to the simplest models, which still account for the effects of variability. Simple simulations, performed in the context of WP2 and WP3, have shown that first order expansions lead to acceptable models in terms of accuracy, i.e., models which capture the relevant behavior of the underlying system subject to parameter variations and to a sufficient accuracy. Such a representation is therefore the one proposed in order to achieve the goals of this project. We should however point out that it is not trivial to assure that in general such models will be adequate, and not too simple.

In the rest of this Section we will report the results of a set of studies on the effects that variations have on several systems. The objective here is to validate the accuracy of the first order approximation in our chosen approach in the different levels of approximation envisaged inside CHAMELEON RF, to know if this affine (first order) approximation fulfills the expectations, or whether, on the other hand, we need t shift up to second or higher order to maintain a desired accuracy, and if this is the case, under which conditions should we do it.

Simple Test-Structures

In order to get a good level of confidence in our proposed first order modeling approach, a large number of experiments have been performed for simple test-structures. For each of these, extraction of the system matrices *C* and *G* was undertaken following the methodology we now describe. First of all, for each test-structure, a parameterized model was fitted onto the extracted results. The fitting was performed by varying the parameters of the test-structures. Then, in a second step, random parameter values were imposed on the test-structures, after which again the *C* and *G* matrices were extracted. The numerical values contained in these matrices were then compared to the ones in the parameterized model obtained from the fitting procedure.

As mentioned, many extractions were performed in order to validate our assumption. Figure 3 gives an indication of the kind of structures used. For each structure, the coupling capacitance and mutual (partial) inductance between two random conductors was computed, as well as the ground-capacitance and self-inductance of a single random conductor.

It should be noted that the structures are relatively small, and that the capacitance and inductance are lumped to single components. For larger structures, more lumped components are needed to accurately describe the circuit, but the principle does not change. The reason is that the significant effects of variability are emanating from local couplings only. In other words, small geometrical changes will in principle not lead to significant changes in large distance couplings, either capacitive or inductive. In order to see that this is true, one should keep in mind that of course only relative changes are of interest. Thus, for larger (more realistic) structures, the errors are likely to become smaller.

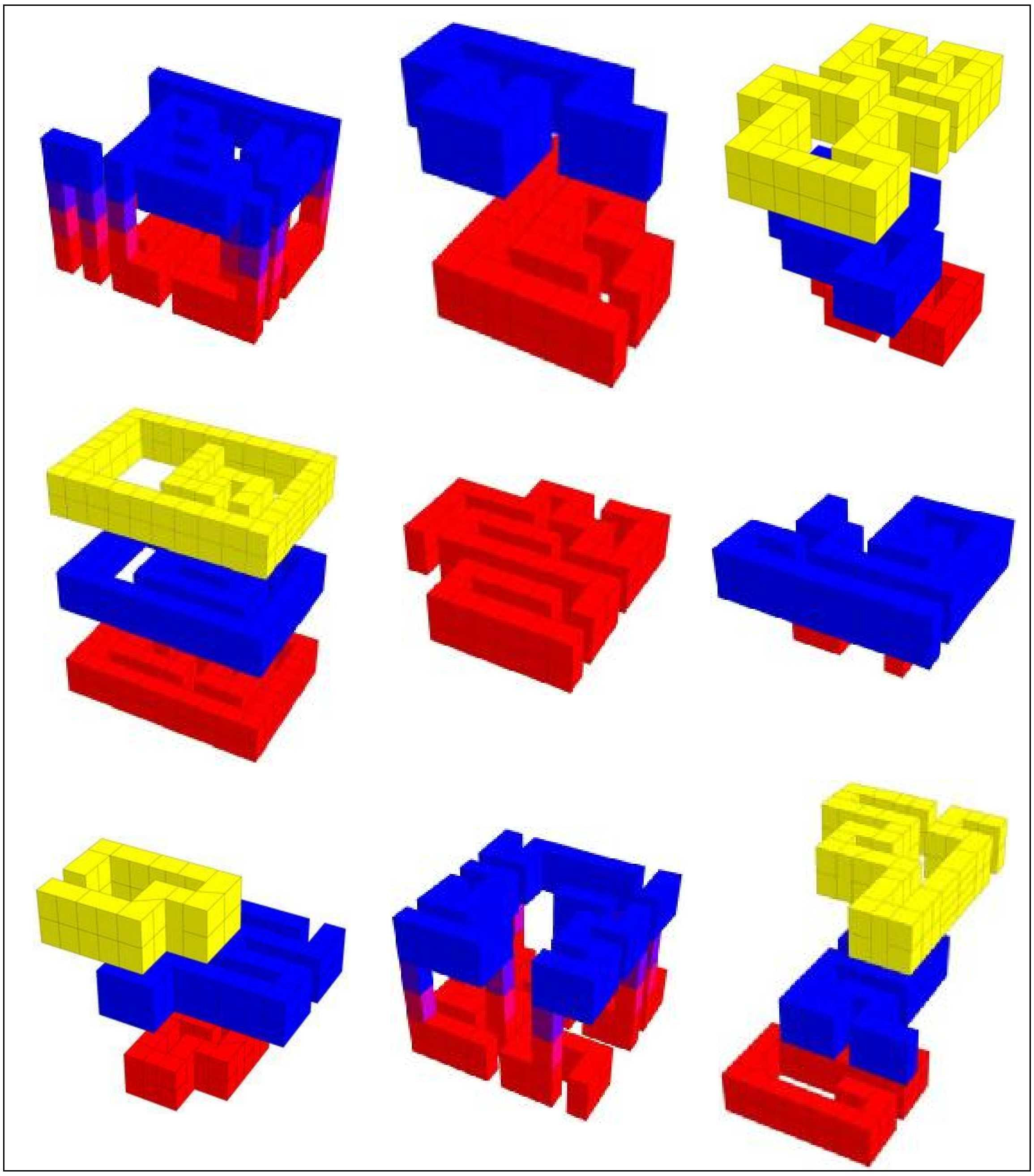


Figure 2. Examples of the test-structures used for the validation of the first-order model approximation.

It should also be noted that no resistance extraction was performed, as it can be easily shown that the resistive part of the system matrices can be accurately modeled using first-order expansions only. Consider the cross-section of a metal conductor in Figure 4 (left). The variables p_1 and p_2 denote the amount of variation in the dimensions of the metal, and we may assume that their values lie between 0 and 0.2. For the conductance Y of the metal conductor, we have the following relation:

$$\frac{Y}{Y_{nom}} = (1 + p_1)(1 + p_2) = 1 + p_1 + p_2 + p_1 p_2$$

where Y_{nom} is the nominal conductance of the conductor. The last term in the above expression is the nonlinear (second-order) part. If we ignore it, an error would be introduced of size:

$$err = \frac{p_1 p_2}{(1 + p_1)(1 + p_2)} \le \frac{p^2}{(1 + p^2)}$$

where we have defined p = max(p_1, p_2). A plot of (the upper bound of) the error as a function of p is shown in Figure 4 (right). With 20% variability, the error is still bounded below 3%, hence the approximation may be considered valid.

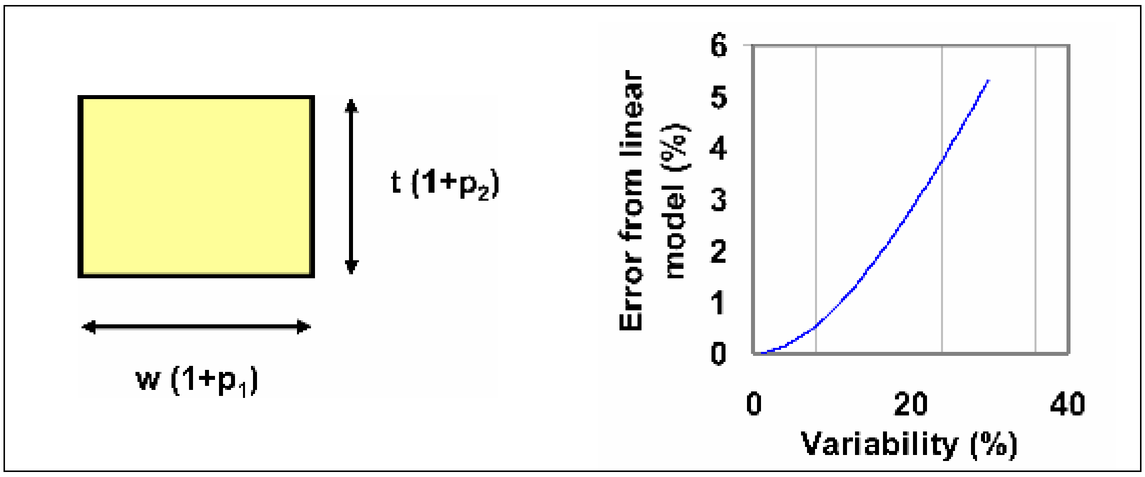


Figure 3. The cross-section of a conductor, where the dimensions have been parameterized by p_1 and p_2 (left). An upper bound of the error in the conductance of the conductor resulting from a linearized model (right).

Figure 5 shows a plot of the behavior of the coupling capacitance between two conductors, when various parameters are changed. For all parameters, the dependence is close to linear, when the parameter variation is within 20%.

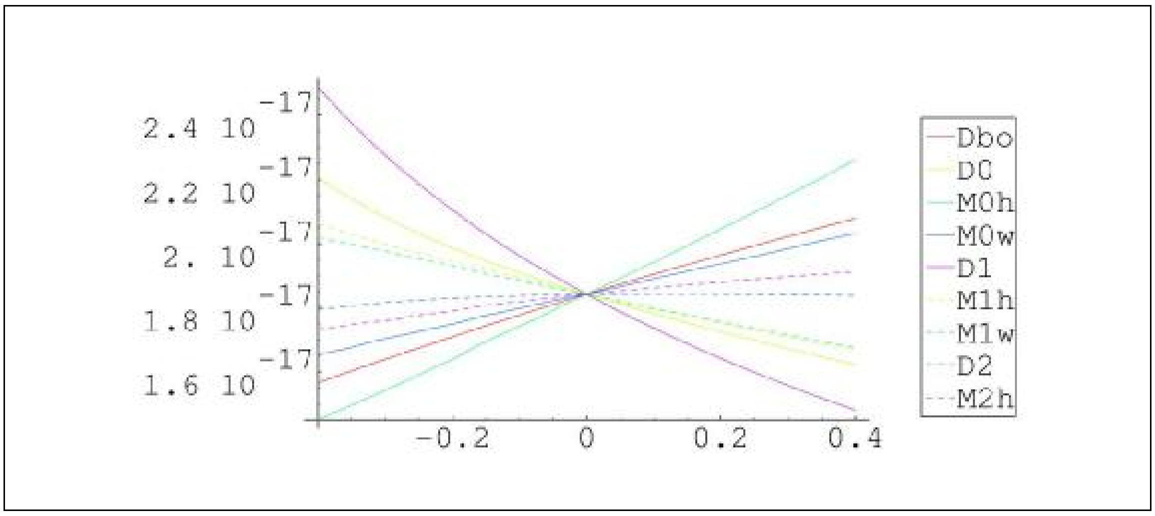


Figure 4. Plot of the coupling capacitance between two conductors for varying parameters, versus normalized parameter value. Each varying parameter corresponds to a different line. The 0 on the x-axis corresponds to the nominal parameter value.

Figure 6, as a reference, shows the errors in coupling capacitance caused by a zero-th order model, when the parameters are varied randomly by 20% (a uniform distribution was used). Thus, in essence, this figure shows the errors incurred by neglecting variability. The average error is 13%. And while 50% of the samples have an error smaller than 11%, this approximation can obviously not be considered acceptable.

Figure 7 shows the same plot for a first-order approximation. In this case, the average error is 1.66%. Furthermore, 95% of the samples have an error of less than 5.11%, and the remaining 5% of the samples have an error of less than 15.3%. Deeper inspection showed that these outliers are generally related to very small (and hence practically insignificant) absolute values of the coupling capacitance, or that they are caused by some infrequent meshing problems in the simulator.

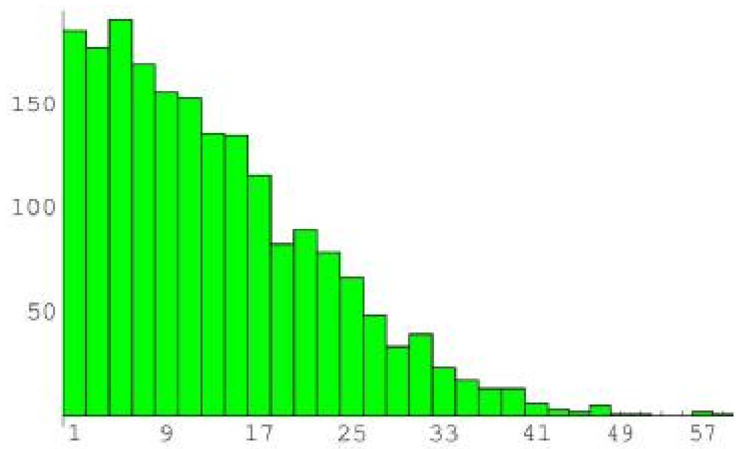


Figure 5. A histogram of the errors in the coupling capacitance caused by a zero-th order model. The vertical axis shows the number of samples in each class, and the horizontal axis shows the error (in percentage) of each class. The average error is 13%. A total number of 1950 samples were computed for this plot.

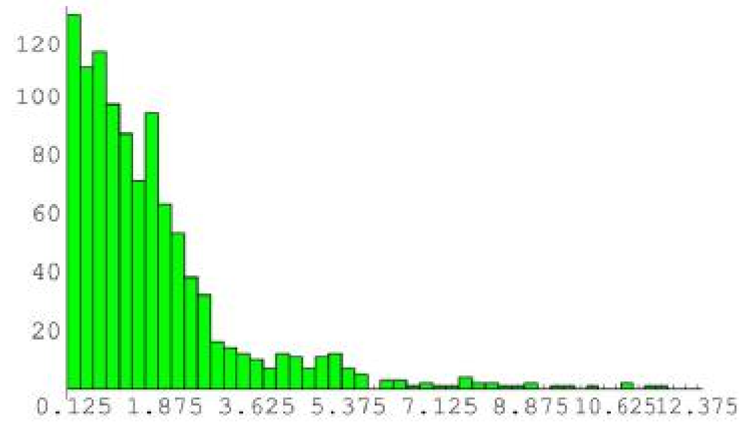


Figure 6. A histogram of the errors in the coupling capacitance caused by a first order model. The average error is 1.66%. A total number of 1050 samples were computed for this plot.

Figure 8 shows the same (first-order approximation) plot for the self-inductance. Note that the errors are even lower in this case. The average error is below 0.3%, and all errors are below 2%.

A plot for the ground (self) capacitance is omitted here, as the manifestation of this effect is mostly similar to the coupling capacitance. Also, a plot for the mutual inductance is omitted, as the errors caused by this effect are even below those of the self-inductance.

With these values of the error, we can confidently claim that a first-order approximation is likely valid for the types of structures and analysis envisioned in the context of CHAMELEON RF. For comparison and completeness, we have also computed the error distribution of the second-order approximation of the coupling capacitance (see Figure 9). As can be seen, by switching to a second order model, the average error can be reduced to less than 0.5%. However, such a model is not of great interest in practice, since variability can be considered a secondary effect in itself, and thus these errors are not as important as the errors in the nominal values of the parasitics.

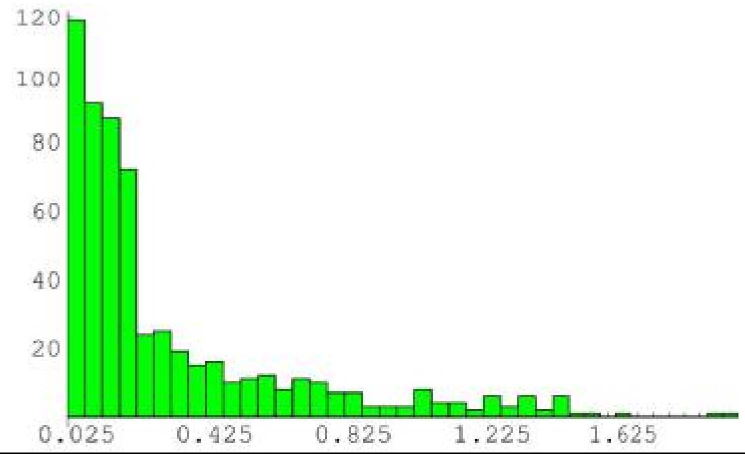


Figure 7. A histogram of the errors in the self-inductance caused by a first order model. The average error is 0.297%. A total number of 600 samples were computed for this plot.

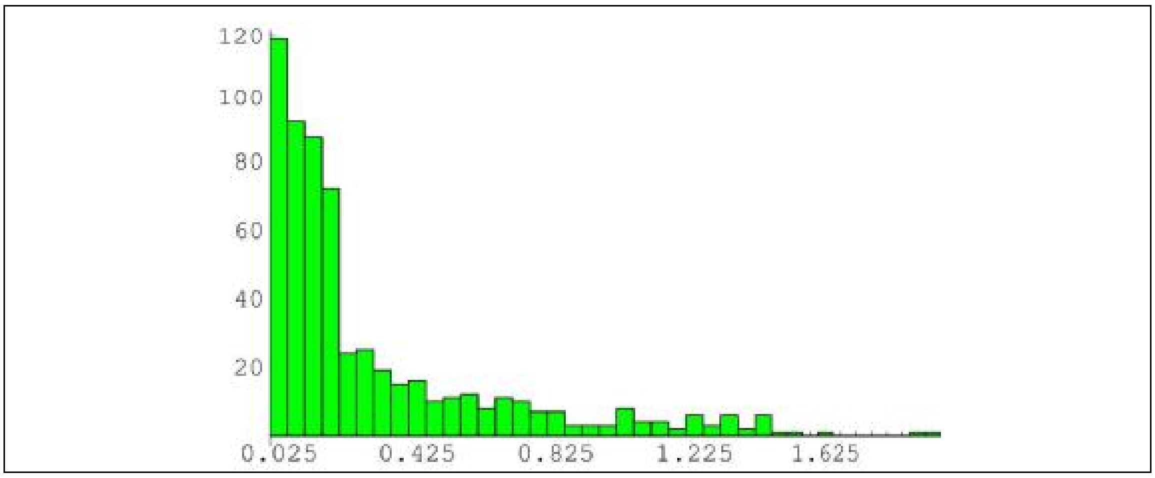


Figure 8. A histogram of the errors in the coupling capacitance caused by a second order model. The average error is 0.495%. A total number of 1100 samples were computed for this plot.

Simple Benchmark - Single Parameter

The proposed validation data shown above is based on statistical analysis and uses a multivariate polynomial expansion. The results show that the average error for first order models is very low, and this justifies the use of a Taylor Series (TS) representation.

Here we present a more realistic validation of the TS representation by comparison of the computed output quantity (e.g. impedance, admittance, or other parameters) in three different ways:

First, using simulation results given by a field solver (e.g. FIT) applied to the system with varied parameters, namely a new discretized model is generated for each set of parameters. This will be taken as the golden solution and reference for comparison. Then

$$y(l,w) = L\left(j\,w\,C(l) + G(l)\right)^{-1} B\,u \tag{8}$$

Second, the sensitivity matrices of the Taylor Series are computed using the Direct Differentiation Method (presented in D1.2), and the approximation of the parameterized system is done using the TS for descriptions based on state space matrices:

$$C_{TS} = C_0 + \frac{\partial C}{\partial l}\left(l - l_0\right),\; G_{TS} = G_0 + \frac{\partial G}{\partial l}\left(l - l_0\right),$$
$$y(l,w) = L\left(j\,w\,C_{TS} + G_{TS}\right)^{-1} B\,u \tag{9}$$

where $C_0 = C\left(l_0\right)$, $G_0 = G\left(l_0\right)$ are the matrices computed for the nominal values of the parameters

Third, the same sensitivities as in the previous case are used, but in this case applied in an approximation of the Taylor Series for the output

$$y(l) = y(l_0) + \frac{\partial y}{\partial l}\left(l - l_0\right) \tag{10}$$

where the sensitivity of the output quantity can be computed from the sensitivities of the state space matrices as follows:

$$y(l_0) = Lx \text{ and } \frac{\partial y}{\partial l} = L\frac{\partial x}{\partial l}, \tag{11}$$

where $\frac{\partial x}{\partial l} = -\left(j\,w\,C + G\right)^{-1}\left[\left(j\,w\frac{\partial C}{\partial l} + \frac{\partial G}{\partial l}\right)x\right],\ x = \left(j\,w\,C + G\right)^{-1} B\,u$

It is important to notice that this last case, although using the same sensitivities as in the second case, is a linear approximation of the output, whereas the second case is a linear approximation of the state-space description, and hence non-linear at the output.

The next example refers to an LShape parameterized conductor (Figure 10). The varying parameter in this case is chosen to be p_2 (see Figure 10). Figure 11 and Figure 12 show the parameter impact on the answer at 3 GHz, i.e. the relative variation of the output (in this case admittance) with respect to the relative variation of the parameter for the system reconstructed using TS. The reference values are the ones obtained using the field solver FIT for each sample individually (first method). First order TS approximation is accurate enough at this frequency, even for a fairly large range of parameter variations. Figure 13 and Figure 14 show the relative error of the output w.r.t the relative variation of the parameter. Interestingly, in this case, sometimes the reconstruction using TS for matrices (second method) is better than the reconstruction using TS for the output (third method), sometimes it is the other way round. Anyway, large variations of the parameter (less than 30 %) lead to small variations of the output (less than 5 %). Figures 15 to 18 show similar results for 60 GHz. In this case, to obtain relative errors of the output less than 5 % using TS, the parameter variation has to be less than 10 %. In conclusion, our findings indicate that first order TS can be used in certain conditions: a specified variation range of the parameter and a specified frequency range. Of course, this statement varies from device to device, as the parametric variation can have very different effects depending on the topology and the characteristics of the element under variation. Therefore the range of parameter variation for which the model is acceptable changes somewhat from case to case. Nonwithstanding, for the ranges envisaged inside the project, first-order TS seems a viable and accurate option.

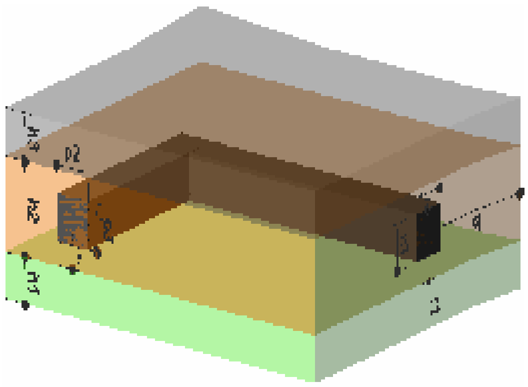


Figure 9. LShape conductor – p2 is the parameter that varies.

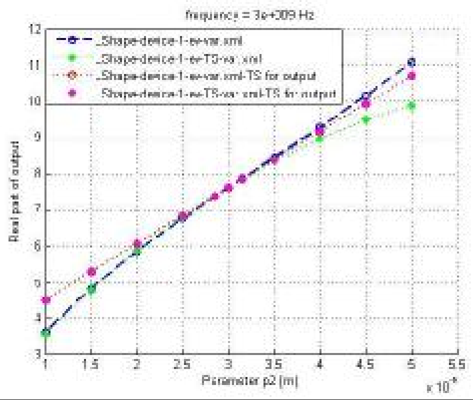
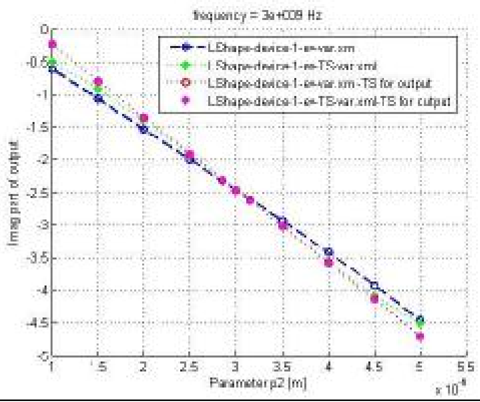
	

Figure 10. Real part of output w.r.t parameter – reference and reconstruction using TS, at 3 GHZ.	**Figure 11. Imaginary part of output w.r.t parameter – reference and reconstruction using TS, at 3 GHZ.**

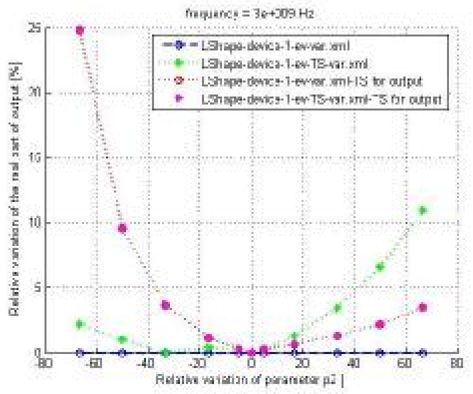
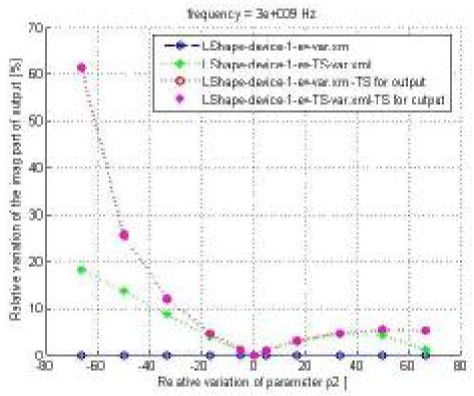

	

Figure 12. Relative difference of the real part of output w.r.t the relative variation of the parameter, at 3 GHz.	**Figure 13. Relative difference of the imaginary part of output w.r.t the relative variation of the parameter, at 3 GHz.**

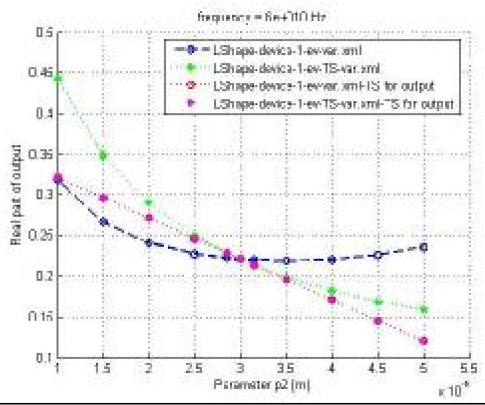


Figure 14. Real part of output w.r.t parameter – reference and reconstruction using TS, at 60 GHZ.

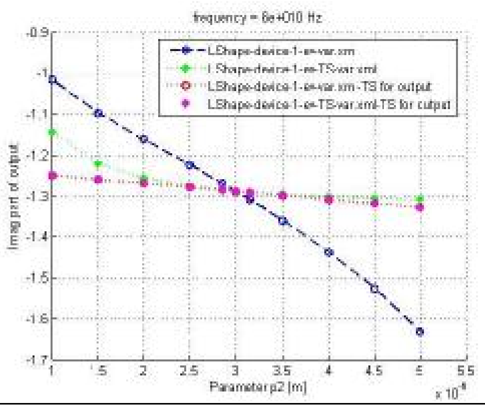


Figure 15. Imaginary part of output w.r.t parameter – reference and reconstruction using TS, at 60 GHZ.

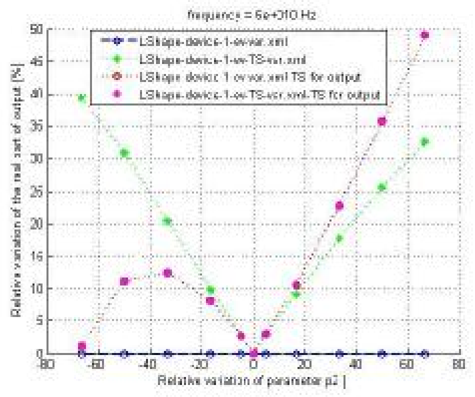


Figure 16. Relative difference of the real part of output w.r.t the relative variation of the parameter, at 60 GHz.

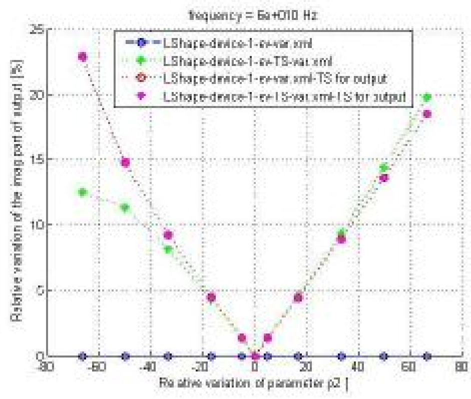


Figure 17. Relative difference of the imaginary part of output w.r.t the relative variation of the parameter, at 60 GHz.

4.4 Representation Standard

Once the model representation is settled, the next natural step is to agree on a standard for its computational descriptor. The idea of having such standard for the representation was based on the need for compatibility with other WPs. Therefore, the format used should be easily read from previous and later simulation steps.

Inside WP3, the common framework for developing and testing of algorithms is the ROM Workbench, a MATLAB toolbox that was started in a previous EC project entitled CODESTAR. The requirements of CHAMELEON RF were not met by the simplified settings envisioned in the available ROM Workbench version, a situation that led to its complete overhaul (see Section 8) and resulted in the creation of the a newer, more powerful and versatile version, entitled the Parametric ROM Workbench. In addition, MATLAB is not the best programming language for the computational needs (in terms of speed and efficiency) of the envisioned algorithms. It was stated that the ROM Workbench inside MATLAB will be useful as a test framework and golden solution for the algorithms to be developed, but not as the platform to be used for actual tool development.

These facts led to the creation of a data format for the models that were going to be the centre of the work in WP3. The basic representation and standard was defined in a general language (XML), meant to be used as the interface with the rest of the CHAMELEON RF data flow and the basis for the model representation. This standard is read into a MATLAB structure, defined with equivalent fields, which is used as the basis inside the new Parametric ROM Workbench, and that can be written, after processing, into a new XML file with the same standard.

The format is based on the Taylor Series representation, being structured in several fields where all the information needed for the mathematical representation is stored. In these structures, not only the Taylor Series information is stored, but also some extra information that is useful for the treatment and reduction of the models. This extra information covers a lot of different aspects of the model, from physical information (e.g. the type of excitation in the outputs, statistical information about the parameters variation, etc) to general (e.g. identification of the model) and even hierarchical information (e.g. the internal structure of the matrices). This standard will be completely presented in Section 8, Appendixes A and B, and in [27].

5 Background in Model Order Reduction

The two most common techniques for Model Order Reduction are based on either truncated balanced realization or projection schemes. For completeness, in the following we review both, describing some of the particulars of each of these. We pay particular attention to the projection schemes, as the techniques we will later describe, are based on projection.

The Balanced Truncation framework [2][3] is based on a balancing transformation of the state space description of the system, so that a direct mapping between an energetic viewpoint of certain input-output characteristics, namely the Controllability and Observabillity, and the states is obtained. The system is therefore prone to be partitioned into two parts, one related to the energetically weak states and other related to the energetically strong states. The energetic measure for the states is given via the Hankel Singular Values. By neglecting the states whose energy falls bellow a desired tolerance, the main behavior of the original system can still be captured, and the error controlled via the energy of the neglected states. This techniques are know for purporting quasi-optimal reduced sizes, but on the other hand they rely on the computation of the Controllability and Observability Gramian, which requires to solve a pair of Layapunov equations and to perform a Eigenvalue Decomposition, which lead to a very high computational cost. For these reasons, TBR is usually not used for large scale systems.

On the other hand, the Krylov techniques are implicit Moment Matching approaches, i.e. are based on the construction of a projection matrix whose columns are obtained from the block moments of the Transfer Function in the frequency domain. The basis that spans such subspace can be iteratively built.

Therefore the Krylov subspace can be efficiently generated via robust and well developed algorithms such as the Block Arnoldi Algorithm [8][9] or the Lanczos Process [7]**.** The most popular Krylov projection algorithm is the PRIMA algorithm [6], where an orthonormal matrix V that spans the Krylov subspace of order q is built and applied via a congruence transformation over the system matrices for obtaining a Reduced Order Model (ROM) of size q that matches the first k block moments of the original transfer function.

A refinement in the previous approach yields the so called Rational, or Multipoint, Krylov Methods [9]. This approach consists in obtaining several Krylov subspaces at several frequency points $s = s_j$ and building a projection matrix from the orthonormal basis of the union of all those subspaces. The Krylov subspace at the chosen frequency $s = s_j$ is obtained by spanning the block moments at that frequency, i.e. the coefficients of the Taylor Series expansion of the Transfer Function around frequency $s = s_j$.

The PRIMA algorithm (and in general any algorithm that applies congruence projection) shows another side advantage. Necessary and sufficient conditions for the system transfer function in (28) to be passive are:

- $\hat{H}(s^*) = \hat{H}^*(s)$ for all complex s, where * is the complex conjugate operator.

- $\hat{H}(s)$ is a positive matrix, that is, $z^{*T} \cdot \left(\hat{H}(s) + \hat{H}^T(s^*)\right) z \geq 0$ for all complex values of s satisfying $\mathrm{Re}(s) > 0$ for any complex vector z.

Any congruence transformation applied to the system matrices satisfies the previous conditions if the original system satisfies them, and so preserves the passivity of the system if the following conditions are true:

- The system matrices are positive definite, $C, G \geq 0$
- $B = L^T$

These conditions are sufficient, but not necessary. They are usually satisfied in the case of electrical circuits, which makes congruence-based projection methods very popular in circuit domains. For further details on passivity see [6].

A drawback of Krylov-based projection techniques is the lack of an efficient technique for error control. Error estimators do exist but are seldom used in practice as they are expensive and cumbersome to use. However the techniques are very efficient and generally produce very good results, which has led to their widespread usage in VLSI settings where reduction of large passive interconnect systems is often required.

A hybrid approach is the Poor's Man TBR (PMTBR) [4][5]. This is a projection MOR technique that exploits the direct relation between the multipoint rational projection framework and the Truncated Balanced Realizations (TBR) [2][3]. This new approach can take advantage of some a priori knowledge of the system properties, and it is based on a statistical interpretation of the system Gramians.

The Gramian can be approximated via a quadrature scheme in the frequency, and if the quadrature rule is accurate, the approximated Gramian, $\hat{X}$, will converge to the exact one, X, which implies that the dominant eigenspace of $\hat{X}$ converges to the dominant eigenspace of X. A Singular Value Decomposition (SVD) can be applied over the subspace obtained in the quadrature, so that a energetic interpretation can be obtained, parallel to the one in the TBR procedures.

The dominant eigenvectors corresponding to the dominant eigenvalues can be used as a projection matrix in a congruence transformation over the system matrices for model order reduction. The eigenvalues can also be used in an *a priori* error estimation in the way the Hankel Singular Values were used in the TBR procedures for error control.

A side advantage is that since this technique is based on a congruence transformation scheme, the passivity preserving conditions are fulfilled as in the case of Krylov methods.

A passive system does not deliver energy and a strictly passive system only consumes energy. A system composed by interconnected passive systems is always passive (this closure property does not hold for stable systems). In the case of electrical circuits, described by linear time-invariant systems (as the ones considered here), passivity is equivalent to the positive realness of the system's impedance and admittance functions, and the scattering parameter matrix of a passive system must have singular values no greater than 1.

The passivity itself is an inherent property of the physical interconnect network, but when these networks are approximated with compact models, to guarantee a passive model in the simulation is a must. The main reason is that a non-passive system, when excited with appropriate stimuli, may present an unbounded response during simulation, thus violating conservation of energy. Physical systems obviously do not behave this way and therefore care must be taken such that all models of such systems show appropriate behaviour. Ensuring that the models are themselves passive is a good step in avoiding such pitfalls. This is applicable also to the MOR techniques, since the reduced order model must maintain the properties of the physical network.

In the projection techniques, the application of a congruence transformation on the system matrices guarantees the positive realness of the transfer function given the mentioned properties of such matrices and, hence, the passivity of the reduced system.

However, under small perturbations, the system can no longer be represented as a traditional state space descriptor, but as a Taylor Series state space descriptor as was shown in section 4.2. But it is necessary to remember that the parametric system is an abstraction of a physical system, where the passive system remains passive whatever the changes the process may have induced over the circuit. Therefore, the parametric model must capture and preserve this characteristic, so we must ensure that our original parametric model approximation as well as the reduced version obtained maintains this physical property of the circuit.

Parametric Model Order Reduction

6.1 Introduction to Parametric Model Order Reduction

Actual fabrication of physical devices is prone to the variation of certain circuit parameters due to deliberate adjustment of the process or random deviations inherent to this manufacturing. This variability leads to a dependence of the circuit extracted elements on several parameters which can have either electrical or geometrical background. After the extraction stage a parametric system is obtained, often represented as a state-space system in its descriptor form as the one shown in (1).

In recent years, some algorithms have appeared in order to deal with this demanding topic. Most of them are extensions or generalizations of former nominal reduction techniques to multi-dimensional variable spaces. A deep review of these techniques was carried out in the early stages of the project, trying to recompile all the possible existing information.

Only the few algorithms we feel will be helpful or important for the flow of CHAMELEON RF will be discussed. They can be split into two major classes: **(Multi-Dimensional) Moment Matching Techniques** and **Sampling-Based Techniques**.

6.2 Multi-Dimensional Moment Matching Techniques

These techniques appear as extensions of former nominal moment matching techniques [6]-[9]. The moment matching algorithms gained a well deserved fame in nominal MOR due to their simplicity and efficiency and in a special place are the Krylov Projection techniques already mentioned. However, they revealed some application problems, especially when trying to obtain some direct relation between model size and error control, being the only possibility *a posteriori* error control, which is awkward to implement.

The extensions of these techniques to parametric cases are usually based on the implicit or explicit moment matching of the parametric transfer function.

$$H(s,l) = L\cdot\left(s\cdot C(l) + G(l)\right)^{-1}\cdot B \tag{12}$$

Obtained from the state space descriptor

$$\begin{aligned} \left(s\cdot C(l) + G(l)\right)\cdot x(s,l) &= B\cdot u(s) \\ y(s,l) &= L\cdot x(s,l) + D\cdot u(s) \end{aligned} \tag{13}$$

Where we have set $D = [0]$ for simplicity and without lost of generality

On the following sections some of the most important and representative parametric moment matching algorithms are presented.

6.2.1 Multi-Parameter Moment Matching

This is a review of several approaches based on Multi-Parameter Krylov subspaces for obtaining a projector matrix and performing a MOR via a congruence transformation as has been shown in previous sections.

These algorithms assume small fluctuations of the parameters, so an affine model based on the Taylor Series expansion is used to approximate the behaviour of the conductance and capacitance, $G(l_1, l_2..., l_P)$ and $C(l_1, l_2..., l_P)$, matrices with respect to the parameters

$$\begin{aligned} G(l_1, l_2..., l_P) &= G_0 + \Delta l_1 \cdot \frac{\partial}{\partial l_1} G + \Delta l_2 \cdot \frac{\partial}{\partial l_2} G + ... + \Delta l_P \cdot \frac{\partial}{\partial l_P} G \\ C(l_1, l_2..., l_P) &= C_0 + \Delta l_1 \cdot \frac{\partial}{\partial l_1} C + \Delta l_2 \cdot \frac{\partial}{\partial l_2} C + ... + \Delta l_P \cdot \frac{\partial}{\partial l_P} C \end{aligned} \quad \textbf{(14)}$$

where G_0 and C_0 are the nominal values for the matrices, i.e. the value of the matrices when the parameters take their nominal value, this is when there is a zero variation of the P parameters, $\Delta l_i = 0$. In the previous expression $\frac{\partial}{\partial l_i} G = G_i$ and $\frac{\partial}{\partial l_i} C = C_i$ are the sensitivities. This Taylor series, here represented for the first order, can be extended up to the desired (or required) order, including cross derivatives.

The simplest idea is to perform a Single-point Expansion [16]. In this approach the parametric transfer function (12) is expanded in a single point in the joint space of the frequency s and the parameters $[l_1, l_2..., l_P]$.

Following the same idea used in the nominal moment matching techniques, a basis V for the subspace formed from these moments can be built. We can use the matrix V as a projection matrix in a congruence transformation for reducing the original system, and this parameterized ROM matches up to the k-*th* order multi-parameter moment of the original system.

The main inefficiency of this method is that process parameters fluctuate in a range much smaller around their nominal value, whereas the frequency range is much bigger, and a higher number of moments are necessary in order to capture the global response all over this frequency range. For this reason, the ROM grows exponentially with the number of parameters and the moments to match.

An upgrade is to perform a Low-Rank approximation of the moments to match [15]. In order to improve the matching, we can apply a 2-norm rank-K_{si} approximation of some generalized moments such as that obtained after doing the Single Value Decomposition (SVD), so that the main behaviours are captured.

For each generalized moment, the subspaces can be computed separately, and combined later to construct an overall projection matrix to be use in a congruence transformation over the parameterized model. The size of the reduced order model can be significantly reduced, because the cross-terms of multi-parameter moments are no longer an issue.

6.2.2 Compact Order Reduction algorithm for parameterized Extraction

This method, as it appears in the literature [17], is based on trying to match the multi-dimensional moments of the transfer function, which as was shown in (12) depends not only on the frequency, but on the parameter variation due to manufacturing uncertainty.

However, despite having the same motivations than the previous method, the development is slightly different. This approach relies on a two-step moment matching scheme that first matches in an explicit way the multi-parameter moments for the process variability parameters l_i, and a second stage where the moments with respect to the frequency are implicitly matched via projection. This two-step approach avoids the exponential growth of the model size with the number of moments matched suffered by the Multi-Parameter Moment Matching method appearing in the previous section.

A parameterized augmented model is obtained after the first step, where the parameter dependence is on the output related matrix L_{AP}. Once the augmented system is built, the implicit moment matching with respect to the frequency s is made via a Krylov standard projection, for example, applying PRIMA [6].

This method allows a certain degree of flexibility as the number of moments matched with respect to the frequency and to the parameters can be different. In principle, despite the bigger size of the augmented model, the final size obtained for the reduced system can be much smaller than in the previous cases. The stability of the ROM is guaranteed, but no conclusions can be drawn with respect to the passivity of the parameterized model.

On the other hand, the structure of the dependence with respect to the parameters is lost since a projection is applied over the augmented system. This is an important drawback, since in this case we cannot recover the original Taylor Series formulation, even though the explicit parameter dependence is not lost.

6.2.3 Passive Parameterized Time-Domain Macromodels

As another multi-dimensional moment matching approach, this technique [18] is based on the computation of multiple subspaces from which a projector is generated. The idea is basically the same as that in previous approaches, but the differentiating factor of this scheme is that the Krylov Subspaces are built separately for each dimension, i.e. the frequency s and the parameter set $[l_1, l_2..., l_P]$.

The first step of the algorithm is to obtain the q_s block moments of the transfer function, i.e. the states x, with respect to the frequency when the parameters take their nominal value. Notice that once the parameters are fixed, the system becomes a non-parametric one, and for this reason any nominal Krylov technique can be applied to obtain the subspace of the moments.

The next step is to obtain the subspace which matches q_{l_i} block moments of x with respect to each parameter l_i. If the variation of the matrices $C(l_1, l_2..., l_P)$ and $G(l_1, l_2..., l_P)$ with respect to the parameter l_i is linear, techniques such as Arnoldi [6] can be applied. If this is not the case, more elaborate techniques [19] must be applied.

In the special case where cross-derivatives play an important role, the subspace of the block moments of x with respect to these cross-derivatives must be obtained.

Once all the subspace have been computed, an orthonormal basis can be obtained so that its columns spans the joint of all subspaces (this can be done for example through a QR decomposition). The matrix can be applied in a congruence projection model reduction over the system matrices, so that it can be assured that the parametric ROM matches q_s moments of the original system with respect to the frequency, and q_{l_i} moments with respect to the parameter l_i. Since the reduction is made via a congruence transformation, the ROM is passive if the original system is passive.

An obvious possibility for improving the accuracy is to obtain a multipoint subspace to match the frequency moments via expansion in different frequency points, i.e. a rational moment matching scheme.

Note that the block moments of x with respect to the parameters can be obtained for parameter values different from zero, if zero is not the nominal parameter value. The value for the frequency $s = s_0$ when obtaining the moments with respect to the parameters, can be fixed to a desired value. A first approach would be to set it to zero, but if so, notice that the sensitivities of the conductance matrix $C(l_1, l_2 ..., l_P)$ would have no effect, so a better option would be to fix it to an arbitrary non-zero value, preferably in the frequency band of interest.

6.3 *Sampling-Based Techniques*

6.3.1 Multi-dimensional Rational Krylov

This method, although being included as a sampling based technique, is a mixture between the two kinds of methods. It is based on the Krylov moment matching techniques, but instead of computing the expansion with respect to all the parameters, this approach proposes to take n_s samples of the perturbed system in the parameter space, so a single sample would be $P_i = [\, l_1 = l_{i1}, l_2 = l_{i2}, ..., l_P = l_{iP} \,]$

For each sample, any Krylov standard (or nominal) method can be used so that a subspace is built. This subspace contains the first k_i moments with respect to the frequency of the parametric transfer function when the parameter sample P_i is applied. Finally, an orthonormal basis can be constructed from the n_s bases, and can be applied as a standard Krylov projector over the parametric system for obtaining a parameterized reduced model. The parameterized reduced model matches k_i moments with respect to the frequency at each parameter sample.

An obvious extension of this approach is not only to sample in the parameter space, but also in the frequency domain in the way the rational Krylov techniques do.

6.3.2 Variational Poor Man's TBR

This work [21] is a generalization of the PMTBR algorithm [4], [5] using the statistical interpretation of the Gramian computation.

The Gramian X can be interpreted as the covariance matrix for a Gaussian variable ($x(0)$) obtained by exciting the ODE with White Noise

And the Gramian can be obtained after applying Parseval's theorem as

$$X_l = \int_{S_l}\int_{\infty}^{\infty} (s\cdot C_l + G_l)^{-1}\cdot B\cdot B^T\cdot(s\cdot C_l + G_l)^{-H}\cdot p(l)\cdot dw\cdot dl \qquad \textbf{(15)}$$

where $p(l)$ is the Probability Density of l in the parameter space S_l

As mentioned in previous sections, PMTBR applies a quadrature rule for approximating the Gramian via numerical computation. The same can be done here, but the weight is given by the ***Probability Density Function*** (PDF) of the parameter l and a frequency constraining. This can be generalized to a set of parameters $[l_1, l_2..., l_P]$, where a PDF of all the parameters can be applied to the joint parameter space, or individual PDF of each parameter can be used.

The first step is to approximate the integral, which can be done by applying Monte Carlo (MC) or Quasi Monte Carlo (QMC) schemes to perform the quadrature. It must be said that, as happens in the nominal PMTBR, the accuracy of the ROM does not depend on the accuracy of the approximation of the integral, but on the projection subspace.

Due to the already mentioned fact that the parameter variability range is much smaller than the frequency range, it is usual to treat the frequency as a special parameter and apply a different sampling method. As happened in the deterministic case, an Error Analysis and Control can be included, via the eigenvalues of the SVD s_k.

In the deterministic case, the error came bounded by:

$$\left\|\hat{x}_0 - x_0\right\|_2^2 \leq \sum_{K=Q+1}^{N} s_k^{\,2}$$

Whereas in the variational case, only an expected error bound can be given:

$$E\left\{\left\|\hat{x}_0 - x_0\right\|_2^2\right\} \leq \sum_{K=Q+1}^{N} s_k^{\,2}$$

The complexity and computational cost is the same as that for the deterministic PMTBR plus the previous quadrature operations, and as can be easily seen, the size of the reduced model does not depend on the number of parameters taken into account.

6.4 Passivity in Parametric MOR

Most of the presented techniques are based on a projection scheme of the system matrices in order to perform the reduction. This has a double advantage. First, the reduced system has the same parametric dependence structure as the original one, which opens several possibilities when included in a simulation environment (see section 4). The second one is that this type of reduction, based on a congruence transformation, guarantees the passivity of the reduced model under certain conditions (see [6] and section 5), that usually are true for electric circuits. This means that if our original Taylor Series system descriptor is passive, the ROM generated will be passive (this is not the case with the CORE algorithm, see section 6.2.2).

6 Parametric Variations in Interconnected Compact Models

In this section we will link the techniques described in the previous section for order reduction of parameterized systems to the ones presented in the deliverable D3.2. As one can easily realize, in the Block Hierarchical System (BHS) scheme the interconnected systems are simply larger systems in their state space description, i.e. the several sub-systems are integrated into the matrices as block diagonals and so are the couplings or hooks (as off-diagonal blocks). For this reason there is no difference between a parametric system and a BHS under parametric variations.

On the other hand the parameters may only affect the blocks related to the system description, but it seems logical that they affect the coupling values as well. The parameters usually reflect geometrical or electrical changes in the system layout that modify their behaviour, so any parameter-induced variation in a device is prone to have an undesired effect in neighbouring devices. This effect is captured via the couplings or the interconnections, i.e. the defined hooks. However, the treatment of these effects can be done via several approaches.

The first one is to suppose that the parameters affect the couplings themselves, and therefore, the off-diagonal entries of the BHS matrices would have a parametric dependence.

Another approach is to assume that the hook effect is captured inside the system, i.e. the hook is treated as an incidence value, representing which nodes of the affected systems are linked in an electromagnetic manner. Since it is envisaged that the hooks will be treated as ports, this seems to be the most natural approach to pursue. In this case, the global system sensitivities would be block diagonal matrices, which would also reflect the effects of the parameter on the devices (i.e. the diagonal blocks of the nominal matrices).

$$C_j = \frac{\partial}{\partial l_j} C(l) = \begin{bmatrix} \frac{\partial}{\partial l_j} C_1(l^1) & & 0 \\ & \mathbf{O} & \\ 0 & & \frac{\partial}{\partial l_j} C_N(l^N) \end{bmatrix}$$

$$G_j = \frac{\partial}{\partial l_j} G(l) = \begin{bmatrix} \frac{\partial}{\partial l_j} G_1(l^1) & & 0 \\ & \mathbf{O} & \\ 0 & & \frac{\partial}{\partial l_j} G_N(l^N) \end{bmatrix} \qquad \textbf{(16)}$$

As we shall both of these approaches can be handled using the procedures to be described.

The aim of this section is to study the feasibility of applying Parametric MOR techniques to the structure preserving scheme presented in D3.2. In such a situation, the system is supposed to have some hierarchical structure so that under parameter variation it can be represented (in a general way) as

$$G(l) = \begin{bmatrix} G_{1,1}(l_{1,1}) & \cdots & G_{1,N}(l_{1,N}) \\ \vdots & \ddots & \vdots \\ G_{N,1}(l_{N,1}) & \cdots & G_{N,N}(l_{N,N}) \end{bmatrix} \quad B = \begin{bmatrix} B_1^T & \ldots & B_N^T \end{bmatrix}^T$$

$$C(l) = \begin{bmatrix} C_{1,1}(l_{1,1}) & \cdots & C_{1,N}(l_{1,N}) \\ \vdots & \ddots & \vdots \\ C_{N,1}(l_{N,1}) & \cdots & C_{N,N}(l_{N,N}) \end{bmatrix} \quad L = \begin{bmatrix} L_1 & \ldots & L_N \end{bmatrix} \tag{17}$$

where $C_{i,j} \in \Re^{n_i x n_j}$, $G_{i,j} \in \Re^{n_i x n_j}$, $B_i \in \Re^{n_i x m}$, $L_i \in \Re^{p x n_i}$, and l is a set of P parameters $[l_1, l_2 ..., l_P]$. $l_{i,j}$ is a set of M parameters affecting that block. The set l contains all the subsets $l_{i,j}$.

Since the BHS has the same structure as a parametric system, it seems possible to use it with any of the previous parametric Model Order Reduction schemes, i.e. Multi-Dimensional Moment Matching or Multi-Dimensional Sampled Based techniques. As was shown in the previous sections, all the parametric techniques that have been considered to be used inside CHAMELEON RF are projection based. These techniques use the parametric variation to compute a suitable subspace able to capture the behaviour of the system under any parametric variation. This subspace does not depend on the parametric variation, nor does any basis whose columns span this subspace. Therefore, after obtaining the orthonormal basis of the computed subspace, it can be used by the Block Hierarchical procedure shown in order to build a projector that preserves the structure and hierarchy of the system when a congruence transformation is applied.

The systems expected in CHAMELEON RF's WP3 are very large, and the global set of parameters may be large (is the union of all the parameters affecting all the sub-systems). Therefore, none of the previous techniques seem to have any advantage over the others, since the high number of parameters may yield an oversize in the Parametric Moment Matching techniques, and the large size of the global system may be a drawback in the computational effort required for the sampling.

A side-advantage of the use of these Block Structure techniques is that the block parameter dependence is also maintained, i.e. if a block depends on a single parameter, it will continue depending only on the same parameter after a Block Structure Preserving projection is applied. Another advantage is the possibility of handling intra and inter-device parameters in a more effective way. The inter-device parameters are those parameters that may affect not only a single device, but also some other devices. These parameters can be handled as a single parameter inside the complete system, and therefore reduce the total number of parameters to treat inside the parametric MOR techniques. On the other hand, the intra-device parameters are those affecting a single device, and therefore the sensitivity will be a very sparse matrix with a single diagonal block with non-zero entries. As has been already pointed out, the Block Structure Preserving Reduction is able not only of maintain the structure of the blocks, but the block parameter dependence as well, so the sensitivity will have a similar level of sparsity after reduction, with non-zero entries only in the affected diagonal block.

$$\hat{G}(l)=\begin{bmatrix}\hat{G}_{1,1}(l_{1,1}) & \cdots & \hat{G}_{1,N}(l_{1,N})\\ \vdots & \ddots & \vdots\\ \hat{G}_{N,1}(l_{N,1}) & \cdots & \hat{G}_{N,N}(l_{N,N})\end{bmatrix}\quad \hat{B}=\left[\hat{B}_1^{\,T} \quad \dots \quad \hat{B}_N^{\,T}\right]^T$$

(18)

$$\hat{C}(l)=\begin{bmatrix}\hat{C}_{1,1}(l_{1,1}) & \cdots & \hat{C}_{1,N}(l_{1,N})\\ \vdots & \ddots & \vdots\\ \hat{C}_{N,1}(l_{N,1}) & \cdots & \hat{C}_{N,N}(l_{N,N})\end{bmatrix}\quad \hat{L}=\left[\hat{L}_1 \quad \dots \quad \hat{L}_N\right]$$

7 Parametric ROM Workbench

The ROM Workbench is a MATLAB toolbox developed under the CODESTAR project, meant to be used as test framework for Model Order Reduction algorithms. At the beginning of CHAMELEON RF it was decided within the consortium that the ROM Workbench would be the common framework for the development and testing of the algorithms to be used. However, the needs of the CHAMELEON RF project extend well beyond those of the CODESTAR project, and this led to significant developments around the former ROM Workbench in order to be useful for the aims envisioned in CHAMELEON RF. These aims pointed into two different although connected paths: one focuses on the variability and their effects in the systems, the other focuses on handling the interactions between the systems. This in turn implied a complete reworking of the inner description of a system, the development of new algorithms for manipulation and reduction of parametrically and interconnected described systems, as well as additional functionality for analysis and evaluation of such systems

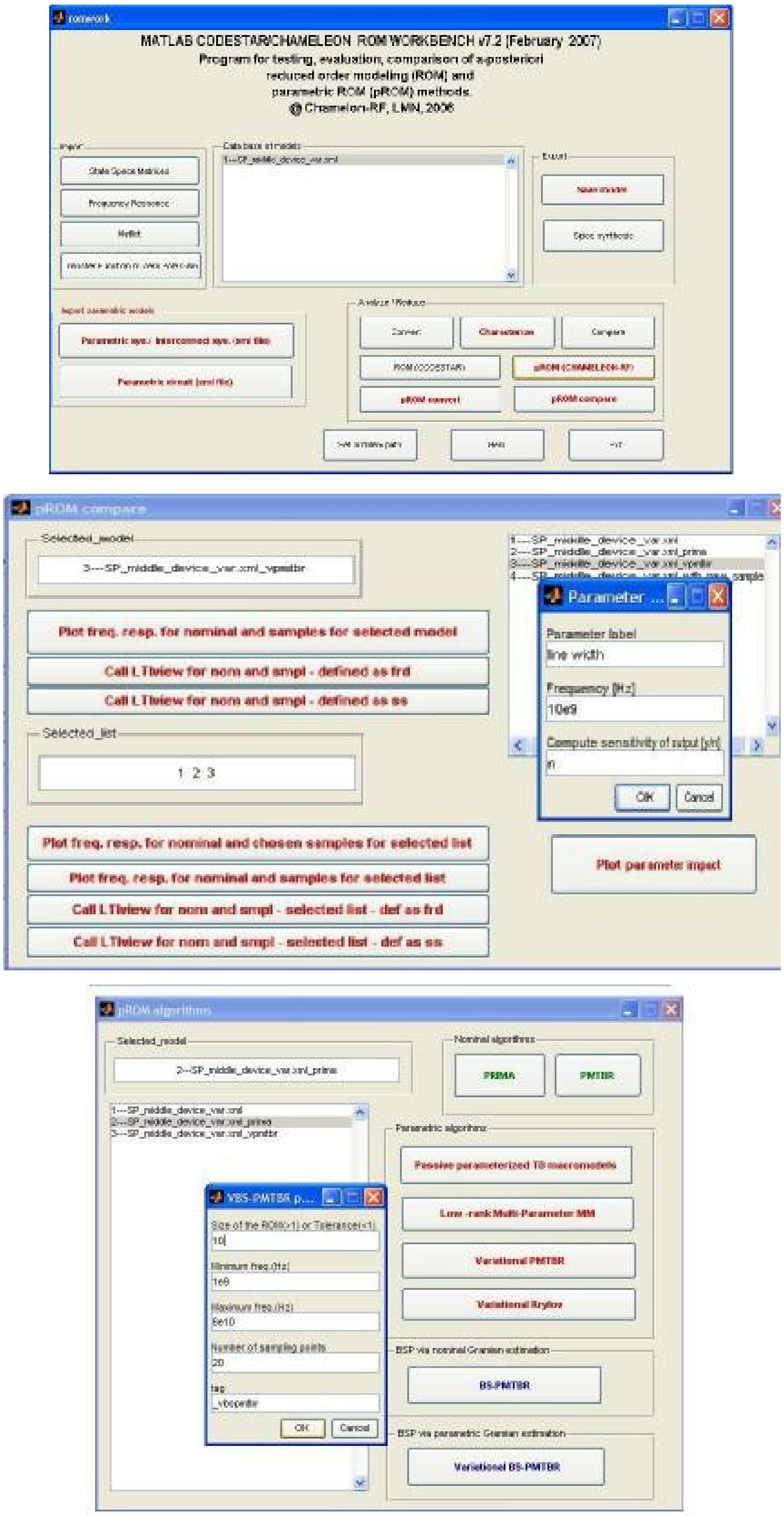
This yielded a complete overhaul, of the ROM Workbench, whose current version is called Parametric ROM Workbench (**pRWB**). The new functionalities were built around the new representation standard in the XML format (see Appendixes A and B). This format includes all the needed information about the parametric system to be stored, including:

- Identifier of the system, identifier of the original system if it is a reduced one, version, type of the representation, number of terminals, type of excitation of the terminals.
- Number of parameters, label of the parameters, nominal values for the parameters, variation ranges and probabilistic information (when available) about the variation of the parameter.
- Nominal system matrices (stored in files), sensitivities and information about them, sampled matrices, projector used in the reduction (if done), block hierarchical information.
- Frequency data for the nominal and the samples and sampling values.

Inside the new pRWB all of the former functions were maintained as well as the compatibility with former procedures and algorithms, and new ones were added in order to handle the new types of systems, including:

- Reading from XML, writing into XML.
- System properties study: sparsity, stability, etc…
- Conversion from interconnected to standard system.
- Parametric and Block Structure Preserving MOR algorithms.
- Representation, comparison and measurement functions: frequency based, parameter impact, time domain, etc…

The pRWB is able to use several parametric MOR algorithms and combine them with Block Structure Preserving MOR techniques for the reduction of interconnected systems. The next figures show some snapshots of the main windows of the pRWB. A detailed description of the functions implemented is given in [26].


MATLAB CODESTAR/CHAMELEON ROM WORKBENCH v7.2 (February 2007)
Program for testing, evaluation, comparison of a-posteriori
reduced order modeling (ROM) and
parametric ROM (pROM) methods.
@ Chameleon-RF, LMN, 2006

Selected_model: 3---SP_middle_device_var.xml_vpmtbr

Selected_list: 1 2 3

Parameter label: line width
Frequency [Hz]: 10e9
Compute sensitivity of output [y/n]: n

Selected_model: 2---SP_middle_device_var.xml_prima

Size of the ROM(>1) or Tolerance(<1): 10
Minimum freq.(Hz): 1e9
Maximum freq.(Hz): 5e10
Number of sampling points: 20
tag: _vbspmtbr

romwork
State Space Matrices
Frequency Response
Netlist
Data base of models
1---SP_middle_device_var.xml
Export
Save models
Spice synthesis
Import parametric models
Parametric sys./ interconnect sys. (xml file)
Parametric circuit (xml file)
Analyze / Reduce
Convert
Characterize
Compare
ROM (CODESTAR)
pROM (CHAMELEON-RF)
pROM convert
pROM compare
Help
pROM compare
Plot freq. resp. for nominal and samples for selected model
Call LTIview for nom and smpl - defined as frd
Call LTIview for nom and smpl - defined as ss
1---SP_middle_device_var.xml
2---SP_middle_device_var.xml_prima
3---SP_middle_device_var.xml_vpmtbr
Parameter ...
OK
Cancel
Plot freq. resp. for nominal and chosen samples for selected list
Plot freq. resp. for nominal and samples for selected list
Call LTIview for nom and smpl - selected list - def as frd
Call LTIview for nom and smpl - selected list - def as ss
Plot parameter impact
pROM algorithms
Nominal algorithms
PRIMA
PMTBR
Parametric algorithms
Passive parameterized TB macromodels
Low -rank Multi-Parameter MM
Variational PMTBR
Variational Krylov
BSP via nominal Gramian estimation
BS-PMTBR
BSP via parametric Gramian estimation
Variational BS-PMTBR
VBS-PMTBR p...

Figure 18. Windows of the pRWB

8 Simulation Results

In this section we present interesting results of the simulations performed within pRWB over real devices, similar to the ones envisioned relevant in the context of CHAMELEON RF. Additional tests were carried out that are not shown. However these results have been taken into account for the conclusions and the decisions taken in the basis to this deliverable.

In the next two sub-sections we show the results of pMOR techniques over a single device dependent on one single parameter, in order to compare the performance of the several algorithms, both parametric and nominal. This will be useful to validate the performance of the algorithms and to show the need for using the parametric techniques.

L Shape

The first example to use is going to be an L-Shape, with one input and one output (SISO), 313 degrees of freedom and one parameter (*p2*) as was shown in Figure 10 (section 4.3). Figure 19 shows the 2D view of the example.

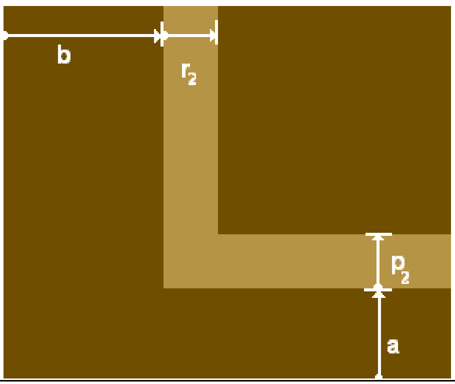
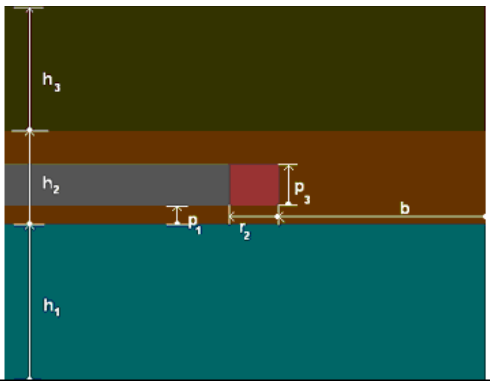
	
L Shape – 2D view (xOz)	L Shape – 2D view (xOy)

Figure 19. LShape – Layout

We use this benchmark to test the parametric algorithms presented and some of the nominal techniques (PRIMA [6] and PMTBR [4]) when applied to parametric systems.

Figures 20 and 21 show, respectively, the frequency response and the absolute error for the nominal system, the original (i.e. Taylor Series) perturbed system, and the ROMs. It can be seen that PRIMA and PMTBR are unable to capture the parametric behaviour. The Lowrank algorithm is inefficient for this problem, as it needs a bigger ROM in order to capture the multi-parameter moments, and requires heavier computations. The rest of the algorithms capture the variational behaviour with acceptable accuracy even for large variational ranges. Figure 21 shows the absolute error only for the parametric algorithms (PPTDM, VPMTBR, VKRYLOV and CORE). It can be seen that CORE loses accuracy at high frequencies because it needs more moments.

Figures 22 and 23 show the variation of the output w.r.t. the variation of the parameter for a fixed frequency, for the nominal (i.e. Taylor Series) and several reduced models.

The *nominal* algorithms (i.e. PRIMA and PMTBR), are not shown, but their response is only accurate for the nominal value of the parameter. The Lowrank algorithm is able to capture the response for small variations. The rest of the algorithms capture the variational response for all the variation range (up to 66% of variation) with enough accuracy. Figure 23 shows the relative error for these parametric algorithms. It can be seen that the largest relative error is less than 15% with a 66% variation for all the algorithms with the exception of the Lowrank algorithm.

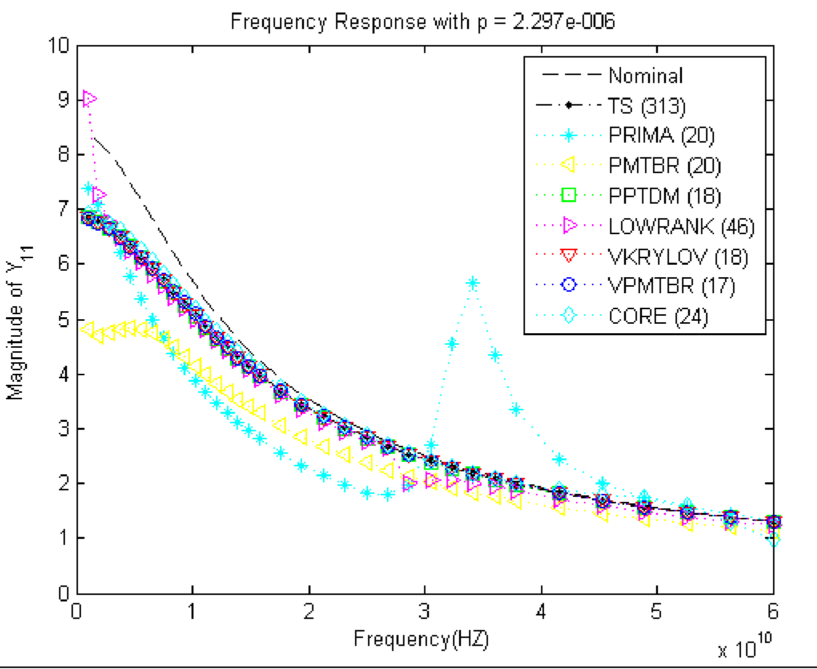


Figure 20. L Shape – Frequency Response of all the Algorithms for a random parameter value

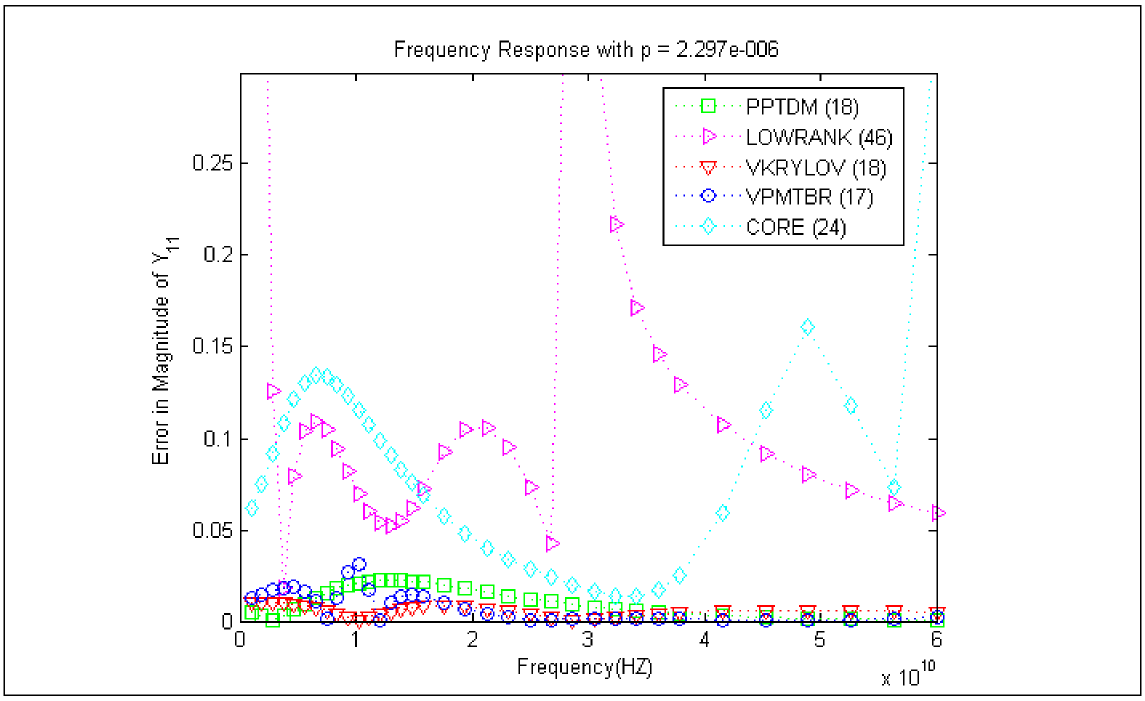


Figure 21. L Shape – Absolute Error in the Frequency Response of the parametric Algorithms for the same random parameter value

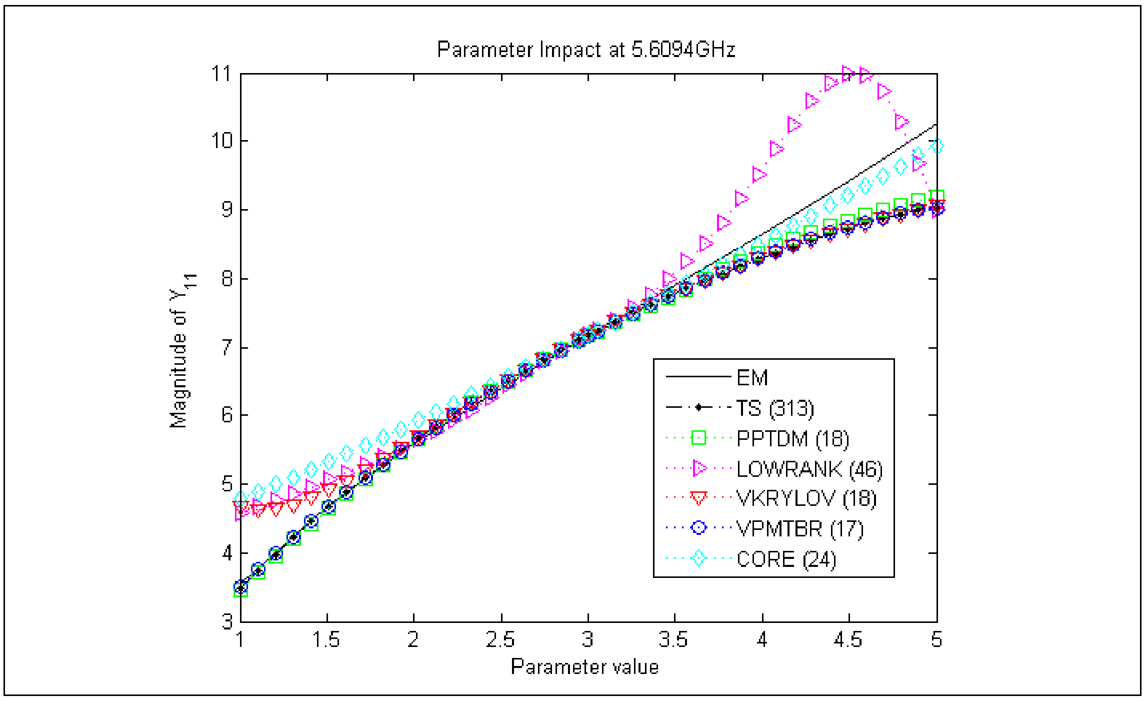


Figure 22. L Shape – Parameter Impact for the parametric Algorithms at a random frequency of 5.6GHz.

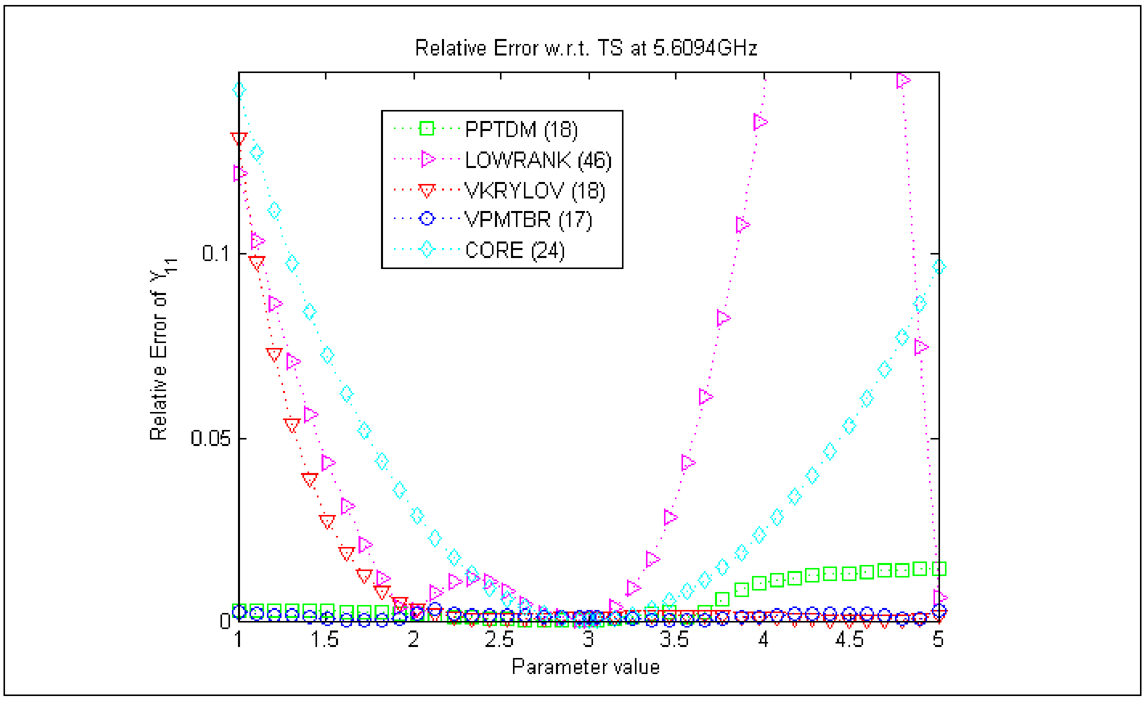


Figure 23. L Shape – Relative Error w.r.t. the Taylor Series versus the Parameter Variation for the parametric Algorithms at a random frequency

Spiral

This is a more complex and realistic example that allows one to see how the parametric algorithms behave in real benchmarks. The example is the Spiral Inductor shown in the Figure 24, described with a dependence on one parameter (in this case the side length; see Figure 24 right), with two inputs and two outputs (2-MIMO) and 4961 degrees of freedom.

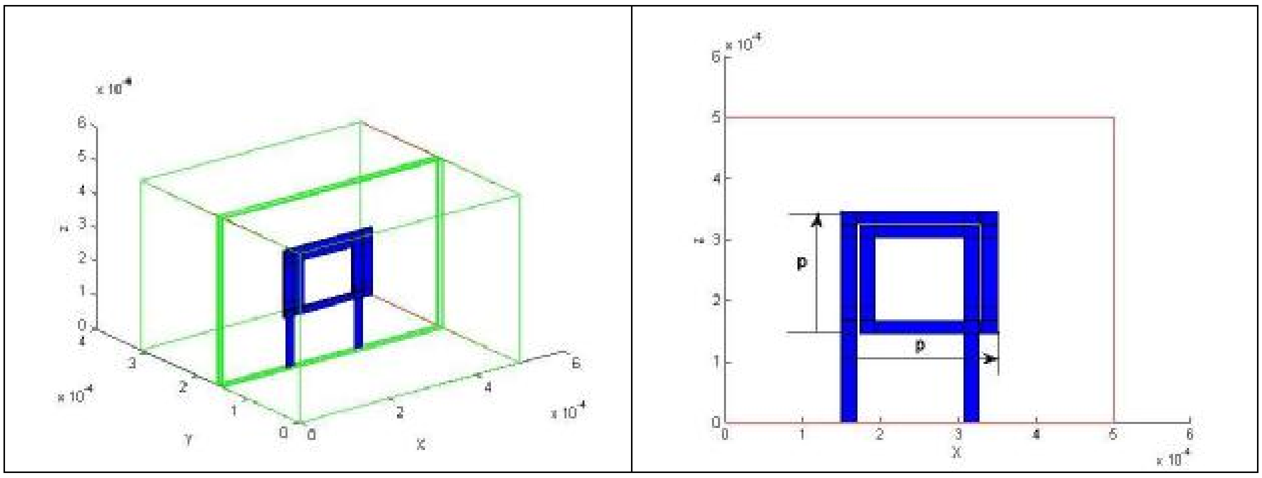


Figure 24. Spiral Layout and Parameter (p) to vary

In this case the nominal algorithms do not perform well for any size variations, and some of the parametric MOR methods do not work well either: The Lowrank algorithm needs heavy computations and runs out of memory, and the PPTDM algorithm is not competitive as it is unable of matching the parametric response with a similar size to the rest of the algorithms.

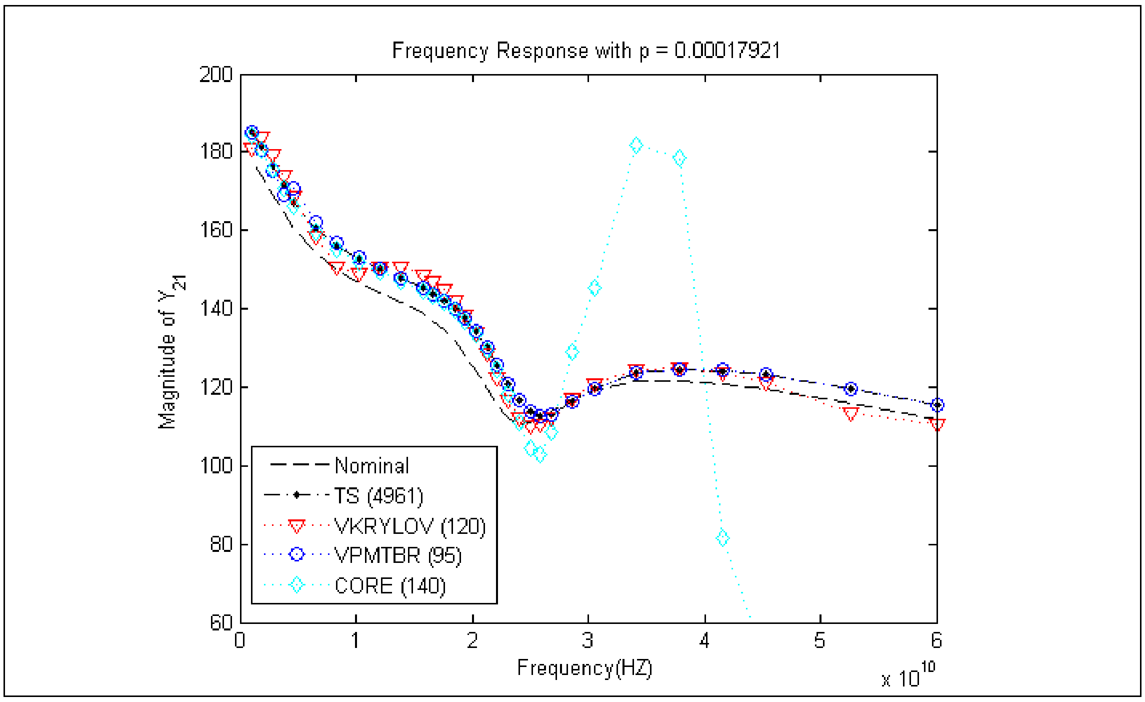


Figure 25. Spiral – Frequency Response of the parametric Algorithms for a random parameter value.

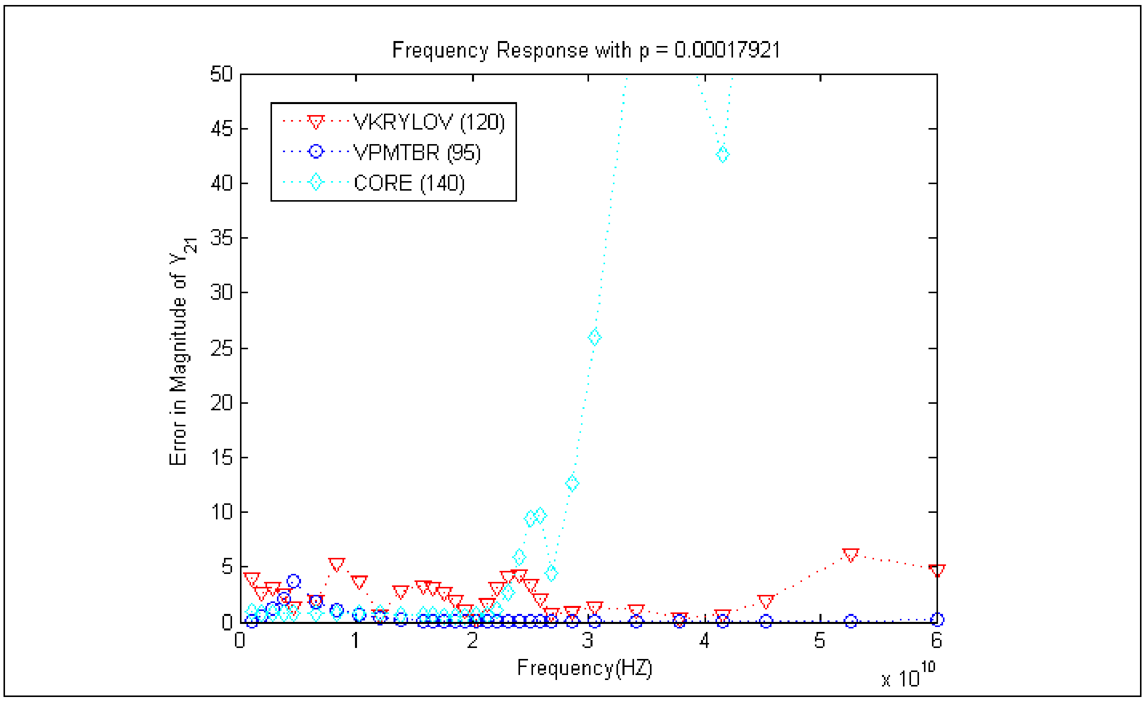


Figure 26. Spiral – Absolute Error in the Frequency Response of the parametric Algorithms for a random parameter value.

The rest of the parametric algorithms (Variational Krylov, Variational PMTBR and CORE) are able to capture the parametric response with different success levels. This can be seen in Figures 25 and 26 that show the frequency response and absolute error for these parametric algorithms. CORE needs a larger size to capture the behaviour at high frequency. The rest manages to maintain a lower error in the complete frequency range.

Figures 27 and 28 show the variation of the output w.r.t. the variation of the parameter for a fixed frequency and the relative error for the same parametric algorithms (VPMTBR, VKrylov and CORE). It can be seen that in this case (a lower frequency) CORE maintains its linear behaviour for any variation, being able to capture the response with enough accuracy. The Sampling based algorithms may lose accuracy at some parameter values due to the sampling (the random sampling may fail to capture some behaviour if the number of samples does not cover the dimensional space), but in general they give a better overall performance in the whole frequency range. In Figure 38 it can be seen that the relative error is less than 10% for the larger variation (40% of variation).

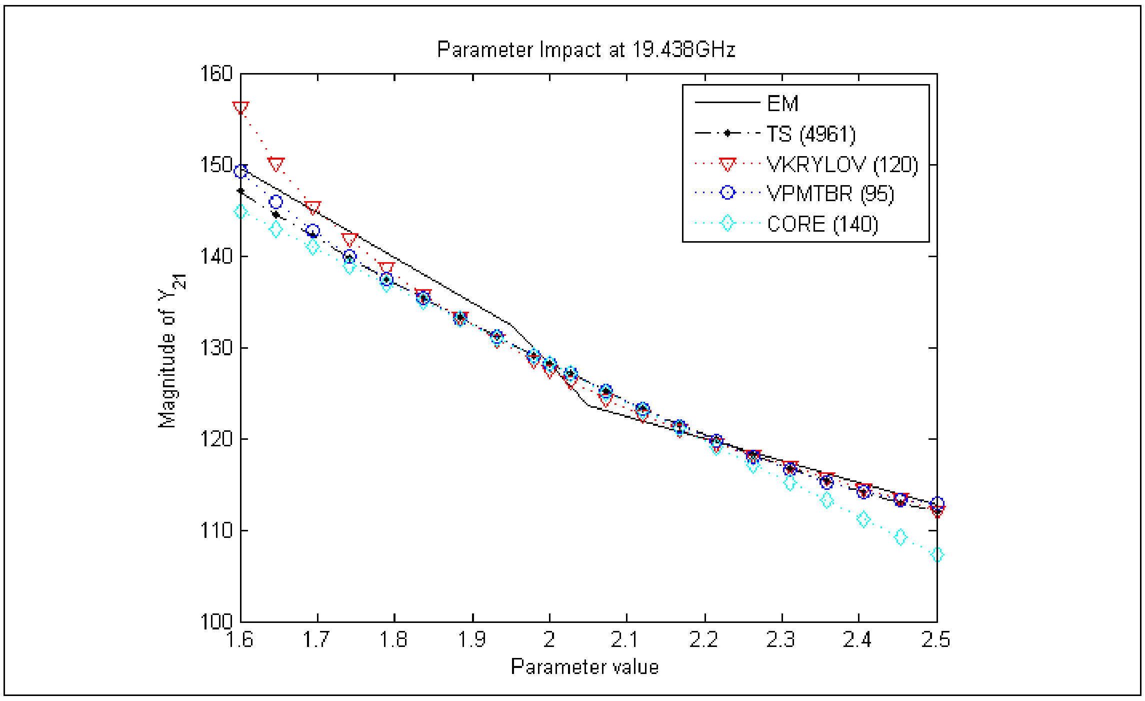


Figure 27. Spiral – Parameter Impact for some of the parametric Algorithms at a random frequency.

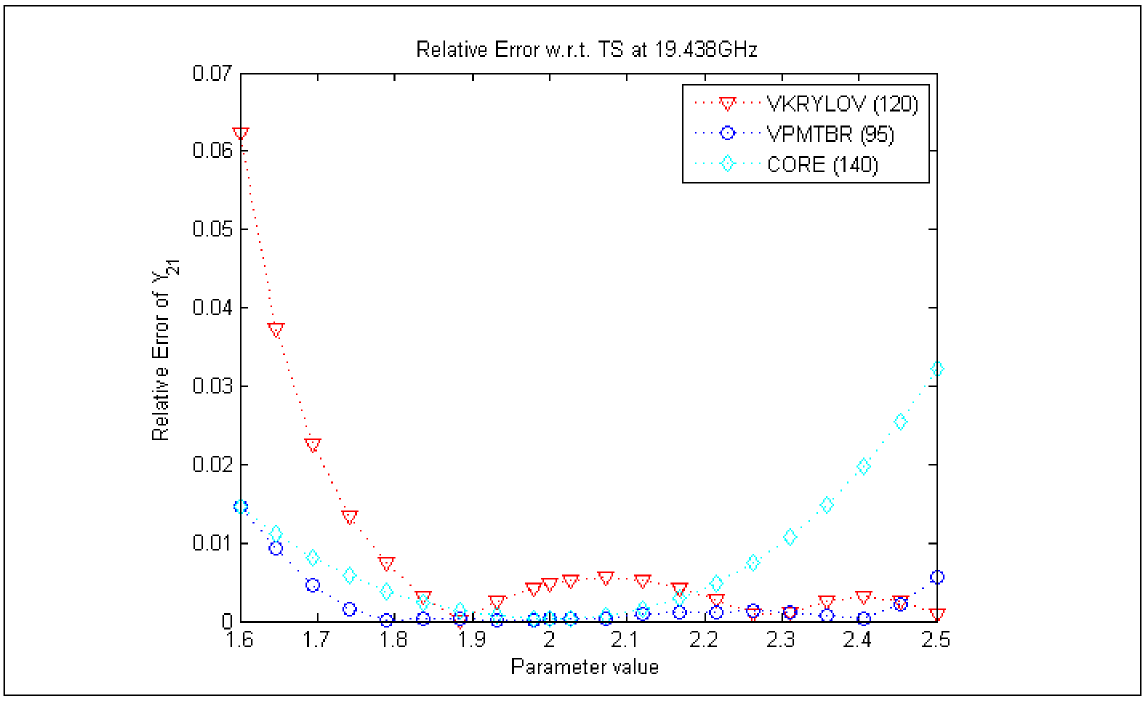


Figure 28. Spiral – Relative Error w.r.t. the Taylor Series versus the Parameter Variation for some of the parametric Algorithms at a random frequency.

It can be seen that in this case, the variation of the response with the variation of the parameter is much more linear than in the case of the LShape, and therefore first series Taylor Series approximation is enough in the parameter variation range.

9 Conclusions

This document summarizes the activities carried out in Task 3.1 of the CHAMELEON RF project. In it, an extensive study of the Parametric Model Order Reduction (pMOR) techniques was presented and simulation results were drawn to support our findings that pMOR techniques are quite useful in reducing the order of parameter-based systems.

As an outcome of this task, we have proposed a Taylor-series based representation for the systems to be handled in WP3 and, through numerous studies conducted by the partners, have provided evidence that indicates that first-order approximations are sufficient for most cases. However, the proposed representation and the framework developed to handle it, is easily extendable to accommodate higher order approximations. This proposal and these results are in agreement with data envisaged inside the project.

A framework for carrying out the mentioned tests was specified and developed in this task, including standards and functions for the representation, processing, reduction and comparison of the various procedures. All these data structures, procedures and functions were included in the Parametric ROM Workbench framework, a comprehensive extension of the previously available ROM Workbench. This parametric framework was also further extended in order to cover the reduction of interconnected models, using techniques consistent with the ones presented in D3.2.

10 References

[1]. A. Vandendorpe and P. Van Dooren, "Model Reduction of Interconnected Systems", in Proc. of 16th International Symposium on Mathematical Theory of Networks and Systems (MTNS 2004), Leuven, Belgium, pp. paper THP3-4, July, 2004.

[2]. B. Moore, "Principal Component Analysis in Linear Systems: Controllability, Observability and Model Reduction", *IEEE Transactions on Automatic Control*, pp. 17-32, Vol AC-26, No. 1, Feb. 1981.

[3]. J.R. Phillips, L. Daniel and L.M. Silveira, "Guaranteed Passive Balancing Transformations for Model Order Reduction", *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, pp. 1027-1041,Vol 22, No. 8, Aug. 2003.

[4]. J.R. Phillips and L.M. Silveira, "Poor man's TBR: A simple model reduction scheme*", In Proc. Design, Automation and Test in Europe Conference and Exhibition, DATE 2004*, pp. 938-943, Paris, France, Feb. 2004.

[5]. J.R. Phillips and L.M. Silveira, "Poor man's TBR: A simple model reduction scheme", *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, pp. 43-55,Vol 24, No. 1, Jan. 2005.

[6]. A. Odabasioglu, M. Celik and L.T. Pileggi, "PRIMA: passive reduced-order interconnect macromodeling algorithm", *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, pp. 645-654,Vol 17, No. 8, Aug. 1998.

[7]. P. Feldmann and R.W. Freund, "Efficient Linear Circuit Analysis by Padé Approximation via the Lanczos Process", *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, pp. 639-649,Vol 14, No. 5, May. 1995.

[8]. L.M. Silveira, M. Kamon, I. Elfadel and J. White, "A Coordinate-Transformed Arnoldi Algorithm for Generating Guaranteed Stable Reduced-Order Models of RLC Circuits", *In International Conference on Computer Aided-Design*, San Jose , CA., USA, pp. 288-294, Nov. 1996

[9]. I.M. Elfadel and D.L. Ling, "A Block Rational Arnoldi Algorithm for Multipoint Passive Model-Order Reduction of Multiport RLC Networks", *In International Conference on Computer Aided-Design*, San Jose, CA., USA, pp. 66-71, Nov. 1997.

[10]. R. Achar and M.S. Nakhla, "Simulation of High-Speed Interconnects", *Inv. Paper in Proc. of the IEEE*, pp. 693-728, Vol. 89, No. 5, May 2001.

[11]. H. Yu, L. He and S.X.D. Tan, "Block Structure Preserving Model Order Reduction", *In IEEE Behavioral Modeling and Simulation Wokshop* pp. 1-6, Sept. 2005.

[12]. R.W. Freund, "SPRIM: Structure-Preserving Reduced-Order Interconnect Macro-Modeling", *In IEEE Proc. International Conference on Computer Aided-Design, San Jose, CA., USA*, pp. 80-87 , Nov. 2004.

[13]. Y. Liu, L.T. Pileggi and A. Strojwas, “Model Order Reduction of RC(L) Interconnect Including Variational Analysis”, *Proc. of 36th ACM/IEEE Design Automation Conference,* pp 201-206 , June 1999.

[14]. P. Heydari and M. Pedram, “Model Reduction of Variable-Geometry Interconnects Using Variational Spectrally-Weighted Balanced Truncation”, *In IEEE Proc. International Conference on Computer Aided-Design, San Jose, CA., USA,* pp. 66-71, Nov. 2001.

[15]. P. Li, F. Liu, X. Li, L.T. Pileggi and S.R. Nassif, “Modeling Interconnect Variability Using Efficient Parametric Model Order Reduction”, *In Proc. Design, Automation and Test in Europe Conference and Exhibition, DATE 2005,* Feb. 2005.

[16]. L. Daniel, O.C. Siong, L. Chay, K.H. Lee and J. White, “A Multi-parameter Moment-Matching Model-Reduction Approach for Generating Geometrically Parameterized Interconnect Performance Models”, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* pp. 678-693,Vol 23, No. 5, May. 2004.

[17]. X. Li, P. Li and L.T. Pileggi, “Parameterized Interconnect Order Reduction with Explicit-and-Implicit Multi-Parameter Moment Matching for Inter/Intra-Die Variations”, *In IEEE Proc. International Conference on Computer Aided-Design, San Jose, CA., USA,* pp. 806-812, Nov 2004.

[18]. P.K. Gunupudi, R. Khazaka, M.S. Nakhla, T. Smy and D. Celo, “Passive Parameterized Time-Domain Macromodels for High-Speed Transmission-Line Networks”, *IEEE Trans. On Microwave Theory and Techniques,* pp. 2347-2354, Vol. 51, No. 12, Dec. 2003.

[19]. P. Gunupudi and M. Nakhla, “Multi-Dimensional Model Reduction of VLSI Interconnects”, *In IEEE Proc. Custom Integrated Circuits Conference,* pp. 499-502, May 2000.

[20]. J. Wang, P. Ghanta and S. Vrudhula, “Stochastic Analysis of Interconnect Performance in the Presence of Process Variations”, *In IEEE Proc. International Conference on Computer Aided-Design, San Jose, CA., USA,* pp. 880-886, Nov. 2004.

[21]. J.R. Phillips, “Variational Interconnect Analysis Via PMTBR”, *In IEEE Proc. International Conference on Computer Aided-Design, San Jose, CA., USA,* pp. 872-879, Nov. 2004.

[22]. S.R. Nassif, “Modeling and Analysis of Manufacturing Variations”, *In IEEE Proc. Custom Integrated Circuits Conference, San,* pp. 223-228, ---- 2001.

[23]. V. Mehrotra, S.L. Sam, D. Boning, A. Chandrakasan, R. Vallishayee and S. Nassif, “A Methodology for Modeling the Effects of Systematic Within-Die Interconnect and Device Variation on Circuit Performance”, *Proc. of 37th ACM/IEEE Design Automation Conference,* pp 172-175, June 2000.

[24]. K. Agarwal, M. Agarwal, D. Sylvester and D. Blaauw, “Statistical Interconnect Metrics for Physical Design Optimization”, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* pp. 1273-1288,Vol 25, No. 7, July. 2006.

[25]. T. Chen and A. Hajjar, “Statistical Timing Analysis of Coupled Interconnects Using Quadratic Delay-Change Characteristics”, *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, pp. 1677-1683,Vol 23, No. 12, Dec. 2004.

[26]. CHAMELEON RF, “WP3.IR.2, Main Functions Specific to pRWB”, *Internal Report Work Package 3*, CHAMELEON RF 2007.

[27]. CHAMELEON RF, “WP3.IR.1, Standard for Parametric Systems Representation, V.7”, *Internal Report Work Package 3*, CHAMELEON RF 2006.

[28]. D. Ioan, G. Ciuprina, W. Schilders, N.v.d. Meijs, K-J.v.d. Kolk, W. Schoenmaker, L.M. Silveira, J. Fernández , “D1.2 – Report on Compact Electromagnetic Modelling of RF Integrated Structures”, *Deliverable Work Package 1*, www.chameleon-rf.org, CHAMELEON RF 2006.

[29]. L.M. Silveira, J. Fernández , D. Ioan, G. Ciuprina, W. Schilders, N.v.d. Meijs, K-J.v.d. Kolk, “D3.2 - Order Reduction of Interconnected Sets of Compact Models”, *Deliverable Work Package 3*, www.chameleon-rf.org, CHAMELEON RF 2006.

Appendix A – Standards for Parametric Model Order Reduction Techniques

A.1 Introduction

The typical representation of a linear time-invariant (LTI) system depending on several parameters is:

$$\left(\mathbf{G}(l_1, l_2, ...l_P) + s\cdot\mathbf{C}(l_1, l_2, ...l_P)\right)\cdot\mathbf{x} = \mathbf{B}\cdot\mathbf{u}$$
$$\mathbf{y} = \mathbf{L}'\cdot\mathbf{x} + \mathbf{D}\cdot\mathbf{u}$$

where $\lambda = [l_1, l_2, ...l_P]$ is the set of parameters to be considered and P is the total number of parameters.

The parameters affecting the behavior of the system are usually the result of process variations on several physical values, such as metal width, length, substrate height, etc…

We can assume that these parameters affect the system matrices **G** and **C**, and do not have a direct effect on the input and output related matrices, **B**, **L** and **D.** (The validity of this statement, which we assume true for now, is dependant on how the equations were constructed).

Generically, the techniques in Parametric Model Order Reduction (pMOR) are usually based on two approaches, which require different representations on the parametric formulation:

- Expansion in Taylor Series of the system matrices **G** and **C**, where these matrices are represented as a linear combination of the powers of the parameters multiplying matrix coefficients. These coefficients are obtained by derivation on the matrices w.r.t. the parameters. The expansion can be done up to any order, depending on the degree of accuracy required:

$$\mathbf{G}(l_1, l_2, ...l_P) = \sum_{i1=0}^{\infty}\sum_{i2=0}^{\infty}\cdots\sum_{iP=0}^{\infty}\mathbf{G}_{i1,...,iP}\cdot\Delta l_1^{i1}\cdots\cdot\Delta l_P^{iP}$$
$$\approx \sum_{i1=0}^{l1}\sum_{i2=0}^{l2}\cdots\sum_{iP=0}^{lP}\mathbf{G}_{i1,...,iP}\cdot\Delta l_1^{i1}\cdots\cdot\Delta l_P^{iP}$$

  Some terms in this approximate mathematical expression can be neglected and the expansion order w.r.t. each parameter (*lk*) may be different.

Notes:

- $\mathbf{G}_{i1...ik}$ are the coefficients of the Taylor series. They are sensitivities, i.e. derivatives of the matrix **G** w.r.t. parameters, divided by some factorial number.
- A first order expansion corresponds to $l_1 = l_2 = ...l_P = 1$ and no cross terms.
- An alternative way of writing the Taylor expansion is given by:

$$\mathbf{G}(l_1, l_2, ..l_P) = \mathbf{G}_{nom} + \sum_{k=1}^{\infty}\left(\sum_{i_1=1}^{P}\sum_{i_2=1}^{P}.....\sum_{i_k=1}^{P}\mathbf{G}_{i_1,...,i_k}\cdot\Delta l_1^{i_1}\cdots\cdot\Delta l_P^{i_k}\right)$$

$$\approx \mathbf{G}_{nom} + \sum_{k=1}^{lG}\left(\sum_{i_1=1}^{P}\sum_{i_2=1}^{P}.....\sum_{i_k=1}^{P}\mathbf{G}_{i_1,...,i_k}\cdot\Delta l_1^{i_1}\cdots\cdot\Delta l_P^{i_k}\right)$$

where $lG = 1$ corresponds to first order expansion for **G**.

- Sampling of the values of the system matrices **G** and **C** when the parameters take a fixed value in their range of variation. The matrices obtained are typically no longer dependant on the parameters.

e.g.: $\left\{\mathbf{G}^{(1)}\left(l_1^{(1)}, l_2^{(1)}, ..l_P^{(1)}\right), ..\mathbf{G}^{(m)}\left(l_1^{(m)}, l_2^{(m)}, ..l_P^{(m)}\right)\right\}$

where m is the number of samples

The pMOR techniques use these representations in a different way, for instance for building up a projection matrix, which is applied in a congruent transformation on the system matrices following the traditional MOR procedure.

If the system is a parametric one, the congruent transformation is applied not only over the nominal matrices, but on the expansion terms as well, resulting in a parametric ROM. This is true for most of the techniques studied so far, with few exceptions.

A standard for storing and representing these parametric systems is now needed. Since it has been decided in CHAMELEON RF to use the ROM Workbench as the framework for algorithmic development, such a representation must be defined and added to the workbench.

The following describes an initial proposal to achieve this goal.

A.2 Motivations

The main motivation was to create a single system item able to handle both representations, and the rest of the information needed in a parametric ROM Workbench.

This parametric ROM Workbench might be an extension to the previous one, where the new techniques should be added. The need of handle different (or extended) information may require some modifications of former procedures.

This item should not be restrictive in relation to the capacity to handle different kinds of data, and the amount of potential data needed for obtaining good accuracy. It should be easy to describe and handle, as well as be easily expanded.

For these reasons, the system item is a cell array. This allows a hierarchical structure, and the use of different kind of data under a single element. The access to a required field is sequential, and the data can be divided and grouped into subfields.

In the ROM Workbench already exists a cell array called ***allsys*** that holds information and data for all the models in the ***database***. Each entry is a structure representing a system, with its respective fields of information. However, these structures do not have into account the parametric information needed for the aim of CHAMELEON RF, so we need to define a new extended structure able to be handle by the future parametric ROM Workbench procedures.

The system item is the same for a ROM or an original (non-reduced) system. The fields are the same for all the systems, and only the data stored is modified.

In some cases, there are fields which are not needed, and they should not be accessed. These fields can be not filled, but at least they should exist. This can be sometimes a loss of used memory, but will help to translate procedures to other languages not as permissive as MATLAB.

A.3 Description

The system will be a structure with several fields. This model, when read into the ROM Workbench data base, will be found in the ***allsys*** variable, for example, as the 5-*th* model:

allsys{5}.type = ‘var’

allsys{5}.prom = Sys;

General Fields

These fields contain general information on the system

- Sys.type = ‘var’ – fixed string
- Sys.ver = 7 – fixed number, version

- **Sys.id_original** – string. It shows the identifier of the system from whom the actual has been obtained (if it has been reduced). This could be useful in multi-step reductions and interconnected systems. It should not be modified during the processing.

- **Sys.id** – string to be used as a label. Identifier is a name for the system. It can be modified during the processing.

- **Sys.lti_rpnt** – representation which can be ***‘CGBLD’*** in this version. Next version may allow also the ***‘ABCD’*** representation

  lti_rpnt is a flag that indicates the system representation, *EABCD* (‘***ABCD’***), with E the identity matrix not dependant on the parameters (in this case we will assume that only A and B depend on the parameters) or a *CGBLD* representation, with both C and G depending on the parameters.

Due to the fact that most of the parametric pMOR techniques studied so far do not consider the *ABCD* representation, with A and B depending on the parameters, it will be considered by now that the parametric procedures will work only with *CGBLF* representations.

- **Sys.system_type** – can be *'TS'* or *'SPL'*. *System_type* indicates the kind of parametric MOR has been used. It can be Taylor Series or Sampling. If another representation is needed, it is easy to expand this.

- **Sys.reduced** – can be 0 (non-reduced) or 1 (reduced). Reduced indicates if the system has been already reduced or not, and if so, the field that contains the projection matrix will be non empty.

- **Sys.no_param** – integer, number of parameters, *no_param* indicates the number of parameters that the representation takes into account. Note that at this point no physical behavior is relevant, and so the parameters only have mathematic meaning. It is not important if it is an intra-die or inter-die, or if it depends on the fabrication process or on the environment. It will have a specific mathematical effect that is determined by a matrix. This may require changes in the future. Any change will allow backward compatibility.

- **Sys.ref_param** – cell array of length equal to the number of parameters, holding the reference value of each parameter. The nominal matrices correspond to these set of parameters.

- **Sys.label_param** – cell array of length equal to the number of parameters, holding labels for each parameter. This will help to remember the physical significance of the parameters.

- **Sys.no_term** – integer, number of inputs; it will be assumed that the number of inputs is equal to the number of outputs, and the grounded terminal is not counted.

- **Sys.term_info** – cell array of length equal to the number of terminals (which is the same as the number of inputs or outputs), telling how the inputs are excited. Each entry of this vector can be '*ec*' (for electric current excitation) or '*ev*' (for electric voltage excitation).

Taylor Series Information Fields (Sys.Taylor_info)

This field indicates information on the Taylor Series representation

Sys.Taylor_info = *tsi* **-** is a structure with the following fields

- **tsi.var_param** – cell array of length equal to the number of parameters, holding the limits of the variation for each parameter. A component of the cell array can be a number or a vector with two elements. If it is a number, it represents the maximum percentage variation of the parameter. If it is a vector, than the components of the vector represents the minimum and maximum possible values of the parameters.

- **tsi.sens_info** – is also a structure with the following fields
  - **no_C_terms** – integer, holding the number of terms in the Taylor expansion of C (*lC* in the formulas above).
  - **C** – cell array of length equal to *no_C_terms*, each component is a vector of integers, of size equal to no of parameters + 1. The first position indicates the order of the derivative and the next order components indicate the parameters w.r.t which the sensitivity is computed (see example below).
  - **no_G_terms** – similar to *no_C_terms.*
  - **G** – similar to *C* field.

To be more explicit, let us take an example. Assuming that we have 3 parameters and the only sensitivities that count are

$$\frac{\partial C}{\partial l_1}, \quad \frac{\partial^2 C}{\partial {l_3}^2}, \quad \frac{\partial^2 C}{\partial l_1 \partial l_3}, \quad \frac{\partial^3 C}{\partial l_1 \partial l_2 \partial l_3}$$

In this case the length of the cell array *tsi.sens_info.C* is 4 and its components are:

```
tsi.sens_info.C{1} = [1 1 0 0]      % order is 1,
                                    % first order derivative w.r.t
                                    % the first parameter
tsi.sens_info.C {2} = [2 3 3 0]    % order 2,
                                   % second order derivative
                                    % w.r.t to
                                    % third parameter, two times
tsi.sens_info.C {3} = [2 1 3 0]
tsi.sens_info.C {4} = [3 1 2 3]
```

Comments:

It is supposed that, independently on the technique to be applied for reduction, this field will always be used, because the sampling is likely to be done over this expansion series.

The matrices G and C depend on the parameters following some complicated function for each parameter. We can imagine a *“functional interface”* that is able to compute new G’s and C’s for every parameter setting. Such a functional interface may call for instance FIT or, another possibility is the computation of semi-state matrices by generating an intermediate equivalent circuit. This functional interface should be done in WP2. WP3 starts with the data (i.e. state spate matrices and their sensitivities).

The most probable input for the RWB will be the terms of the Taylor expansion series, i.e. nominal values of the matrices and the sensitivities of the state space matrices C and G w.r.t. the parameters.

In order to apply some parametric MOR techniques that use samples for building the projector to be applied in a congruent transformation, we need to sample the parameter space. We use the Taylor Series expansion for computing the perturbed matrices with the formula (illustrated for first order sensitivities):

$$Cpert = C + \sum_{k=1}^{P} \Delta l_k \frac{\partial C}{\partial l_k}$$

So, for having a matrix sampling, we apply the probabilistic information to constrain the parameter sampling, and compute the value Δl_k (if no probabilistic information is available, only the limits can be defined in a random sampling).

We obtain the value *Cpert* (and *Gpert*), and these are going to be the sampled matrices used by the algorithm.

So, the Taylor Series information and the expansion terms must always exist whatever the technique (sampling, moment matching, etc…) to apply.

The random sampling information (PDF, means and deviations) should always exist, whatever the technique, because it is useful not only on the sampling techniques, but as well as for giving results that are more related to the real behavior of the problem.

The flag ‘*TS’* or ‘*SMP*’ make sense after reduction. If a reduction has been made through sampling, *'SMP'* should indicate that the field with the information of the samples used in the reduction is filled. If not the field with the samples have no sense (in fact it should be empty).

Probabilistic Information (Sys.rand_spl_info)

Sys.rand_spl_info = *rsi* **-** is a structure with the following fields

- **fdp_type** – cell array equal to the number of parameters, each component being an integer code such as 'norm' -0- 'exp' -1- ...
- **mean** - cell array equal to the number of parameters, each component being the mean of the parameter.
- **deviation** - cell array equal to the number of parameters, each component being the deviation of the parameter.

This field stores lower level information about the variation of the parameters.

This field is only expected to be relevant in sampling-based techniques, but as well when obtaining the results, in order to show information closer to reality.

It has probabilistic information about each parameter, which is going to help to constraint the sampling choice in the parameters sampling space.

It is supposed that the sampling is going to be made over the Taylor Series expansion, where the perturbations (Δl_k) will be fixed to a value (with the help of this probabilistic information), and applied so a fixed value for the system matrices *Cpert* and *Gpert* is obtained.

In this probabilistic information we find, for each parameter, the kind of Probability Distribution Function (PDF) which will be between a predetermined group, the mean and the standard deviation for this PDF.

Sampling Data information (Sys.spl_data)

Sys.spl_data = *SPL_D* - structure with the following fields

- **SPL_D.no_spl** - integer, number of samples.
- **SPL_D.spl_points** - a cell array of length equal to the number of samples, each component is a vector of length equal to the number of parameters, holding the parameters for a sample.
- **SPL_D.no_freq** - cell array of length equal to the number of samples, each component is a vector holding the no of frequencies used for that sample. This information is not directly read for the input xml, nor written directly in xml when saving the model.

- **SPL_D.frequencies** - cell array of length equal to the number of samples, each component is a vector holding the frequencies used for that sample. This information is not directly read for the input xml, nor written directly in xml when saving the model.

- **SPL_D.snp_data** - cell array of length equal to the number of samples, each entry contains complex numbers (Z or Y, according to the significance of the *terminalinfo*) – it holds the frequency response of each sample, computed in the frequency data given by *Spl_points*. This information is not directly read for the input xml, nor written directly in xml when saving the model.

- **SPL_D.snp_filename** - cell array of length equal to the number of samples, each entry contains the *snp* filename in which the frequency answer is stored.

- **SPL_D.AFSflag** - cell array of length equal to the number of samples, each entry contains a flag telling if *AFS* is used (1) or not (0) for a specific sample.

- **SPL_D.AFSmax** - cell array of length equal to the number of samples, maximum number of points for *AFS* procedure.

- **SPL_D.AFSerr** - cell array of length equal to the number of samples, error for *AFS* procedure.

This field stores information of the values of the parameters and frequency used in order to obtain each sampled matrix.

The information here stored has nothing about probabilistic information of the parameters, but have information of the several samples taken.

The first information is the number of samples stored.

The second is a full matrix, which contains on each row the values for the different parameters, which correspond to the sampled matrix. So, the *'i'* row has the following entries:

$$\left[\quad value_i\_parameter_1 \quad \ldots \quad value_i\_parameter_P \right]$$

Note that frequency is **not** considered as a parameter here. For a given parameter set, a different set of frequencies might be used.

Also, to allow an easier implementation of Adaptive Frequency Sampling procedures, we prefer to use a separate matrix with frequencies. Each row of this matrix corresponds to a sample of the parameter space. The first element in the row indicates the no of frequencies used for that sampling (this no is less than *nof*). When increasing the accuracy of the *AFS* procedure, the number of frequencies used for a sample may increase, thus filling the raw with values.

The matrices G and C do not depend on the frequency, but in the sampling techniques, the projection matrix is built from samples not only in the parameter space, but as well in the frequency space.

This frequency space is usually much larger than the parameter space, so it is better to handle it as a different item. Some techniques (*A Multi-Parameter Moment-Matching Model-Reduction Approach for Generating Geometrically Parameterized Interconnect Performance Models, L. Daniel et al.*) treat the frequency as a parameter, but this turns up to be inefficient. It seems to be better to treat frequency as a "special" parameter.

However, this sampling information is stored after a sampling technique is applied, and it reflects the information used for building the projection matrix. The frequency is an essential part of this. We think it is important to store this information, because in can be useful for future refinement of the algorithms (for example, we may want to introduce extra samples for improving the accuracy).

Nominal matrices (Sys.nominal_matrices)

Sys.nominal_matrices = *mtx*- is a structure with the following fields

- **mtx.filename** - filename from which the nominal matrices are read.
- **mtx.C** - C NOMINAL matrix. This information is not directly read for the input xml, nor written directly in xml when saving the model.
- **mtx.G** - G NOMINAL matrix. This information is not directly read for the input xml, nor written directly in xml when saving the model.
- **mtx.B** - B matrix, not dependant on the parameters. This information is not directly read for the input xml, nor written directly in xml when saving the model.
- **mtx.L** - L matrix, not dependant on the parameters. This information is not directly read for the input xml, nor written directly in xml when saving the model.
- **mtx.D** - D matrix, not dependant on the parameters. This information is not directly read for the input xml, nor written directly in xml when saving the model.

The first two entries are the value of matrix C and G when all the variations (Δl_k) have zero value. The rest are the values of the matrices that do not depend on the parameters.

Nominal frequency response (Sys.nominal_freq_response)

Sys.nominal_freq_resp = *NFR* – is a structure with the following fields

- **NFR.snp_filename** – name of the *snp* file in which the frequency response of the nominal system is stored.

- **NFR.AFSflag** – flag, 1 if *AFS* is used and 0 otherwise.

- **NFR.AFSmax** – max no of freq points computed in the *AFS* procedure.

- **NFR.AFSerr** – error for the *AFS* procedure.

Sampled Matrices (Sys.spl_matrices)

Sys.spl_matrices = *SPL_M* is a structure with the following fields

- **SPL_M.filenames** - cell array of length equal to the no of samples, each component holding the file from which the matrices of a given sample are read.

- **SPL_M.C** – *Cspl*, cell array of length equal to the no of samples, each component holds a C matrix for a given sample. This information is not directly read for the input xml, nor written directly in xml when saving the model.

- **SPL_M.G** - *Gspl*, cell array of length equal to the no of samples, each component holds a G matrix for a given sample. This information is not directly read for the input xml, nor written directly in xml when saving the model.

This has the sampled matrices, where the ***i-th*** stored matrix was obtained by applying the above mentioned '*i*' row parameter values of the field ***sys.spl_data.spl_points*** on the Taylor Series.

Taylor Series expansion terms for C (Sys.sensC)

Sys.sensC = *SC* – structure with the following fields

- **SC.filenames** - cell array of length equal to the no of sensitivities of C terms, each component holding a filename from which the sensitivity is read.

- **SC.matrices** - cell array of sensitivities of C This information is not directly read for the input xml, nor write directly in xml when saving the model.

Taylor Series expansion terms for G (Sys.sensG)

Sys.sensG = *SG* – structure with the followingfields

- **SG.filenames** - cell array of length equal to the no of sensitivities of G terms, each component holding a filename from which the sensitivity is read.

- **SG.matrices** - cell array of sensitivities of G This information is not directly read for the input xml, nor write directly in xml when saving the model.

This field contains the Taylor Expansion terms (i.e. sensitivities). These terms are matrices, and they are expected to be very sparse, even sparser than the nominal C and G. They are stored in a linear way, and the meaning of each term corresponds to *TSI.sensitivity_info_C* or *G*. For the example given above, *lC = 4*, *SC.matrices{1}* is a sparse matrix holding the values of the first order sensitivities of C w.r.t the first parameter, *SC.matrices{2}* is a sparse matrix holding the values of the second order derivative w.r.t the third parameter, *SC.matrices{3}* is a sparse matrix holding the second order derivatives w.r.t the first and third parameters, and *SC.matrices{4}* is a sparse matrix holding a third order derivative w.r.t the first, second and third parameters.

Projection matrix (Sys.prj_matrix)

Sys.prj_matrix = *PM* - structure with the following fields

- **Filename** – Name of the file where the matrix is stored.

- **matrix** - This information is not directly read for the input xml, nor write directly in xml when saving the model.

Projection matrix built up with MOR or PMOR techniques

This is the projection matrix that is going to be applied in a congruent transformation, over the nominal matrices of the original system, and over each Taylor series expansion term (i.e. sensitivity) in order to obtain a **Parametric Reduced Order Model**.

Once the reduction is performed, the new reduced system is going to be a parametric one, so its representation is going to be another TS representation (whatever the technique applied).

Using ***Cpert*** and ***Gpert*** – the projection matrix **X** is computed, and then the reduced matrices **C**r, **G**r, etc. The TS representation of the reduced system should contain the sensitivities of these reduced matrices.

The projection is applied both in the nominal matrix and in the expansion terms (sensitivities), for obtaining a parametric ROM.

A.4 Interconnected Systems

An interconnected system will be a structure with several fields. When read into the ROM Workbench data base, we will find it as (for example)

```
allsys{6}.type = 'mvar';
allsys{6}.mprom = msys;
```

msys is a structure with the following fields:

- **msys.mvar** - string, fixed to '*mvar'*.
- **msys.ver** - value, fixed to 7.0.
- **msys.id** - string, label for the system.
- **msys.no_sys** - integer, number of interconnected systems.
- **msys.components** - MSYSC a structure with the following fields
  - **MSYSC.filename** - cell array, of length equal to the number of interconnected systems, holding the filenames from which these systems will be read (the files are xml files with structures of *var* type, as defined in the previous section.
  - **MSYS.sys** - structure of *var* type. This field will be filled with data after the reading of the xml file. That is why there is no corresponding tag in the xml file describing interconnected systems.
- **msys.mtx_intcnt** = *MTX* – structure with the following fields
  - **MTX.filename** - string, the filename from which the interconnection matrices K, H and F will be read.
  - **MTX.K** – interconnection among systems. This field will be filled with data after the reading of the file with matrices. That is why there is no corresponding tag in the xml file describing interconnected systems.
  - **MTX.H** – linear combination of inputs. This field will be filled with data after the reading of the file with matrices. That is why there is no corresponding tag in the xml file describing interconnected systems.
  - **MTX.F** –linear combination of outputs. This field will be filled with data after the reading of the file with matrices. That is why there is no corresponding tag in the xml file describing interconnected systems.

This is another item, built as a cell array, for handling several systems and their interconnections.

It is based on the paper: *A. Vandendorpe, P. Van Dooren, "Model Reduction of Interconnected Systems".*

The first field is an identifier for the global system.

The second indicates the number of subsystems that are going to be part of the global one.

The third field contains the systems. Each subentry is a system, and for this reason, each subentry will correspond to a *"system item"* shown in the previous section, with all its information about the system and its parametric information.

The forth field is a group of subentries, where each one is filled with a matrix. These matrices define the interconnections between the subsystems (K), and their connection to the inputs (H) and the outputs (F).

A.5 Some General Considerations

The matrices **G** and **C** depend on the parameters following some complicated function for each parameter. For a given parameter set, i.e. for a given configuration, there is a procedure to compute **G** and **C** (such as *FIT* that allows the computation of state space matrices, or another possibility is the computation that uses the equivalent circuit).

In order to apply variational procedures, you need to sample the parameter space. Rather than compute *exactly* the matrices for different parameters (i.e. apply FIT several times), we will be able to use Taylor Series expansion for doing this (apply FIT to compute nominal values of state space matrices and whatever technique to find their sensitivities) and then compute perturbed matrices with the formula (illustrated for first order sensitivities)

$$\mathbf{C}pert = \mathbf{C} + \sum_{k=1}^{P} \Delta l_k \frac{\partial \mathbf{C}}{\partial l_k}$$

Some extra information can be given by the way these physical parameters vary (what we have named probabilistic information). This probabilistic information is used in the computation of $\Delta\lambda_k$ above. It depends on the information available that we may use another mathematical representation in the future.

So, what we have now is a nominal value for the matrices **G** and **C**, which is the value when the parameters take their nominal physical value (or reference value). We also have the expansion terms (also called sensitivities) which are going to multiply the parameter variation (difference between reference parameter value and actual value after variation, denoted by $\Delta\lambda$). And we also have some probabilistic information on how this difference varies (or how the parameter value varies). In shot, the probabilistic information affects the computation of $\Delta\lambda$.

It is important to notice that this was the mathematical world, where the parameters were related to (and were meant to model) physical effects. It was a model of the reality. Now we change to the computational world within ROM Workbench.

What a parametric ROM Workbench is going to receive is the nominal matrices, the expansion terms (sensitivities), that may be zero matrices or not and some probabilistic information, which will be used for the computation on $\Delta\lambda$.

Now the computational parameters vary between a minimum and a maximum, following some probabilistic information (the same that was given before), and they rule the values of the matrices C and G in the following way:

$$\mathbf{G}(l_1, l_2, ... l_P) = \mathbf{G}_0 + l_1 \cdot \mathbf{G}_1 + ... + l_P \cdot \mathbf{G}_P + l_1^2 \cdot \mathbf{G}_{11} + ... + l_P^2 \cdot \mathbf{G}_2 + ...$$

and the same for **C** (above, in the *rhs*, $\mathbf{G}_k$ are sensitivities and λ_k are variations).

In fact, λ_1 and λ_1^2 are different parameters in most of the parametric MOR techniques, and we keep this notation only for the fact that the variation of the second must be the square of the variation of the first. Some techniques may have them as the same parameter and may take advantage of this, but it is not usual.

This is the parametric ROM Workbench model (this may be not the only one in a future).

Now, this is the information we have and the only one we can use in ROM Workbench:

- Nominal matrices.
- Expansion terms (sensitivities).
- Probabilistic information of the variation of the mathematical parameters (or probabilistic information of the variation of the physical parameter).
- A maximum variation range.

The computational parameters (more precisely the variations $\Delta\lambda$) are now only numbers, this is, weights related to the contribution of the corresponding expansion term (sensitivity) to the perturbation over the nominal matrix.

Now the nominal matrix $\mathbf{G}_0$ or $\mathbf{C}_0$ is obtained when the computational parameters are zero (the difference between the physical actual value and the physical nominal value).

And the perturbed matrix is obtained by applying the variation on the parameters and obtaining the result of the linear combination.

Sometimes, we will call *"computational parameters"* the difference between the physical nominal value and the physical actual value, i.e. the perturbation from nominal ($\Delta l = l_{pert} - l_{nom}$).

Another option is that the probabilistic information was about the variation of the value of the parameter, and not about the difference between the nominal and the actual. In this case, the computational parameter continues to be the difference, but when the probabilistic information is applied, it must be done in a different way. The only change is that an extra subtraction must be done.

So, as a conclusion, ROM Workbench should be a software tool for the treatment (Parametric Model Order Reduction, Representation of Parametric Results and Comparatives of PMOR techniques at least) of the data received. All the information related to physical or mathematical representations is supposed to be encoded previously and translated to the kind of computational data handle by ROM Workbench.

Appendix B – XML Standard for Parametric Systems Representation

B.1 XSD File for Parametric Systems

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2006 rel. 3 sp2 (http://www.altova.com) by
Gabriela Ciuprina (Politehnica University of Bucharest) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
      <xs:element name="root">
            <xs:annotation>
                  <xs:documentation>Parametric system - in accordance
with version 7.0 of the specification</xs:documentation>
            </xs:annotation>
            <xs:complexType>
                  <xs:sequence>
                        <xs:element name="type" fixed="var">
                              <xs:complexType>
                                    <xs:simpleContent>
                                          <xs:extension
base="xs:string">
                                                <xs:attribute
name="idx" use="required" fixed="1"/>
                                                <xs:attribute
name="type" use="required" fixed="char"/>
                                                <xs:attribute
name="size" use="required" fixed="1 3"/>
                                          </xs:extension>
                                    </xs:simpleContent>
                              </xs:complexType>
                              <!--Type fixed to 'var'-->
                        </xs:element>
                        <xs:element name="ver" fixed="7.0">
                              <xs:complexType>
                                    <xs:simpleContent>
                                          <xs:extension
base="xs:double">
                                                <xs:attribute
name="idx" use="required" fixed="1"/>
                                                <xs:attribute
name="type" use="required" fixed="double"/>
                                                <xs:attribute
name="size" use="required" fixed="1 1"/>
                                          </xs:extension>
                                    </xs:simpleContent>
                              </xs:complexType>
                        </xs:element>
                        <xs:element name="id_original">
                              <xs:complexType>
                                    <xs:simpleContent>
                                          <xs:extension
base="xs:string">
                                                <xs:attribute
name="idx" use="required" fixed="1"/>
                                                <xs:attribute
name="type" use="required" fixed="char"/>
```

```
                                                <xs:attribute
name="size" use="required"/>
                                          </xs:extension>
                                    </xs:simpleContent>
                              </xs:complexType>
                        </xs:element>
                        <xs:element name="id">
                              <xs:complexType>
                                    <xs:simpleContent>
                                          <xs:extension
base="xs:string">
                                                <xs:attribute
name="idx" use="required" fixed="1"/>
                                                <xs:attribute
name="type" use="required" fixed="char"/>
                                                <xs:attribute
name="size" use="required"/>
                                          </xs:extension>
                                    </xs:simpleContent>
                              </xs:complexType>
                        </xs:element>
                        <xs:element name="lti_rpnt" fixed="CGBLD">
                              <xs:complexType>
                                    <xs:simpleContent>
                                          <xs:extension
base="xs:string">
                                                <xs:attribute
name="idx" use="required" fixed="1"/>
                                                <xs:attribute
name="type" use="required" fixed="char"/>
                                                <xs:attribute
name="size" use="required" fixed="1 5"/>
                                          </xs:extension>
                                    </xs:simpleContent>
                              </xs:complexType>
                        </xs:element>
                        <xs:element name="system_type">
                              <xs:complexType>
                                    <xs:simpleContent>
                                          <xs:extension
base="xs:string">
                                                <xs:attribute
name="idx" use="required" fixed="1"/>
                                                <xs:attribute
name="type" use="required" fixed="char"/>
                                                <xs:attribute
name="size" use="required"/>
                                          </xs:extension>
                                    </xs:simpleContent>
                              </xs:complexType>
                        </xs:element>
                        <xs:element name="reduced">
                              <xs:complexType>
                                    <xs:simpleContent>
                                          <xs:extension
base="xs:boolean">
                                                <xs:attribute
name="idx" use="required" fixed="1"/>
                                                <xs:attribute
name="type" use="required" fixed="boolean"/>
                                                <xs:attribute
name="size" use="required" fixed="1 1"/>
                                          </xs:extension>
                                    </xs:simpleContent>
                              </xs:complexType>
                        </xs:element>
                        <xs:element name="no_param">
                              <xs:complexType>
                                    <xs:simpleContent>
                                          <xs:extension
base="xs:integer">
                                                <xs:attribute
name="idx" use="required" fixed="1"/>
                                                <xs:attribute
name="type" use="required" fixed="double"/>
                                                <xs:attribute
name="size" use="required" fixed="1 1"/>
                                          </xs:extension>
                                    </xs:simpleContent>
                              </xs:complexType>
                        </xs:element>
                        <xs:element name="ref_param">
                              <xs:complexType>
                                    <xs:sequence>
                                          <xs:element name="item"
maxOccurs="unbounded">
                                                <xs:complexType>

      <xs:simpleContent>

      <xs:extension base="xs:double">

      <xs:attribute name="idx" use="required"/>

      <xs:attribute name="type" use="required" fixed="double"/>

      <xs:attribute name="size" use="required" fixed="1 1"/>

      </xs:extension>

      </xs:simpleContent>
                                                </xs:complexType>
                                          </xs:element>
                                    </xs:sequence>
                                    <xs:attribute name="idx"
use="required" fixed="1"/>
                                    <xs:attribute name="type"
use="required" fixed="cell"/>
                                    <xs:attribute name="size"
use="required"/>
                              </xs:complexType>
                        </xs:element>
                        <xs:element name="label_param">
                              <xs:complexType>
                                    <xs:sequence>
                                          <xs:element name="item"
maxOccurs="unbounded">
                                                <xs:complexType>

      <xs:simpleContent>

      <xs:extension base="xs:string">

      <xs:attribute name="idx" use="required"/>

      <xs:attribute name="type" use="required" fixed="char"/>

      <xs:attribute name="size" use="required"/>
      </xs:extension>

      </xs:simpleContent>
                                                </xs:complexType>
                                          </xs:element>
                                    </xs:sequence>
                                    <xs:attribute name="idx"
use="required" fixed="1"/>
                                    <xs:attribute name="type"
use="required" fixed="cell"/>
                                    <xs:attribute name="size"
use="required"/>
                              </xs:complexType>
                        </xs:element>
                        <xs:element name="no_term">
                              <xs:complexType>
                                    <xs:simpleContent>
                                          <xs:extension
base="xs:integer">
                                                <xs:attribute
name="idx" use="required" fixed="1"/>
                                                <xs:attribute
name="type" use="required" fixed="double"/>
                                                <xs:attribute
name="size" use="required" fixed="1 1"/>
                                          </xs:extension>
                                    </xs:simpleContent>
                              </xs:complexType>
                        </xs:element>
                        <xs:element name="term_info">
                              <xs:complexType>
                                    <xs:sequence>
                                          <xs:element name="item"
maxOccurs="unbounded">
                                                <xs:complexType>

      <xs:simpleContent>

      <xs:extension base="xs:string">

      <xs:attribute name="idx" use="required"/>

      <xs:attribute name="type" use="required" fixed="char"/>

      <xs:attribute name="size" use="required" fixed="1 2"/>

      </xs:extension>

      </xs:simpleContent>
                                                </xs:complexType>
                                          </xs:element>
                                    </xs:sequence>
                                    <xs:attribute name="idx"
use="required" fixed="1"/>
                                    <xs:attribute name="type"
use="required" fixed="cell"/>
                                    <xs:attribute name="size"
use="required"/>
                              </xs:complexType>
                        </xs:element>
                        <xs:element name="tsi">
                              <xs:complexType>
                                    <xs:sequence>
```

```
                                          <xs:element
name="var_param">
                                                <xs:complexType>
                                                      <xs:sequence>

      <xs:element name="item" maxOccurs="unbounded">

      <xs:complexType>

      <xs:simpleContent>

            <xs:extension base="xs:string">

                  <xs:attribute name="idx" use="required"/>

                  <xs:attribute name="type" use="required"
fixed="double"/>

                  <xs:attribute name="size" use="required"/>

            </xs:extension>

      </xs:simpleContent>

      </xs:complexType>

      </xs:element>
                                                      </xs:sequence>
                                                      <xs:attribute
name="idx" use="required" fixed="1"/>
                                                      <xs:attribute
name="type" use="required" fixed="cell"/>
                                                      <xs:attribute
name="size" use="required"/>
                                                </xs:complexType>
                                          </xs:element>
                                          <xs:element
name="sens_info">
                                                <xs:complexType>
                                                      <xs:sequence>

      <xs:element name="no_C_terms">

      <xs:complexType>

      <xs:simpleContent>

            <xs:extension base="xs:integer">

                  <xs:attribute name="idx" use="required" fixed="1"/>

                  <xs:attribute name="type" use="required"
fixed="double"/>

                  <xs:attribute name="size" use="required" fixed="1
1"/>

            </xs:extension>

      </xs:simpleContent>

      </xs:complexType>

      </xs:element>
      <xs:element name="C">
      <xs:complexType>
      <xs:sequence>
            <xs:element name="item" maxOccurs="unbounded">
                  <xs:complexType>
                        <xs:simpleContent>
                              <xs:extension base="xs:string">
                                    <xs:attribute name="idx"
use="required"/>
                                    <xs:attribute name="type"
use="required" fixed="double"/>
                                    <xs:attribute name="size"
use="required"/>
                              </xs:extension>
                        </xs:simpleContent>
                  </xs:complexType>
            </xs:element>
      </xs:sequence>
      <xs:attribute name="idx" use="required" fixed="1"/>
      <xs:attribute name="type" use="required" fixed="cell"/>
      <xs:attribute name="size" use="required"/>
      </xs:complexType>
      </xs:element>
      <xs:element name="no_G_terms">
      <xs:complexType>
      <xs:simpleContent>
            <xs:extension base="xs:integer">
                  <xs:attribute name="idx" use="required" fixed="1"/>
                  <xs:attribute name="type" use="required"
fixed="double"/>
                  <xs:attribute name="size" use="required" fixed="1
1"/>
            </xs:extension>
      </xs:simpleContent>
      </xs:complexType>

      </xs:element>

      <xs:element name="G">

      <xs:complexType>

      <xs:sequence>

            <xs:element name="item" maxOccurs="unbounded">

                  <xs:complexType>

                        <xs:simpleContent>

                              <xs:extension base="xs:string">

                                    <xs:attribute name="idx"
use="required"/>

                                    <xs:attribute name="type"
use="required" fixed="double"/>

                                    <xs:attribute name="size"
use="required"/>

                              </xs:extension>

                        </xs:simpleContent>

                  </xs:complexType>

            </xs:element>

      </xs:sequence>

      <xs:attribute name="idx" use="required" fixed="1"/>

      <xs:attribute name="type" use="required" fixed="cell"/>

      <xs:attribute name="size" use="required"/>

      </xs:complexType>

      </xs:element>
                                                  </xs:sequence>
                                                  <xs:attribute
name="idx" use="required" fixed="1"/>
                                                  <xs:attribute
name="type" use="required" fixed="struct"/>
                                                  <xs:attribute
name="size" use="required" fixed="1 1"/>
                                             </xs:complexType>
                                        </xs:element>
                                   </xs:sequence>
                                   <xs:attribute name="idx"
use="required" fixed="1"/>
                                   <xs:attribute name="type"
use="required" fixed="struct"/>
                                   <xs:attribute name="size"
use="required" fixed="1 1"/>
                              </xs:complexType>
                        </xs:element>
                        <xs:element name="rand_spl_info">
                              <xs:complexType>
                                    <xs:sequence>
                                          <xs:element name="fdp_type">
                                                <xs:complexType>
                                                      <xs:sequence>

      <xs:element name="item" maxOccurs="unbounded">

      <xs:complexType>

      <xs:simpleContent>

            <xs:extension base="xs:integer">

                  <xs:attribute name="idx" use="required"/>

                  <xs:attribute name="type" use="required"
fixed="double"/>

                  <xs:attribute name="size" use="required" fixed="1
1"/>

            </xs:extension>

      </xs:simpleContent>

      </xs:complexType>

      </xs:element>
                                                      </xs:sequence>
                                                      <xs:attribute
name="idx" use="required" fixed="1"/>
                                                      <xs:attribute
name="type" use="required" fixed="cell"/>
                                                      <xs:attribute
name="size" use="required"/>
                                                </xs:complexType>
                                          </xs:element>
                                          <xs:element name="mean">
                                                <xs:complexType>
                                                      <xs:sequence>

      <xs:element name="item" maxOccurs="unbounded">

      <xs:complexType>

      <xs:simpleContent>

            <xs:extension base="xs:double">

                  <xs:attribute name="idx" use="required"/>

                  <xs:attribute name="type" use="required"
fixed="double"/>

                  <xs:attribute name="size" use="required" fixed="1
1"/>

            </xs:extension>

      </xs:simpleContent>
      </xs:complexType>

      </xs:element>
                                                  </xs:sequence>
                                                  <xs:attribute
name="idx" use="required" fixed="1"/>
                                                  <xs:attribute
name="type" use="required" fixed="cell"/>
                                                  <xs:attribute
name="size" use="required"/>
                                            </xs:complexType>
                                      </xs:element>
                                      <xs:element
name="deviation">
                                            <xs:complexType>
                                                  <xs:sequence>

      <xs:element name="item" maxOccurs="unbounded">

      <xs:complexType>

      <xs:simpleContent>

            <xs:extension base="xs:double">

                  <xs:attribute name="idx" use="required"/>

                  <xs:attribute name="type" use="required"
fixed="double"/>

                  <xs:attribute name="size" use="required" fixed="1
1"/>

            </xs:extension>

      </xs:simpleContent>

      </xs:complexType>

      </xs:element>
                                                  </xs:sequence>
                                                  <xs:attribute
name="idx" use="required" fixed="1"/>
                                                  <xs:attribute
name="type" use="required" fixed="cell"/>
                                                  <xs:attribute
name="size" use="required"/>
                                            </xs:complexType>
                                      </xs:element>
                                </xs:sequence>
                                <xs:attribute name="idx"
use="required" fixed="1"/>
                                <xs:attribute name="type"
use="required" fixed="struct"/>
                                <xs:attribute name="size"
use="required" fixed="1 1"/>
                          </xs:complexType>
                    </xs:element>
                    <xs:element name="spl_data">
                          <xs:complexType>
                                <xs:sequence>
                                      <xs:element name="no_spl">
                                            <xs:complexType>
      <xs:simpleContent>

      <xs:extension base="xs:integer">

      <xs:attribute name="idx" use="required" fixed="1"/>

      <xs:attribute name="type" use="required" fixed="double"/>

      <xs:attribute name="size" use="required" fixed="1 1"/>

      </xs:extension>

      </xs:simpleContent>
                                                            </xs:complexType>
                                                      </xs:element>
                                                      <xs:element
name="spl_points">
                                                            <xs:complexType>
                                                                  <xs:sequence>

      <xs:element name="item" maxOccurs="unbounded">

      <xs:complexType>

      <xs:simpleContent>

            <xs:extension base="xs:string">

                  <xs:attribute name="idx" use="required"/>

                  <xs:attribute name="type" use="required"
fixed="double"/>

                  <xs:attribute name="size" use="required"/>

            </xs:extension>

      </xs:simpleContent>

      </xs:complexType>

      </xs:element>
                                                                  </xs:sequence>
                                                                  <xs:attribute
name="idx" use="required" fixed="1"/>
                                                                  <xs:attribute
name="type" use="required" fixed="cell"/>
                                                                  <xs:attribute
name="size" use="required"/>
                                                            </xs:complexType>
                                                      </xs:element>
                                                      <xs:element
name="snp_filename">
                                                            <xs:complexType>
                                                                  <xs:sequence>

      <xs:element name="item" maxOccurs="unbounded">

      <xs:complexType>

      <xs:simpleContent>

            <xs:extension base="xs:string">
```

```
                  <xs:attribute name="idx" use="required"/>

                  <xs:attribute name="type" use="required"
fixed="char"/>

                  <xs:attribute name="size" use="required"/>

            </xs:extension>

      </xs:simpleContent>

      </xs:complexType>

      </xs:element>
                                                </xs:sequence>
                                                <xs:attribute
name="idx" use="required" fixed="1"/>
                                                <xs:attribute
name="type" use="required" fixed="cell"/>
                                                <xs:attribute
name="size" use="required"/>
                                          </xs:complexType>
                                    </xs:element>
                                    <xs:element name="AFSflag">
                                          <xs:complexType>
                                                <xs:sequence>

      <xs:element name="item" maxOccurs="unbounded">

      <xs:complexType>

      <xs:simpleContent>

            <xs:extension base="xs:double">

                  <xs:attribute name="idx" use="required"/>

                  <xs:attribute name="type" use="required"
fixed="double"/>

                  <xs:attribute name="size" use="required" fixed="1
1"/>

            </xs:extension>

      </xs:simpleContent>

      </xs:complexType>

      </xs:element>
                                                </xs:sequence>
                                                <xs:attribute
name="idx" use="required" fixed="1"/>
                                                <xs:attribute
name="type" use="required" fixed="cell"/>
                                                <xs:attribute
name="size" use="required"/>
                                          </xs:complexType>
                                    </xs:element>
                                    <xs:element name="AFSmax">
                                          <xs:complexType>
                                                <xs:sequence>
      <xs:element name="item" maxOccurs="unbounded">

      <xs:complexType>

      <xs:simpleContent>

            <xs:extension base="xs:double">

                  <xs:attribute name="idx" use="required"/>

                  <xs:attribute name="type" use="required"
fixed="double"/>

                  <xs:attribute name="size" use="required" fixed="1
1"/>

            </xs:extension>

      </xs:simpleContent>

      </xs:complexType>

      </xs:element>
                                                      </xs:sequence>
                                                      <xs:attribute
name="idx" use="required" fixed="1"/>
                                                      <xs:attribute
name="type" use="required" fixed="cell"/>
                                                      <xs:attribute
name="size" use="required"/>
                                                </xs:complexType>
                                          </xs:element>
                                          <xs:element name="AFSerr">
                                                <xs:complexType>
                                                      <xs:sequence>

      <xs:element name="item" maxOccurs="unbounded">

      <xs:complexType>

      <xs:simpleContent>

            <xs:extension base="xs:double">

                  <xs:attribute name="idx" use="required"/>

                  <xs:attribute name="type" use="required"
fixed="double"/>

                  <xs:attribute name="size" use="required" fixed="1
1"/>

            </xs:extension>

      </xs:simpleContent>

      </xs:complexType>

      </xs:element>
                                                      </xs:sequence>
                                                      <xs:attribute
name="idx" use="required" fixed="1"/>
```

```
                                                        <xs:attribute
name="type" use="required" fixed="cell"/>
                                                        <xs:attribute
name="size" use="required"/>
                                                  </xs:complexType>
                                            </xs:element>
                                      </xs:sequence>
                                      <xs:attribute name="idx"
use="required" fixed="1"/>
                                      <xs:attribute name="type"
use="required" fixed="struct"/>
                                      <xs:attribute name="size"
use="required" fixed="1 1"/>
                                </xs:complexType>
                          </xs:element>
                          <xs:element name="nominal_matrices">
                                <xs:complexType>
                                      <xs:sequence>
                                            <xs:element name="filename">
                                                  <xs:complexType>

      <xs:simpleContent>

      <xs:extension base="xs:string">

      <xs:attribute name="idx" use="required" fixed="1"/>

      <xs:attribute name="type" use="required" fixed="char"/>

      <xs:attribute name="size" use="required"/>

      </xs:extension>

      </xs:simpleContent>
                                                  </xs:complexType>
                                            </xs:element>
                                      </xs:sequence>
                                      <xs:attribute name="idx"
use="required" fixed="1"/>
                                      <xs:attribute name="type"
use="required" fixed="struct"/>
                                      <xs:attribute name="size"
use="required" fixed="1 1"/>
                                </xs:complexType>
                          </xs:element>
                          <xs:element name="nominal_freq_response">
                                <xs:complexType>
                                      <xs:sequence>
                                            <xs:element
name="snp_filename">
                                                  <xs:complexType>

      <xs:simpleContent>

      <xs:extension base="xs:string">

      <xs:attribute name="idx" use="required" fixed="1"/>

      <xs:attribute name="type" use="required" fixed="char"/>

      <xs:attribute name="size" use="required"/>

      </xs:extension>
      </xs:simpleContent>
                                                <xs:complexType>
                                          </xs:element>
                                          <xs:element name="AFSflag">
                                                <xs:complexType>

      <xs:simpleContent>

      <xs:extension base="xs:double">

      <xs:attribute name="idx" use="required" fixed="1"/>

      <xs:attribute name="type" use="required" fixed="double"/>

      <xs:attribute name="size" use="required" fixed="1 1"/>

      </xs:extension>

      </xs:simpleContent>
                                                </xs:complexType>
                                          </xs:element>
                                          <xs:element name="AFSmax">
                                                <xs:complexType>

      <xs:simpleContent>

      <xs:extension base="xs:double">

      <xs:attribute name="idx" use="required" fixed="1"/>

      <xs:attribute name="type" use="required" fixed="double"/>

      <xs:attribute name="size" use="required" fixed="1 1"/>

      </xs:extension>

      </xs:simpleContent>
                                                </xs:complexType>
                                          </xs:element>
                                          <xs:element name="AFSerr">
                                                <xs:complexType>

      <xs:simpleContent>

      <xs:extension base="xs:double">

      <xs:attribute name="idx" use="required" fixed="1"/>

      <xs:attribute name="type" use="required" fixed="double"/>

      <xs:attribute name="size" use="required" fixed="1 1"/>

      </xs:extension>

      </xs:simpleContent>
                                                </xs:complexType>
                                          </xs:element>
                                    </xs:sequence>
                                    <xs:attribute name="idx"
use="required" fixed="1"/>
                                    <xs:attribute name="type"
use="required" fixed="struct"/>
```

```
                                    <xs:attribute name="size"
use="required" fixed="1 1"/>
                              </xs:complexType>
                        </xs:element>
                        <xs:element name="spl_matrices">
                              <xs:complexType>
                                    <xs:sequence>
                                          <xs:element
name="filenames">
                                                <xs:complexType>
                                                      <xs:sequence>

      <xs:element name="item" maxOccurs="unbounded">

      <xs:complexType>

      <xs:simpleContent>

            <xs:extension base="xs:string">

                  <xs:attribute name="idx" use="required"/>

                  <xs:attribute name="type" use="required"
fixed="char"/>

                  <xs:attribute name="size" use="required"/>

            </xs:extension>

      </xs:simpleContent>

      </xs:complexType>

      </xs:element>
                                                      </xs:sequence>
                                                      <xs:attribute
name="idx" use="required" fixed="1"/>
                                                      <xs:attribute
name="type" use="required" fixed="cell"/>
                                                      <xs:attribute
name="size" use="required"/>
                                                </xs:complexType>
                                          </xs:element>
                                    </xs:sequence>
                                    <xs:attribute name="idx"
use="required" fixed="1"/>
                                    <xs:attribute name="type"
use="required" fixed="struct"/>
                                    <xs:attribute name="size"
use="required" fixed="1 1"/>
                              </xs:complexType>
                        </xs:element>
                        <xs:element name="sensC">
                              <xs:complexType>
                                    <xs:sequence>
                                          <xs:element
name="filenames">
                                                <xs:complexType>
                                                      <xs:sequence>

      <xs:element name="item" maxOccurs="unbounded">

      <xs:complexType>
      <xs:simpleContent>

            <xs:extension base="xs:string">

                  <xs:attribute name="idx" use="required"/>

                  <xs:attribute name="type" use="required"
fixed="char"/>

                  <xs:attribute name="size" use="required"/>

            </xs:extension>

      </xs:simpleContent>

      </xs:complexType>

      </xs:element>
                                                      </xs:sequence>
                                                      <xs:attribute
name="idx" use="required" fixed="1"/>

                                                      <xs:attribute
name="type" use="required" fixed="cell"/>

                                                      <xs:attribute
name="size" use="required"/>
                                                </xs:complexType>
                                          </xs:element>
                                    </xs:sequence>
                                    <xs:attribute name="idx"
use="required" fixed="1"/>
                                    <xs:attribute name="type"
use="required" fixed="struct"/>
                                    <xs:attribute name="size"
use="required" fixed="1 1"/>
                              </xs:complexType>
                        </xs:element>
                        <xs:element name="sensG">
                              <xs:complexType>
                                    <xs:sequence>
                                          <xs:element
name="filenames">
                                                <xs:complexType>
                                                      <xs:sequence>

      <xs:element name="item" maxOccurs="unbounded">

      <xs:complexType>

      <xs:simpleContent>

            <xs:extension base="xs:string">

                  <xs:attribute name="idx" use="required"/>

                  <xs:attribute name="type" use="required"
fixed="char"/>

                  <xs:attribute name="size" use="required"/>

            </xs:extension>

      </xs:simpleContent>
      </xs:complexType>

      </xs:element>
                                                                  </xs:sequence>
                                                                  <xs:attribute
name="idx" use="required" fixed="1"/>
                                                                  <xs:attribute
name="type" use="required" fixed="cell"/>
                                                                  <xs:attribute
name="size" use="required"/>
                                                            </xs:complexType>
                                                      </xs:element>
                                                </xs:sequence>
                                                <xs:attribute name="idx"
use="required" fixed="1"/>
                                                <xs:attribute name="type"
use="required" fixed="struct"/>
                                                <xs:attribute name="size"
use="required" fixed="1 1"/>
                                          </xs:complexType>
                                    </xs:element>
                                    <xs:element name="prj_matrix">
                                          <xs:complexType>
                                                <xs:sequence>
                                                      <xs:element name="filename">
                                                            <xs:complexType>

      <xs:simpleContent>

      <xs:extension base="xs:string">

      <xs:attribute name="idx" use="required" fixed="1"/>

      <xs:attribute name="type" use="required" fixed="char"/>

      <xs:attribute name="size" use="required"/>

      </xs:extension>

      </xs:simpleContent>
                                                            </xs:complexType>
                                                      </xs:element>
                                                </xs:sequence>
                                                <xs:attribute name="idx"
use="required" fixed="1"/>
                                                <xs:attribute name="type"
use="required" fixed="struct"/>
                                                <xs:attribute name="size"
use="required" fixed="1 1"/>
                                          </xs:complexType>
                                    </xs:element>
                              </xs:sequence>
                              <xs:attribute name="xml_tb_version" use="required"
fixed="3.1"/>
                              <xs:attribute name="idx" use="required" fixed="1"/>
                              <xs:attribute name="type" use="required"
fixed="struct"/>
                              <xs:attribute name="size" use="required" fixed="1
1"/>
                  </xs:complexType>
      </xs:element>

</xs:schema>
```

B.2 XSD File for Interconnected Parametric Systems

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2006 rel. 3 sp2 (http://www.altova.com) by
Gabriela Ciuprina (Politehnica University of Bucharest) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
      <xs:element name="root">
            <xs:annotation>
                  <xs:documentation>Interconnected system, in
accordance with version 7.0 of the documentation</xs:documentation>
            </xs:annotation>
            <xs:complexType>
                  <xs:sequence>
                        <xs:element name="type" fixed="mvar">
                              <xs:complexType>
                                    <xs:simpleContent>
                                          <xs:extension
base="xs:string">
                                                <xs:attribute
name="idx" use="required" fixed="1"/>
                                                <xs:attribute
name="type" use="required" fixed="char"/>
                                                <xs:attribute
name="size" use="required" fixed="1 4"/>
                                          </xs:extension>
                                    </xs:simpleContent>
                              </xs:complexType>
                        </xs:element>
                        <xs:element name="ver" fixed="7.0">
                              <xs:complexType>
                                    <xs:simpleContent>
                                          <xs:extension
base="xs:double">
                                                <xs:attribute
name="idx" use="required" fixed="1"/>
                                                <xs:attribute
name="type" use="required" fixed="double"/>
                                                <xs:attribute
name="size" use="required" fixed="1 1"/>
                                          </xs:extension>
                                    </xs:simpleContent>
                              </xs:complexType>
                        </xs:element>
                        <xs:element name="id">
                              <xs:complexType>
                                    <xs:simpleContent>
                                          <xs:extension
base="xs:string">
                                                <xs:attribute
name="idx" use="required" fixed="1"/>
                                                <xs:attribute
name="type" use="required" fixed="char"/>
                                                <xs:attribute
name="size" use="required"/>
                                          </xs:extension>
                                    </xs:simpleContent>
                              </xs:complexType>
                        </xs:element>
                        <xs:element name="no_sys">
                              <xs:complexType>
                                    <xs:simpleContent>
```

```
                                          <xs:extension
base="xs:integer">
                                                <xs:attribute
name="idx" use="required" fixed="1"/>
                                                <xs:attribute
name="type" use="required" fixed="double"/>
                                                <xs:attribute
name="size" use="required" fixed="1 1"/>
                                          </xs:extension>
                                    </xs:simpleContent>
                              </xs:complexType>
                        </xs:element>
                        <xs:element name="components">
                              <xs:complexType>
                                    <xs:sequence>
                                          <xs:element name="filename">
                                                <xs:complexType>
                                                      <xs:sequence>

      <xs:element name="item" maxOccurs="unbounded">

      <xs:complexType>

      <xs:simpleContent>

            <xs:extension base="xs:string">

                  <xs:attribute name="idx" use="required"/>

                  <xs:attribute name="type" use="required"
fixed="char"/>

                  <xs:attribute name="size" use="required"/>

            </xs:extension>

      </xs:simpleContent>

      </xs:complexType>

      </xs:element>
                                                      </xs:sequence>
                                                      <xs:attribute
name="idx" use="required" fixed="1"/>
                                                      <xs:attribute
name="type" use="required" fixed="cell"/>
                                                      <xs:attribute
name="size" use="required"/>
                                                </xs:complexType>
                                          </xs:element>
                                    </xs:sequence>
                                    <xs:attribute name="idx"
use="required" fixed="1"/>
                                    <xs:attribute name="type"
use="required" fixed="struct"/>
                                    <xs:attribute name="size"
use="required" fixed="1 1"/>
                              </xs:complexType>
                        </xs:element>
                        <xs:element name="mtx_intcnt">
                              <xs:complexType>
                                    <xs:sequence>
                                          <xs:element name="filename">
                                                <xs:complexType>
      <xs:simpleContent>

      <xs:extension base="xs:string">

      <xs:attribute name="idx" use="required" fixed="1"/>

      <xs:attribute name="type" use="required" fixed="char"/>

      <xs:attribute name="size" use="required"/>

      </xs:extension>

      </xs:simpleContent>
                                                </xs:complexType>
                                          </xs:element>
                                    </xs:sequence>
                                    <xs:attribute name="idx"
use="required" fixed="1"/>
                                    <xs:attribute name="type"
use="required" fixed="struct"/>
                                    <xs:attribute name="size"
use="required" fixed="1 1"/>
                              </xs:complexType>
                        </xs:element>
                  </xs:sequence>
                  <xs:attribute name="xml_tb_version" use="required"
fixed="3.1"/>
                  <xs:attribute name="idx" use="required" fixed="1"/>
                  <xs:attribute name="type" use="required"
fixed="struct"/>
                  <xs:attribute name="size" use="required" fixed="1
1"/>
            </xs:complexType>
      </xs:element>
</xs:schema>
```