Politecnico di Milano

School of Industrial Engineering

Master of Science in Mechanical Engineering

UNCERTAINTY QUANTIFICATION IN PIPE FLOW SYSTEMS

Supervisor: Prof. Paolo Trucco, Politecnico di Milano

Co-Supervisor: Dr Athanasios Kolios, Cranfield University UK

Masters Degree Thesis:

Daniele Casali 780856

Academic Year: 2012 - 2013

Politecnico di Milano

School of Industrial Engineering

Master of Science in Mechanical Engineering

UNCERTAINTY QUANTIFICATION IN PIPE FLOW SYSTEMS

Supervisor: Prof. Paolo Trucco, Politecnico di Milano

Co-Supervisor: Dr Athanasios Kolios, Cranfield University UK

Masters Degree Thesis:

Daniele Casali 780856

Academic Year: 2012 - 2013

This thesis is submitted in partial fulfilment of the requirements
for the degree of Master of Science

ABSTRACT

Pipe flow systems are used in a vast variety of applications, ranging from large scale oil and gas and refineries pipelines to very small scale application such as those used for medical applications. Design of such systems is currently governed by standards and common practices. Although they provide systems of adequate safety, they don't allow a systematic assessment of their real-time performance due to presence of various sources of uncertainty. A detailed analysis aiming to quantify the impact of these uncertainties can lead to a more efficient design, more reliable systems and reduced maintenance requirements. Surface roughness, pressure drop coefficients, flow properties, corrosion deterioration etc, are only few of the variables governed by a high degree of randomness. This work has been developed with particular reference to multiphase flow systems where above mentioned uncertainties are a major issue for safe operation.

The field of pipelines and piping systems in general is dominated by qualitative methods and no quantitative reliability assessment at the design stage has been applied yet. This work presents a systematic methodology to assess quantitatively the impact of uncertainties on the system so as to achieve a better understanding of the system performance. The results obtained can influence decision making in operation and maintenance of single system components.

Here two different scales of flow systems are considered; one for large scale and another one for small scale applications, and the applicability of the method proposed was determined for both. The methodology that is proposed considers both direct numerical simulations and different response surface methods comparing their effectiveness and benchmarking their performance in terms of computational time and accuracy in the results. Particularly polynomial and non-linear response surfaces are considered and then an estimate of the probability of exceedance of a permissible threshold is obtained through First Order Reliability Method, which result is heavily dependent on the accuracy of the response surface used. With these techniques it is also possible to know which points of the system are less reliable, thus allowing improvements in system safe operability.

The methodology that is presented herein can be extended for different relevant problems encountered in engineering and scientific applications.

Keywords:

Quantitative Method for Probabilistic Assessment, Surrogate Modelling, Kriging, Dynamically Kriged Limit State, Pipeline, Predictive Engineering

ACKNOWLEDGEMENTS

Firstly I would like to deeply thank Prof Paolo Trucco for giving me the opportunity to develop this Thesis abroad. Secondly I want to thank Dr Athanasios Kolios for giving me guidance at Cranfield Uni.

I have to thank my parents for their undying efforts in raising me and being my very first models. Their integrity, proneness to sacrifice and ambitiousness are totally reflected in me and I hope to be a source of pride for them. Thank you for your moral and financial support and for selflessly granting me an opportunity that was never afforded to you. Thanks also for supporting me in all my decisions and letting me go on my own way. A special thank you is also extended to my 'granny' whom I respect too much to thank in English. Grazie anche a mia nonna esempio di umiltà, laboriosità e forza d'animo. Thanks also to the rest of the family whose support I know I can rely on.

Lastly I want to thank all my friends for letting me open some windows on the world: engineering is cool but is not the only goggle through which analysing things. I want to add a special thank you to the ones who fuelled, driven by my own life philosophy, some fleeting moments of carefreeness we shared.

All of you together contributed to what I am and what I achieved.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

Fluids are widely used and processed in many industrial fields. They often have to cover long distances between process operations or process units and they are handled and transported thanks to piping systems. Many different kind of processes and utilities are possible thanks to pipe networks: water distribution, refrigeration systems, fire protection systems, steam and compressed air utilities, waste and storm water, oil systems, lubrication circuits, gases and flue gases.

These fluids are transported across a plant or a site thanks to pipes which are selected in accordance with the characteristics of the fluid and the operating conditions of the same (flow rate, pressure, temperature). Pipe selection procedure aims at choosing the best material, pipe diameter, pipe thickness for the current application. Pipes are connected and arranged in layouts according to the physical constraints and barriers on site by means of bends and fittings having various angles and pressure losses. Flow is controlled and diverted thanks to valves which exist in different shapes and with different functions (gate, globe and check just to mention some). In most occasions straight pipes are joined by means of a flange-gasket-flange assembly.

During the design of this kind of system many different variables are to be measured and accuracy in this phase is required to minimise the final uncertainty of the system. Randomness in preliminary phases will result in applying safety factors, over-sizing components hence loss of money due to uncertainty.

During the operation phase of the system uncertainties are even more present because is even more difficult to measure correctly key variables while the system is operating. For example pipe inner wall roughness or pipe thickness are variables largely affected by operating conditions and maintenance plans are required to inspect and replace components every now and then. This, as said, is a result of the intrinsic ignorance around the process itself. For example flow rate, pressure, temperature, surface roughness and wall thickness are some of the important quantities to be measured during the operation of such systems and they all affect the way the circuit safely operates.

A large span of systems present the same uncertainty problem ranging from small medical application systems through water distribution ones and industrial utilities to large oil and gas pipelines.

1.1 **Background**

In order to face the problem and dealing in a smart way with uncertainties several methodologies have been developed throughout the years.

Exploring the literature we can realise that many efforts have been put in qualitative approaches developing case-dependent methodologies based on historical data (WOAD, Oreda etc...) and personal experience. Some of those techniques such as RAM (Reliability Availability and Maintainability analysis), RCM (Reliability-Centered Maintenance), FMEA/FMECA (Failure Modes and Effects Analysis/Failure Modes,

Criticality and Effect Analysis) and FTA (Fault Tree Analysis) are referred to as traditional techniques. Together with innovative techniques such as R6 and GO-FLOW they are applied to get a qualitative estimate of the reliability of a system. Even if these are very rigorous procedures, the output they give is highly subjective and dependant on the person carrying out the analysis. Moreover, if this subjectivity has to be removed historical data can be used and efficiently manipulated. Globally: a) this procedures are only possible for already known systems for which past data are available; b) the qualitative evaluation is highly time consuming and still doesn't give us a number to rely on.

Particular focus has been put on leakage detection, which is somehow different from the scope of this work, and corrosion impact estimation. For the first purpose Bayesian Networks have been successfully applied to pipelines [8] and a comparative study on different corrosion models and assessment of corrosion impact on failure has been carried out [13].

Some quantitative analysis has been done using cumbersome simulation methods such as Monte Carlo Simulation technique, which is easy-to-implement and uses a simple algorithm but has the drawback of a long simulation time. Some other quantitative studies have been carried out with the support of DnV Proban © software package in [34] and [13], basically to solve the joint probability integral of the probability of failure (see Chapter 4 for further details on this aspect).

A Markov model has been developed in [52] for the analysis of in-service inspection strategies for nuclear power plant piping systems which has been focusing on the transition probability between states of the system (success, flow, leak and rupture).

As said corrosion is a main issue in the operability of piping systems, especially in offshore pipeline application where both the harsh environment and the fluid flowing in the pipe, with several and different mechanisms, contribute to pipe deterioration. In literature a lot of material is found to estimate pipe wall corrosion and a lot of work is carried out to assess the reliability and safe operability of a worn out pipe wall in ASME B31G-2009 and in [2, 4, 5, 6, 7]. Most of the times the assessment is based on inspection and no stochasticity of variables is considered. In some cases quantitative assessment is performed but only direct simulation techniques are employed.

Also published data can be found about corrosion rates estimation, for example for CO_2 corrosion (e.g. corrosion due to CO_2 content in natural gas flowing in the pipe).

Summarizing what found in literature, we can state that:

- no systematic and analytical methods have been developed yet
- largely qualitative methods are applied
- most techniques available are based on historical data, which is a limiting factor for the development of innovative systems
- some quantitative techniques are available out there but they imply inacceptable simulation times and are not effective with low probabilities of failure (e.g. 10^{-3}, 10^{-4}) [33]
- focus on corrosion impact and corrosion prediction

1.2 **Measuring techniques and sources of uncertainty**

Flow rate can be measured with many different kind of devices based on several principles. Classic flow meters include differential pressure and positive displacement devices such as Venturi tube, orifice plate, V-cone, drag-plate, rotameter, turbine and diaphragm flow meters. Modern flow meters include vortex, electromagnetic, ultrasonic, Coriolis and thermal capacity meters. Many factors affect the reliability of the measurements obtained and they change mainly according to the operating principle of the flow meter. For example fluid conductivity, density and viscosity just to mention some, are affecting it. Moreover each measured value is affected by the velocity profile of the flow passing through it and this parameter is affected by the installation conditions of the meter itself. For example, if a flow meter is installed downstream two 90° bends placed in orthogonal planes the well-known swirl effect occurs and bad measurements ensue from it. Another source of uncertainty may be the corrosion or fouling of the inner pipe wall from which bad estimation of the cross sectional area and finally of the flow rate derive. As said fluid properties affect the measurement and many devices working principles are based on them. If fluid properties are badly measured or estimated or are changing through the process a bad final measurement follows.

Surface roughness of the inner pipe wall can be measured with different techniques using contact and non-contact devices. Profilometers (also known as styluses or phonographs) and atomic force microscopes are belonging to the former, optical and laser to the latter (ISO 25178). Moreover, because of the irregularity of the profiles, data elaboration is needed to achieve a meaningful result. This procedure is only possible if the surface to be evaluated is directly accessible by the instrument, if not no data can be drawn. This situation occurs when the component is in operation and is highly affecting the degree of randomness of the problem. Since the roughness is directly proportional to the friction coefficient hence to the frictional head loss, this parameter heavily affects the operation mode of the system and its output. Throughout the life of the system and because of the interaction material-fluid, fouling resulting in a non-quantifiable change of roughness occurs.

If the fluid is corrosive to the material, extensive consumption of the pipe inner wall might happen. This phenomenon is not quantifiable because the measurement of the actual inner pipe diameter while the component is in operation is not possible. A too thin pipe wall thickness may affect the operability of the pipe in terms of allowable operating pressure, affecting in this way the limit states e.g. so-called burst pressure.

Some other uncertainty can be brought in the design process by the pump characteristic. Because of ageing of the pump, intrinsic periodicity of the machine, lack of precision in testing the pump in order to build experimentally the characteristic curve and some other minor issues the information given by the manufacturer is always subjected to uncertainty. In particular for the last point stressed ISO 9906-2012 is a good reference and gives us a global picture of the problem. Pumps can be divided in three categories according to the accuracy in their characteristic: Grade 1, Grade 2 and Grade 3 which have respectively 3, 4 and 10% uncertainty in the differential head.

Pipe fittings determine further uncertainties in the model because the losses across them are obtained in a similar experimental way as the pump differential head is obtained. Basically the pressure is measured before and after the device and the loss is then

calculated. Some experimental correlation between variables is always done by both manufacturers and pure academics as shown in [10, 12]. Anyway some uncertainty is always present because of the accuracy in these correlation derivation processes and because of changes in operating condition or device preservation.

Many other variables, which are measured indirectly, present an intrinsic uncertainty. Some of them which can be taken into consideration in our case we find the overall heat transfer coefficient, the yield strength of a material etc…

In the next subchapter a definition of the uncertainties seen overviewed up to now is done and in this way the inputs to our problems are quantitatively shaped.

1.3 **Uncertainties definition**

Since what is done in this section will affect the results obtained is important to first stress some points. Without having any experimental data from which fit a distribution for the considered input variables, assumptions on their distributions are done. Basically they are all considered to be normally distributed around the mean and values for standard deviation are derived from typical measurement accuracy as follows. If a variable, from actual data, is found to be distributed in a different way transformations can be applied to normalise the variable as shown in [9, 53].

According to measurements standards every value for accuracy is given at interval of confidence 95%, which in a normal distribution corresponds to +/- 2σ. If we indicate with a the accuracy of the measurement;

$$\pm a = \pm 2\sigma \tag{1.1}$$

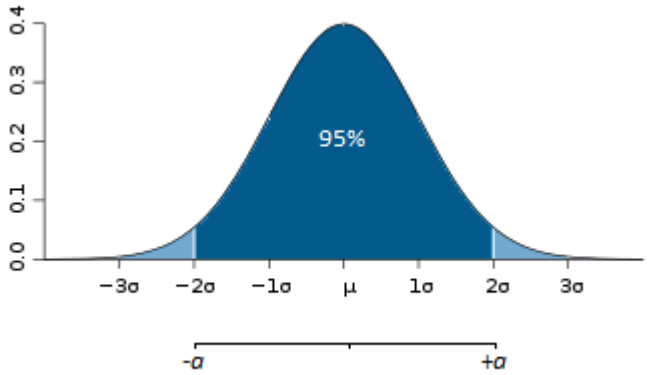


Figure 1.1 Accuracy in the measurement affects variable uncertainty

If a variable is not directly measured a reasonable coefficient of variation CoV=10% is considered. Uncertainties in the considered variables are gathered in Table 1.1. Later on these will be referred to actual design values for each variable.

Table 1.1. Typical accuracies

	Accuracy (on reading)	Reference
Pressure	0.50%	[43]
Temperature	0.20%	[43]
Flow rate	0,1-1%	[44]
Roughness	10-12%	[43]
Level	0,5-1%	[16]
Pump characteristic	10%	ISO 9906
Valve pressure drop	1,5-3%	[34]
Multiphase flow rate	0,15-0,5%	[49]
Well pressure	20%	assumed
Well temperature	5-10%	assumed
Sea water temperature	~1,5%	[47]
Heat transfer coefficient	20%	assumed

1.4 **Aims & Objectives**

The aim of this work is to apply a generic procedure for Reliability Analysis to a pipe flow system. Some methodologies are already available and largely applied in structural reliability, but they show some drawbacks.

In this work some of these problems are tackled and solutions are found.

The main focus of the work is on one-phase flow systems but the generic procedure is also applied to a two-phase flow system. This is done just to show the wide applicability of the method also to different systems, being a system "*a set of things working together as parts of a mechanism or an interconnecting network, a complex whole"*. [50]. Once the methodology is developed and shown to be applicable, the reader could be able to use such tools for a more accurate analysis.

As said first a one-phase flow system is analysed, in particular the test rig used in the FlowLab, Cranfield University.

Then a fictitious but realistic pipeline plus riser from wellhead to platform is investigated. Here also a corrosion rate model is embedded in the probabilistic analysis. What we would like to obtain from this analysis is a prediction on failure in the pipeline due to exceeded burst pressure (depending on corroded thickness during pipeline expected life).

2 PIPE SYSTEMS MODELLING

2.1 **One-phase system**

The system has been modelled using different tools: Matlab and Pipe Flow Expert 2010, available in a trial version. Comparing the responses to different inputs from a commercial software package is possible to perform a validation of the Matlab analytical model. Having different operating conditions for the system a different output is obtained and comparing the output from the Matlab code with the one coming from the commercial software package we can state if the code is right or wrong, accurate or not.

A reliable code is necessary for our application because we can shape it in whatever way we want giving as input variables those quantities which present an intrinsic uncertainty (e.g. pressure, flow rate, roughness, pressure drop coefficients...). Commercial packages are all working in the other way round: specifying two pressures, or tank levels, across a geometrically defined system they calculate the flow rate necessary to obtain these conditions, being the pressure difference driving the flow. This is maybe less intuitive but it's the way nature works so software packages have been developed in this way.

Since both a global solution of the system (from differential pressure to flow rate) and a partial solution (from flow rate and one pressure to the missing one) are needed to analyse the system, two generic analytical models are developed. An analytical derivation of the basic equation for the two is given from now on and the validation procedure followed for both is given later on.

2.2 **Analytical model**

The model built in Matlab is based on well-known fluid mechanics basic equations. Having a system, or a sub-system, delivering a flow from an inlet node to an outlet node we can define its geometry specifying the quantities below:

- L, straight length [m]
- h, elevation [m]
- D, pipe diameter [cm]
- ε, pipe roughness [mm]
- number and type of bends and valves
- k_{comp}, component losses (from a pump, a measuring instrument..)
- pump characteristic

Inlet conditions of the operating fluid have to be specified as well:

- pressure [Pa]
- flow rate [m^3/s]

- temperature [K]

Before analysing the solution procedure for the circuit it is appropriate to explain properly some of these variables. The elevation is considered positive if increasing moving from the inlet node to the outlet node. The pipe roughness is dependent from the material and, as said, can change during the system life and so represent a source of uncertainty. Bends and valves in a piping system have a huge variability and building a code taking into account even half of them is first of all useless and second of all too cumbersome. Only few of them (3 bends and 3 valves types) are considered and the code will ask the number of each of them present in the system. A free variable is left where a total sum of all other component losses can be put allowing in this way flexibility and completeness to the code. A pump characteristic can be defined with half a parabolic curve symmetrical to the y-axis and concave downwards, hence only two coefficients are needed to define its behaviour (pump head vs. flow rate). No heat loss is considered from the pipe (temperature constant).

According to the Bernoulli theorem, for a perfect incompressible fluid in permanent motion that doesn't exchange energy in whatever form with the surroundings and with straight and parallel trajectories:

$$\frac{\partial H}{\partial s} = 0 \tag{2.1}$$

where *s* is an infinitesimal length of the trajectory and *H*;

$$H = z + \frac{p}{\gamma} + \frac{v^2}{2g} \tag{2.2}$$

Basically, to solve the system the energy conservation equation has to be applied to our case and solved:

$$H_{in} + H_{pump} = H_{out} + H_{loss} \tag{2.3}$$

where;

$$H_{pump} = a - b\dot{V}^2 \tag{2.4}$$

$$H_{loss} = H_{frictional} + H_{component} \tag{2.5}$$

Frictional losses represent the portion of head loss occurring because of the presence of a component where the flow is disturbed and turbulence is induced e.g. a bend, a valve... They can be generally defined as;

$$H_{component} = \sum k \frac{v^2}{2g} \tag{2.6}$$

where *k* is a typical value for each fitting which can be either found in [12] or in the PFE2010 database.

Component losses can be defined according to the Darcy-Weisbach equation [10];

$$H_{frictional} = fL \frac{v^2}{2gD} \tag{2.7}$$

where *f* is defined as a friction factor coefficient for which several formulations are given. Below are reported the Colebrook-White implicit formulation and the Haaland explicit one;

$$\frac{1}{\sqrt{f}} = -2 \log_{10} \left[\frac{\varepsilon}{3.71D} + \frac{2.51}{Re\sqrt{f}} \right] \tag{2.8}$$

$$\frac{1}{\sqrt{f}} = -1.8 \log_{10} \left[\left(\frac{\varepsilon}{3.7D} \right)^{1.11} + \frac{6.9}{Re} \right] \tag{2.9}$$

either the first one or the second one can be used, obtaining results of the same accuracy [10]. The first one is implicit i.e. *f* cannot be derived explicitly because on both sides of the equal sign, hence for easiness of solution the assumption of fully turbulent flow i.e. $Re \rightarrow \infty$ can be done so that the second term in the logarithm can be neglected and an explicit formulation is obtained. A check on the acceptability of the assumptions has to be done at the end of the routine, particularly the ones about incompressibility of the fluid and fully turbulent flow. Further assumptions are made about fluid incompressibility and they also have to be checked.

If the aim is to globally solve the circuit, so to obtain flow rate value knowing the differential pressure the following equation applies;

$$v = \sqrt{\frac{\frac{\Delta p}{\gamma} + (z_{out} - z_{in}) - a}{-b\dot{V}^2 - \frac{fL}{2gD} - \sum \frac{k_i}{2g}}} \tag{2.10}$$

then simply multiplying by the cross sectional area we can obtain the value for the flow rate.

Considering the partial solution of the circuit, so knowing the flow rate and wanting to obtain the outlet pressure, the condition at the outlet is obtained solving the energy balance equation given above;

$$p_{out} = \gamma \left[\frac{p_{in}}{\gamma} + (z_{in} - z_{out}) + a - b\dot{V}^2 - H_{loss} \right] \tag{2.11}$$

Density and viscosity of the fluid have to be estimated and simple subroutines are built considering properties from [46]. Matlab code is reported in Appendix C, the first global solver is called *dp2fr.m* (differential pressure to flow rate) and the second one is called *fr2pout.m* (flow rate to outlet pressure).

2.3 **Commercial software**

Pipe Flow Expert 2010 has been chosen to validate the Matlab code because a free trial version is available. The software works in steady-state only and has two environments: the designer and the solver.

In the first one fluid characteristic (type, temperature) and features of the circuit can be chosen. Modelling the transportation of a fluid from a node to another, situation which can be seen both as a system or a sub-system, fittings and valves can be specified choosing in the available library or customising them. Pumps can be placed along the pipe specifying the characteristic curve (reported in Appendix A), the efficiency curve and the $NPSH_r$ curve point-by-point. Pump data have been assumed reasonably considering a medium application as a 100mm diameter impeller pump, running constantly at 3450rpm. Curves and data are reported in Appendix A.

In this situation we don't need to consider a particular layout detailing how fittings and valves come one after the other since a general solution equation for outlet values is used by the solver. If we are particularly interested in a point of a circuit we can split it into two sub-systems and solve them in series, giving the output from the first one as an input for the second one.

Unfortunately, the logic behind this software is different from the one implemented in Matlab *fr2pout.m* script as here pressure specifications at nodes or level specification for tanks are given as an input and the flow rate, driven by pressure, is then calculated

by means of the Hardy-Cross method. In the model validation procedure section details about matching the two worlds are given.

In the second environment detailed calculations are reported and there is the possibility to edit and export results sheets.

2.4 **Model validation procedure**

After having the Matlab code and a test-bench circuit in PFE2010 many different scenarios have been created playing with circuit features and operating conditions.

In order to match the two worlds the model in PFE2010 has been built after the Matlab run. Tanks are placed at inlet and outlet nodes and their level is chosen to obtain a certain pressure at the bottom of the tank. For the inlet tank inlet pressure is considered, for the outlet tank the result from the Matlab run is imposed. Running the PFE model we obtain a flow rate as an output and comparing this to the one given as an input to Matlab we can see if the models match or not. A visualisation of this procedure is given in Figure 2.1 and Figure 2.2 below.

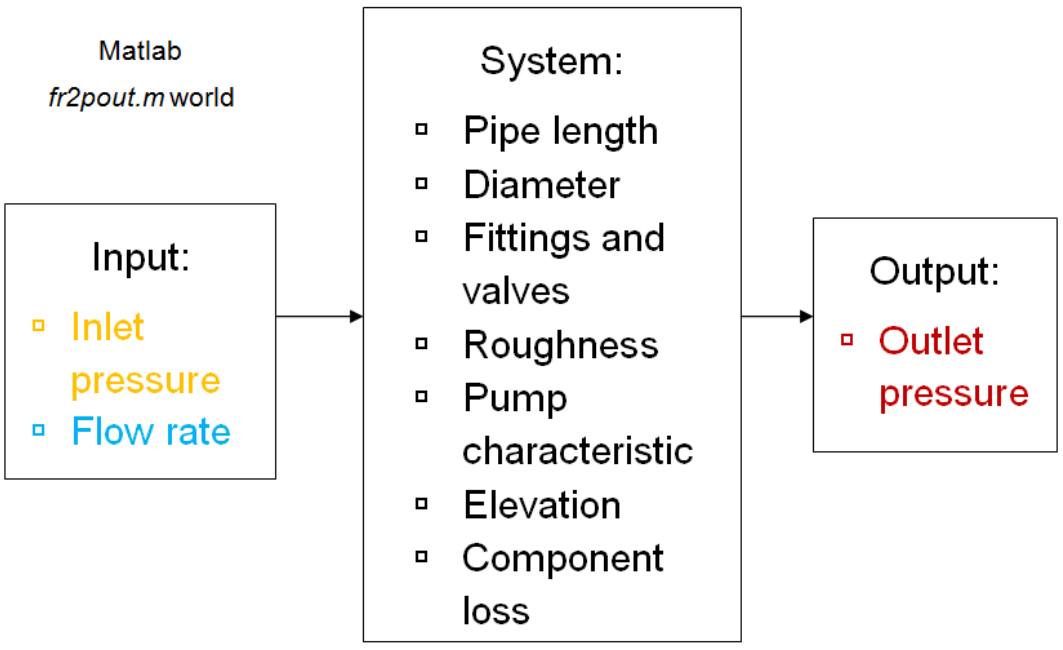


Figure 2.1. *fr2pout.m* logic

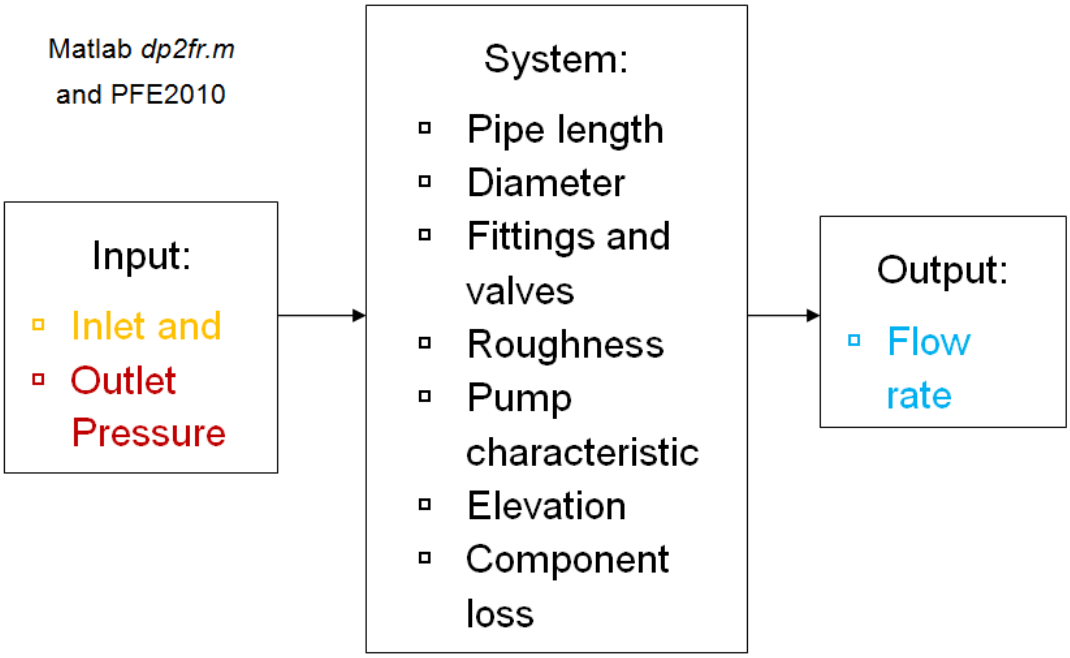


Figure 2.2. *dp2fr.m* logic

A validation of the model has been done cross-checking results obtained from several scenarios. Validation data are reported in Appendix A. A further check using formulas given in the previous section has been done by hand calculations. By means of hand calculations further confidence is obtained in the tools developed and additional understanding is obtained.

2.5 **Two-phase system**

The two-phase system analysed is modelled in a specialised software for offshore pipeline applications, which is OLGA © from SPT Group. This particular software is strong in slug detection and slugging mechanisms modelling, hence on dynamic flow modelling. In this work just steady state simulation will be run since slug control and detection go beyond the scope of this Thesis.

The software permits us to specify the geometry of the pipeline giving coordinates of pipe sections ending points. Data on pipe diameter, thickness, material and roughness can also be given. Each pipe section can be divided into segments, basically integration points where the software will compute fluid properties.

Giving estimated heat transfer coefficients, surrounding environment temperatures and piping material conductivity the software is also able to compute the heat loss in the fluid and its temperature in each segment.

Basically, the model implemented in the software is a modified version of a complete example given in the OLGA © manual [51]. This example is expanded and modified in order to perform probabilistic analysis

Having a wellhead we need to connect it to the offshore platform. The platform, named Wigot Alpha is standing 510m from the sea bed and 30 above the sea water level. The well, named Harthun is 255m below the sea surface.

The riser is vertical and 510m long, then we have a 100m horizontal top-side pipe. Both have a 4 inch (0.1m) diameter pipe having 0.0075m wall thickness. The pipeline connecting the well head to the riser base is composed by 4 pipes laying on the sea bed which is irregular and going deep up to 500m depth. This pipe is 0.12m diameter and has 0.009m wall thickness. All the pipes are assumed to be in steel.

The sea water temperature is assumed to be 6°C, the well pressure 100bar, the mass flow rate 15kg/s and the heat transfer coefficient 6.5 $W/m^2.°C$.

The situation is depicted in Figure 2.3 below and the sketch of the geometry editor in OLGA © is also given in Figure 2.4.

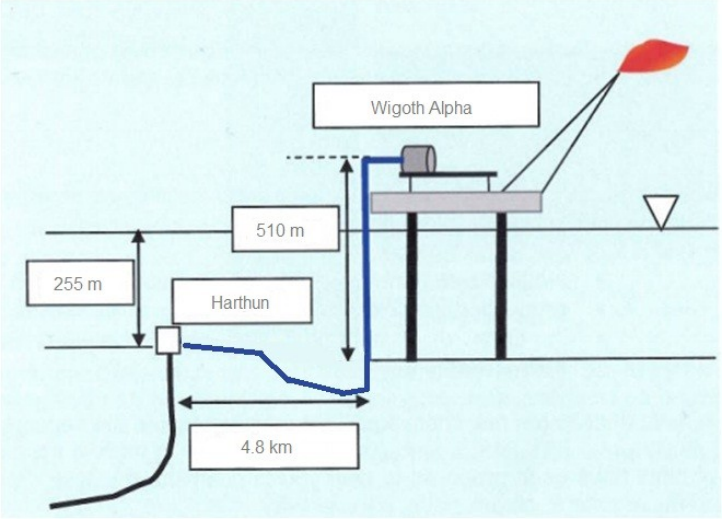


Figure 2.3. Pipeline and riser depiction

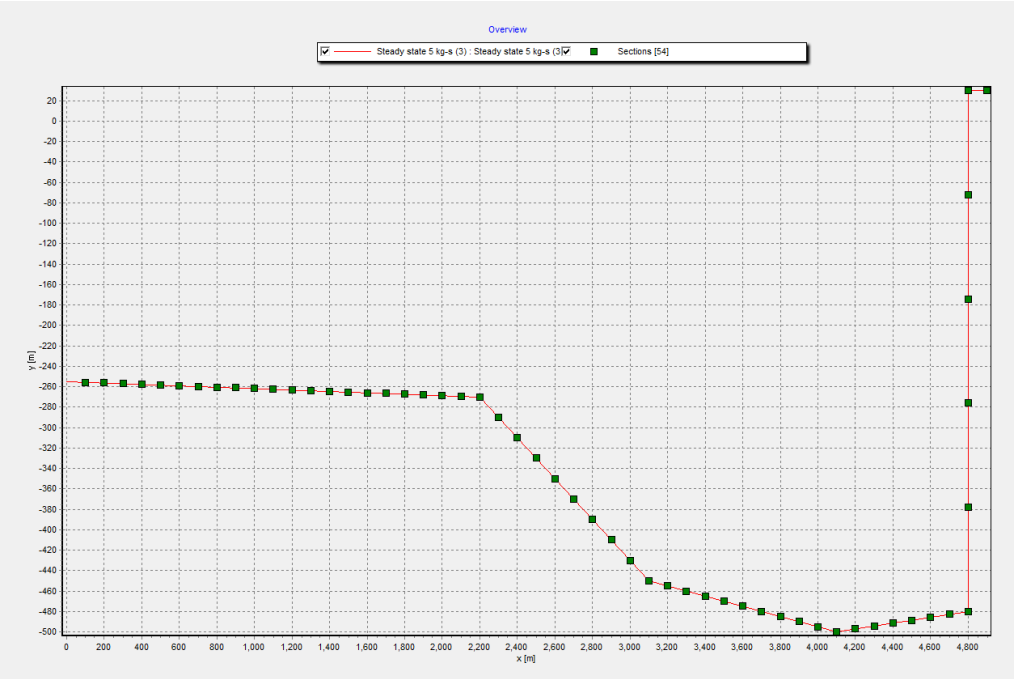


Figure 2.4. Pipeline and riser geometry

The fluid is assumed to be two-phase flow only (e.g. gas and oil) so nor water neither sand is assumed to flow together with it.

Normal steady-state operating conditions, for what concerns pressure and temperature, are reported in Figure 2.5 below.

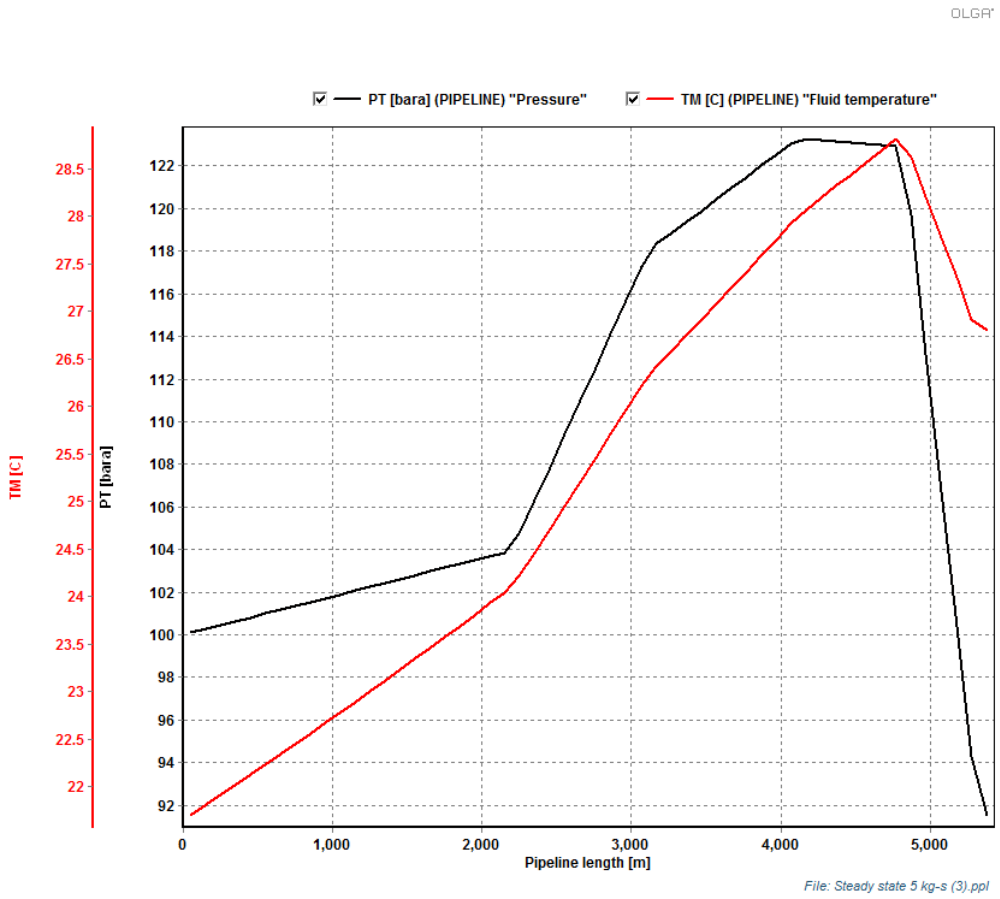


Figure 2.5. Pressure and temperature profile along the pipeline

2.6 **Corrosion modelling**

Literature has been explored looking for empirical correlations for pipeline corrosion. In particular the interest was in finding some correlations coming from appropriate data mining, which also gave distribution over time like the ones found in [24, 29] Few and incomplete data are available [30, 31] and the main focus is on CO_2 corrosion, which all reported to the same source correlation (e.g. eq. 2.12)

Since the aim is to carry on a simple analysis to show the applicability of the methodologies without any specific goal other than that, corrosion is assumed to act internally only and in particular caused by CO_2. This is reasonable because of absence of water/sand in the pipe and assumed external protective coating on the pipeline.

So, corrosion rate is modelled thanks to the well-known DeWaard and Milliams correlation [4, 5]

$$\log(v_c) = 5.8 - \frac{1710}{T} + 0.67 \cdot \log_{10}(n_{CO_2} \cdot p) \qquad (2.12)$$

Where $n_{CO_2}$is the CO_2 mole fraction, the corrosion rate is given in mm/year, the temperature in K and the pressure in bar.

Corrosion is the main factor affecting thickness deterioration hence the maximum allowable pressure in a pipe. Typically the burst pressure is given by the Barlow's formula, where we can see the direct proportionality between thickness and pressure; [5]

$$p_{all} = \frac{2 \cdot SYMS \cdot t}{D} F \tag{2.13}$$

Where F is a design factor 0.72, SMYS the specified maximum yield strength (i.e. the yield strength itself)

3 SURROGATE MODELLING

In the last years the need of very cumbersome system analysis and simulation led the academics world to focus on developing model approximation techniques basically building a function between interesting inputs and outputs from a model bridging the gap between them. After doing this, the function obtained is used as a black box with no care of what is happening in the system itself but just of the outputs coming from it. This bridge between input and output variables is called surrogate model. If the surrogate model is built properly it can permit us to have a lighter model to handle and to perform simulation analysis directly on it. As shown in [17] probabilistic analysis tools are more effective and less time consuming.

If the function is smooth and with no or few singularities this surrogate modelling process can be accomplished using simple techniques, as an k-dimensional quadratic polynomial where k is the number of design variables we are interested in. If the function to approximate is curvier other, more complex, methods can be applied so as to catch the sharpness of the function all over the design domain. One of these is Kriging and a combination of this method with Monte Carlo Simulation in order to carry out probabilistic analysis on the system can be found in [37].

The surrogate model which best fits the actual real function cannot be chosen *a priori* since the shape to imitate is unknown. In a trial and error process, tuning different methods and comparing their response on the whole design space we can state which tool best suits for our specific application.

Two methods are later applied in this work and hence a dissertation on these is given in this document. Further information can be found in [15] and in [22].

In particular, Stochastic Response Surface Method with quadratic polynomial and Kriging are considered. The first one present simple algorithm but is not accurate in presence of non linearities and to catch particular behaviour of the functions. This may lead to an erroneous probabilistic analysis of the system and incorrect estimation of low probabilities of failure; but this concept will be recalled and deepened later. The second one, theoretically much more complex and in the algorithmic implementation is shown to be more accurate and has already been introduced in probabilistic analysis and coupled with Monte Carlo Simulation.

3.1 **Stochastic Response Surface Method**

Assuming the shape of the real function can be imitated with a simple mathematical function a quadratic polynomial can be chosen and their response compared to the real one.

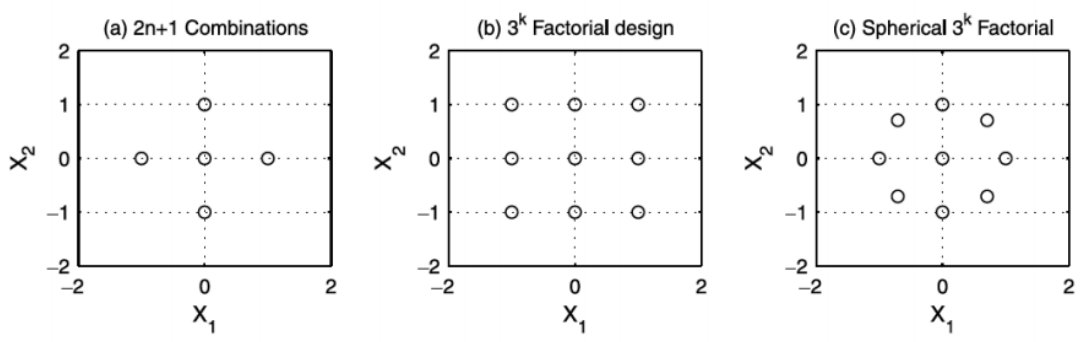


Figure 3.1. Sampling strategies [17]

Generally, the quadratic relation between inputs and the output can be written as

$$\hat{y} = a_0 + \sum b_i x_i + \sum c_i {x_i}^2 \tag{3.1}$$

where every design variable appears in the linear form and the quadratic one.

The coefficients a, b_i and c_i are unknown and we can determine their value solving a system of equations obtained sampling the real function. In order to sample evenly the whole design space different sampling strategies can be applied. In Figure 3-1 we can see a graphic representation for some of them.

Taking into account that what we need is a system of $2k + 1$ equations where k is the number of design variables, the first sampling strategy is the most suitable for our application. As said, it is important to spread evenly the samples across the design domain and this can be done considering a $\pm 3\sigma$ variation on each variable maintaining all the others unchanged on the mean value. After sampling our function what we obtain should be in the form;

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\alpha} \tag{3.2}$$

where $\boldsymbol{Y}$ and $\boldsymbol{\alpha}$ are the column vectors of output and coefficients respectively and $\boldsymbol{X}$ is the so-called design matrix, where by row the input variables appear first in linear and then in quadratic form.

Working out the expression given in 3-2 and multiplying by the transpose design matrix on both sides we obtain

$$\boldsymbol{X^T Y} = (\boldsymbol{X^T X})\boldsymbol{\alpha} \tag{3.3}$$

and then the solution derivation

$$\boldsymbol{\alpha} = (\boldsymbol{X^T X})^{-1} \boldsymbol{X^T Y} \tag{3.4}$$

In this way we can write the predicted value from our brand new surrogate as

$$\widehat{\boldsymbol{Y}} = \boldsymbol{X\alpha} + \boldsymbol{e} \tag{3.5}$$

$$y = a_0 + b \sum x_i + c \sum {x_i}^2 + \varepsilon \tag{3.6}$$

in vectorial form and in scalar form, respectively. The residuals are indicated with ε and the vector containing them with $\boldsymbol{e}$.

$$\boldsymbol{e} = \begin{Bmatrix} \varepsilon_1 \\ \dots \\ \varepsilon_n \end{Bmatrix} \tag{3.7}$$

where $n=2k+1$ is the number of samples taken.

An important remark has to be done concerning close to singular matrices. If our design matrix X presents such a property, implementing the solution of the system of equation 3.2 may represent a challenge. The inversion results heavily affected by the singularity of the matrix hence if a warning is returned by Matlab probably the matrix is close to singular and function *pinv.m* can be employed to overcome this.

3.2 **Kriging**

3.2.1 Introduction to kriging

Kriging is an approximation method developed by the mining engineer Danie Kriege in the 1950s to predict the concentration of mineral in the ground having just some distributed samples and draw conclusions about the suitability of this ground to be mined or not.

Assuming to have an array of n sample data such that

$$\boldsymbol{X} = \{\boldsymbol{x}^{(1)}, \dots, \boldsymbol{x}^{(l)}, \dots, \boldsymbol{x}^{(i)}, \dots, \boldsymbol{x}^{(n)}\}^T \tag{3.8}$$

Where each $\boldsymbol{x}$ is a vector of k design variables which represents a set of input data for the problem we can assume to have an output variable

$$\boldsymbol{y} = \{y^1, \dots, y^l, \dots, y^i, \dots, y^n\}^T \tag{3.9}$$

As the result from a random variable $\boldsymbol{Y}$**.**

$$\boldsymbol{Y} = \{Y(\boldsymbol{x}^{(1)}), \dots, Y(\boldsymbol{x}^{(l)}), \dots, Y(\boldsymbol{x}^{(i)}), \dots, Y(\boldsymbol{x}^{(n)})\}^T \tag{3.10}$$

This may be seen as a vector of combinations of stochastic design variables, which have the same mean and same standard deviation.

3.2.2 Kriging predictor

If we would find a value of the same function in another point of the space a sort of common behaviour between the known points should be found. Assuming noise-free data, this may be seen as a prediction of the output and can be written as

$$\hat{\boldsymbol{y}}(\boldsymbol{x}) = \boldsymbol{b}^{\boldsymbol{T}}\boldsymbol{\psi} = \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{b_i}\boldsymbol{\varphi}\left(\left|\boldsymbol{x} - \boldsymbol{c}^{(i)}\right|\right) \tag{3.11}$$

which is linear combination of radial basis functions evaluated in the distance between the predicted point and the centre point *c* of the i-th sample. Another way of representing the same, shifting all the centres on sample points, is

$$\hat{\boldsymbol{y}}(\boldsymbol{x}) = \boldsymbol{a} + \sum_{\boldsymbol{i=1}}^{\boldsymbol{n}} \boldsymbol{b_i} \cdot \boldsymbol{\varphi}\left(\left|\boldsymbol{x} - \boldsymbol{x}^{(\boldsymbol{i})}\right|\right) \tag{3.12}$$

which is a linear combination of radial basis function, expressed in a general form and centred on a sample point, plus a constant term. So the prediction happens always starting from a value a which will be constant and moving with some deviations from there. Intuitively, this value can be thought to be somewhere in the middle of the sampled points and will be derived later on.

The local deviation at the prediction point is formulated using stochastic processes. The sample points are fitted using the Gaussian random function as correlation function which is slightly modified for this particular purpose of the Kriging predictor

$$\psi^{(i)} = e^{-\left(\sum_{j=1}^{k} \theta_j |x_j{}^{(i)} - x_j|^{p_j}\right)} \tag{3.13}$$

This basis function is clearly similar to the Gaussian basis function and differs from it just for p_j and θ_j which can control the shape of the basis function in each dimension of the design space. They account for a different impact of the design variables on the predicted output.

The common behaviour among the sample sets affects the prediction and this property of the predictor can be expressed by the correlation of random variables.

3.2.3 Kriging correlation

From what discussed above random variables are assumed to be correlated in the following form

$$corr[Y(\boldsymbol{x}^{(\boldsymbol{i})}), Y(\boldsymbol{x}^{(\boldsymbol{l})})] = e^{-\left(\sum_{j=1}^{k} \theta_j |x_j{}^{(i)} - x_j{}^{(l)}|^{p_j}\right)} \tag{3.14}$$

hence, for the assumptions done it is possible to state the same for a generic sample and the predicted value.

$$corr\left[Y\left(\boldsymbol{x}^{(\boldsymbol{i})}\right), Y\left(\boldsymbol{x}^{(*)}\right)\right] = e^{-\left(\sum_{j=1}^{k} \theta_j |x_j{}^{(i)} - x_j{}^{(*)}|^{p_j}\right)} \tag{3.15}$$

As we can see the correlation depends mainly on three parameters.

First, the Euclidean distance between the points. If it goes to zero (points close to each other) the correlation rends to unity, if it goes to infinity correlation tends to zero.

Then, the smoothness coefficient p_j expresses the quickness of the function. If it is small, the correlation goes suddenly to one as soon as the distance goes to zero and goes slowly to zero if the distance tends to infinity. If the smoothness is high, the correlation goes smoothly to one when the distance goes to zero and drops to zero when the distance increases.

Eventually, the "width parameter" or "activity parameter" θ_j describes how wide the basis function is around the training data. In other words, it expresses how large the sphere of influence is for the data points that are evaluated. The more the j-th variable affects the correlation the higher this value and the faster the correlation drops as you move in the j-th direction. Summarising:

- having a small θ_j the output dependent on x_j approximates to a flat line or in other words the j-th variable is not so active on the response
- having a big θ_j, the j-th variable will have a high influence on the response and this will be changing a lot and rapidly depending on this variable.

The distance between the prediction and the sample will matter just if the correlation coefficient is high; in the other case it will have a little influence anyway.

If the activity parameter for a certain variable is low, in order to obtain a better prediction, the corresponding variable can be removed from the analysis and assume it's not influencing the output at all. Reducing the dimension of the design space kriging can achieve higher accuracies.

We can gather all the correlations in a correlation matrix

$$\boldsymbol{\Psi} = \begin{bmatrix} corr[Y(\boldsymbol{x^{(1)}}), Y(\boldsymbol{x^{(1)}})] & \dots & corr[Y(\boldsymbol{x^{(1)}}), Y(\boldsymbol{x^{(n)}})] \\ \dots & \dots & \dots \\ corr[Y(\boldsymbol{x^{(n)}}), Y(\boldsymbol{x^{(1)}})] & \dots & corr[Y(\boldsymbol{x^{(n)}}), Y(\boldsymbol{x^{(n)}})] \end{bmatrix} \tag{3.16}$$

which, by definition of correlation, will have ones on the diagonal and is symmetric because $corr[Y(\boldsymbol{x^{(i)}}), Y(\boldsymbol{x^{(l)}})] = corr[Y(\boldsymbol{x^{(l)}}), Y(\boldsymbol{x^{(i)}})]$. Since all $Y(\boldsymbol{x^{(i)}})$ have the same mean and same standard deviation, according to the definition of correlation we can write

$$cov[\boldsymbol{Y}] = \sigma^2 \boldsymbol{\Psi} \tag{3.17}$$

This characterizes the way $\boldsymbol{Y}$ varies moving in different coordinate directions and is depending on parameters $\sigma^2, \mu, \theta_j \; and \; p_j$ which are unknown for our problem and have

to be estimated. A way to do it is to maximize the likelihood of the experimental data. If we consider only a random realization, since it is assumed to be normally distributed with mean μ and standard deviation σ^2 we can write its likelihood as

$$L(Y^i) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left[\frac{Y^i-\mu}{\sigma}\right]^2} \tag{3.18}$$

If we want to express the likelihood of all our sample sets we can just combine the single likelihood expressions

$$L\left(Y^{\mathbf{1}}, \dots, Y^{\boldsymbol{n}}\right)|\mu, \sigma) = \sum_{i=1}^{n} L(Y^i) = \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}}} e^{\left[-\Sigma\frac{(Y_i-\mu)^2}{2\sigma^2}\right]} \tag{3.19}$$

which, following the notation proposed, can be rewritten as

$$L = \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}}|\boldsymbol{\Psi}|} e^{-\left[\frac{(\boldsymbol{y}-\mathbf{1}\mu)^{\boldsymbol{T}}\boldsymbol{\Psi}^{-\mathbf{1}}(\boldsymbol{y}-\mathbf{1}\mu)}{2\sigma^2}\right]} \tag{3.20}$$

In order to maximize it, a simplification turning it in logarithmic scale is found to be useful and effective. An expression for the likelihood named log-likelihood is found

$$\ln(L) = -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln(\sigma^2) - \frac{1}{2}\ln|\boldsymbol{\Psi}| - \left[\frac{(\boldsymbol{y}-\mathbf{1}\mu)^{\boldsymbol{T}}\boldsymbol{\Psi}^{-\mathbf{1}}(\boldsymbol{y}-\mathbf{1}\mu)}{2\sigma^2}\right] \tag{3.21}$$

Differentiating this function with respect to σ^2, μ and setting the derivatives to zero means that we want to find the values for σ^2, μ, indicated as $\hat{\sigma}^2, \hat{\mu}$, which maximize the likelihood of our sample. Looking for a maximum value for this expression is interpreted as the will of having the function which is most compatible with our observed data.

$$\frac{\partial\ln(L)}{\partial\mu} = 0 \quad \rightarrow \hat{\mu} \tag{3.22}$$

$$\frac{\partial\ln(L)}{\partial\sigma^2} = 0 \quad \rightarrow \hat{\sigma}^2 \tag{3.23}$$

which result, after some basic algebraic manipulation in an estimate at maximum likelihood (indicated from now on by a circumflex over the quantity)

$$\hat{\mu} = \frac{\mathbf{1}^T \boldsymbol{\Psi}^{-1} \boldsymbol{y}}{\mathbf{1}^T \boldsymbol{\Psi}^{-1} \mathbf{1}} \tag{3.24}$$

$$\hat{\sigma}^2 = \frac{(\boldsymbol{y} - \mathbf{1}\mu)^T \boldsymbol{\Psi}^{-1} (\boldsymbol{y} - \mathbf{1}\mu)}{n} \tag{3.25}$$

Now, substituting in equation 3-21 and neglecting the constant terms an expression for the concentrated log-likelihood function is found and given in equation 3-26. We may notice that it's a function of $\boldsymbol{\theta}, \boldsymbol{p}$ only and we should find values for these which maximize it as well

$$\ln(L) \sim -\frac{n}{2}\ln(\sigma^2) - \frac{1}{2}\ln|\boldsymbol{\Psi}| = f(\boldsymbol{\theta}, \boldsymbol{p}) \tag{3.26}$$

since this expression is not differentiable with respect to $\boldsymbol{\theta}, \boldsymbol{p}$ a numerical optimization technique to find the function directly is then used. Applicable techniques may be genetic algorithm or simulated annealing.

The parameter p_j can assume many different values from 0 to 2 or even higher depending on the smoothness of the problem to solve and singularities. In current practice, for most engineering applications, p_j is chosen constant equal to 2 and a maximization considering only $\boldsymbol{\theta}$ is carried out.

Particular care has to be taken when computing the log-likelihood because an iterative evaluation procedure is followed and it may turn out to be too much computationally expensive. In particular, inverting matrix $\boldsymbol{\Psi}$ can be so and a Cholesky factorization followed by backwards and forward substitution is suggested.

3.2.4 Kriging prediction

If we call the predicted value by the model at the new sampling point $\hat{y}$, we can expand the vector of the responses as

$$\tilde{\boldsymbol{y}} = \begin{Bmatrix} \boldsymbol{y} \\ \hat{y} \end{Bmatrix} \tag{3.27}$$

and we can write and nx1 vector of correlations and rewrite the correlation matrix as

$$\boldsymbol{\psi} = \begin{Bmatrix} corr\left[Y\left(\boldsymbol{x}^{(1)}\right), Y(\boldsymbol{x})\right] \\ \dots \\ corr\left[Y\left(\boldsymbol{x}^{(n)}\right), Y(\boldsymbol{x})\right] \end{Bmatrix} = \begin{Bmatrix} \psi^1 \\ \dots \\ \psi^n \end{Bmatrix} \tag{3.28}$$

$$\widetilde{\boldsymbol{\Psi}} = \begin{bmatrix} \boldsymbol{\Psi} & \boldsymbol{\psi} \\ \boldsymbol{\psi^T} & 1 \end{bmatrix} \tag{3.29}$$

If we apply the same procedure seen in the previous chapter, we obtain an augmented log-likelihood function which has to be maximized as well.

$$\text{a ln}(L) = -\frac{n}{2}\ln(2\pi) - \frac{n}{2}\ln\left(\widehat{\sigma^2}\right) - \frac{1}{2}\ln\,|\widetilde{\boldsymbol{\Psi}}| - \left[\frac{(\tilde{\boldsymbol{y}} - \mathbf{1}\hat{\mu})^{\boldsymbol{T}}\widetilde{\boldsymbol{\Psi}}^{-\mathbf{1}}(\tilde{\boldsymbol{y}} - \mathbf{1}\hat{\mu})}{2\hat{\sigma}^2}\right] \tag{3.30}$$

The latter expression can be approximated and rewritten as

$$\text{a ln}(L) \sim -\frac{\begin{pmatrix} \boldsymbol{y} - \mathbf{1}\hat{\mu} \\ \hat{y} - \hat{\mu} \end{pmatrix}^{\boldsymbol{T}} \begin{pmatrix} \boldsymbol{\Psi} & \boldsymbol{\psi} \\ \boldsymbol{\psi^T} & 1 \end{pmatrix}^{-\mathbf{1}} \begin{pmatrix} \boldsymbol{y} - \mathbf{1}\hat{\mu} \\ \hat{y} - \hat{\mu} \end{pmatrix}}{2\hat{\sigma}^2} \tag{3.31}$$

Inverting matrix $\widetilde{\boldsymbol{\Psi}}$ can be cumbersome and time consuming as well and since it can be written in the form expressed in 1-24 the procedure derived in [55] may be applied. (See Appendix on Matrix Algebra for further details)

$$\widetilde{\boldsymbol{\Psi}}^{-\mathbf{1}} = \begin{bmatrix} \boldsymbol{\Psi}^{-\mathbf{1}} + \boldsymbol{\Psi}^{-\mathbf{1}}\boldsymbol{\psi}(1 - \boldsymbol{\psi}^{\boldsymbol{T}}\boldsymbol{\Psi}^{-\mathbf{1}}\boldsymbol{\psi})^{-\mathbf{1}}\boldsymbol{\psi}^{\boldsymbol{T}}\boldsymbol{\Psi}^{-\mathbf{1}} & -\boldsymbol{\Psi}^{-\mathbf{1}}\boldsymbol{\psi}(1 - \boldsymbol{\psi}^{\boldsymbol{T}}\boldsymbol{\Psi}^{-\mathbf{1}}\boldsymbol{\psi})^{-\mathbf{1}} \\ -(1 - \boldsymbol{\psi}^{\boldsymbol{T}}\boldsymbol{\Psi}^{-\mathbf{1}}\boldsymbol{\psi})^{-\mathbf{1}}\boldsymbol{\psi}^{\boldsymbol{T}}\boldsymbol{\Psi}^{-\mathbf{1}} & (1 - \boldsymbol{\psi}^{\boldsymbol{T}}\boldsymbol{\Psi}^{-\mathbf{1}}\boldsymbol{\psi})^{-\mathbf{1}} \end{bmatrix} \tag{3.32}$$

Manipulating equation 1-23 we can write and differentiate with respect to $\hat{y}$

$$\begin{aligned} \text{a ln}(L) \sim & \left(\frac{-1}{2\hat{\sigma}^2(1 - \boldsymbol{\psi}^{\boldsymbol{T}}\boldsymbol{\Psi}^{-\mathbf{1}}\boldsymbol{\psi})}\right)(\hat{y} - \hat{\mu})^2 \\ & + \left(\frac{\boldsymbol{\psi}^{\boldsymbol{T}}\boldsymbol{\Psi}^{-\mathbf{1}}(\boldsymbol{y} - \mathbf{1}\hat{\mu})}{\hat{\sigma}^2(1 - \boldsymbol{\psi}^{\boldsymbol{T}}\boldsymbol{\Psi}^{-\mathbf{1}}\boldsymbol{\psi})}\right)(\hat{y} - \hat{\mu}) \end{aligned} \tag{3.33}$$

$$\frac{\partial \text{a ln}(L)}{\partial \hat{y}} = 0 \quad \rightarrow \hat{y} \tag{3.34}$$

$$\left(\frac{-1}{\hat{\sigma}^2(1-\boldsymbol{\psi}^T\boldsymbol{\Psi}^{-1}\boldsymbol{\psi})}\right)(\hat{y}-\hat{\mu})+\left(\frac{\boldsymbol{\psi}^T\boldsymbol{\Psi}^{-1}(\boldsymbol{y}-\mathbf{1}\hat{\mu})}{\hat{\sigma}^2(1-\boldsymbol{\psi}^T\boldsymbol{\Psi}^{-1}\boldsymbol{\psi})}\right)=0 \tag{3.35}$$

and solving

$$y^*(\boldsymbol{x})=\hat{\mu}+\boldsymbol{\psi}^T\boldsymbol{\Psi}^{-1}(\boldsymbol{y}-\mathbf{1}\hat{\mu}) \tag{3.36}$$

which is the expression of the Kriging predictor and can be compared to the general expression given in 1-5. We can see the constant term a assuming the form of $\hat{\mu}$ and the sum expressed in matrix form. Here we can distinguish $\boldsymbol{\psi}^T$ as b_i and $\boldsymbol{\Psi}^{-1}(\boldsymbol{y}-\mathbf{1}\hat{\mu})$ as $\varphi\left(\boldsymbol{x}-\boldsymbol{x}^{(i)}\right)$.

3.2.5 Kriging errors and remarks

There are different behaviours for which the augmented log-likelihood function can depend on the predicted value. It can happen that, moving away from the optimum value of the prediction $\hat{y}$ (i.e. the best guess possible) the function drops or remains almost flat. In the first case, it means that if the predicted value is not the best one the consistency of the prediction with the observed data is low. In the second case, many different values of $\hat{y}$ around the best one give the same performance hence there is no discrimination between them. In the latter situation is less the confidence we put in the model and a greater potential error may occur.
A measure of this potential error is the curvature of the augmented log-likelihood function: the higher the lower the potential error, and vice versa. The curvature can be expressed taking the second derivative of the augmented log-likelihood function with respect to $\hat{y}$.

$$\frac{\partial^2 a\ln(L)}{\partial\hat{y}^2}=\frac{1}{\hat{\sigma}^2(1-\boldsymbol{\psi}^T\boldsymbol{\Psi}^{-1}\boldsymbol{\psi})} \tag{3.37}$$

hence from considerations done above;

$$potenential\ error=\hat{\sigma}^2\left(1-\boldsymbol{\psi}^T\boldsymbol{\Psi}^{-1}\boldsymbol{\psi}\right) \tag{3.38}$$

The strength of kriging is in the fact that it doesn't use Euclidean distances but

$$distance = \sum_{j=1}^{k} \theta_j \left| x_j^{(i)} - x_j \right|^{p_j} \tag{3.39}$$

which is weighted differently on each variable axis. The tuning of the function, maximizing the likelihood, permits us to achieve optimum values for θ_j, p_j and a better interpretation of each variable contribution. Moreover scaling of variable is not necessary as in other fitting methods because the difference is absorbed by the weight parameter. It's also worth saying that a proper scaling on design variables allows us to obtain θ_j comparable from problem to problem. One more wariness to get better results is to look for $\theta_j's$ on a logarithmic scale, since we can experimentally spot out that there is more change in the correlations with low weight parameter values than with high ones.

3.2.6 Regression kriging for noise removal

In case of a high number of variables and a lot of sample responses it may happen that noise occurs. Even if the trend is clear because of the pass-through all points condition kriging struggles in being accurate [15]. This scattering is well represented in Figure 3-2.

To filter this scattering in the data series a well known technique may be employed [15]: adding a so-called regression constant to the main diagonal of the correlation matrix

$$\boldsymbol{\Psi_r} = \boldsymbol{\Psi} + \boldsymbol{\lambda I} \tag{3.40}$$

Of course modifying this definition also the other variables calculated in the Kriging prediction procedure will have another formulation. In particular

$$y_r^*(\boldsymbol{x}) = \hat{\mu}_r + \boldsymbol{\psi^T}(\boldsymbol{\Psi} + \boldsymbol{\lambda I})^{-1}(\boldsymbol{y} - \mathbf{1}\hat{\mu}_r) \tag{3.41}$$

$$\hat{\mu}_r = \frac{\mathbf{1^T}(\boldsymbol{\Psi} + \boldsymbol{\lambda I})^{-1}\boldsymbol{y}}{\mathbf{1^T}(\boldsymbol{\Psi} + \boldsymbol{\lambda I})^{-1}\mathbf{1}} \tag{3.42}$$

$$\hat{\sigma}_r^{\,2} = \frac{(\boldsymbol{y} - \mathbf{1}\hat{\mu}_r)^{\boldsymbol{T}}(\boldsymbol{\Psi} + \boldsymbol{\lambda I})^{-1}(\boldsymbol{y} - \mathbf{1}\hat{\mu}_r)}{n} \tag{3.43}$$

The value of the regression constant is not known *a priori* and is searched with the MLE procedure, so directly searching a maximum for the likelihood function.

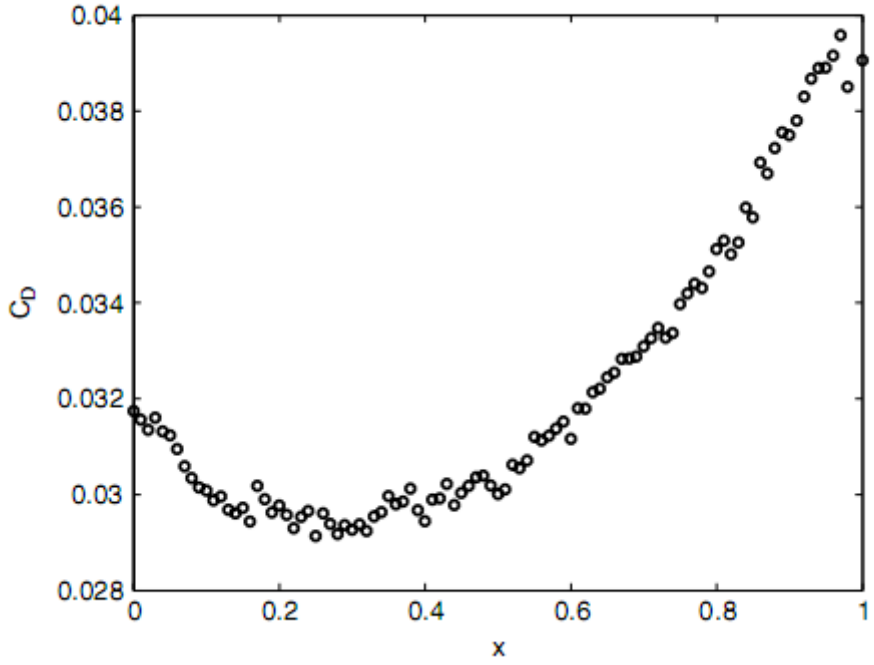


Figure 3.2. Noise sample responses for variable C_D [15]

3.3 **SRSM vs Kriging comparison**

From the overview of the two methods given in this chapter we can see at a first sight that the algorithmic complexity is a determining factor in Kriging. But it is actually this aspect that may give us benefits using it.

3.3.1 General qualitative comparison considerations

Kriging, as shown in the previous section can get through all the sampling points and if there are some on peaks it can get those points. SRSM with quadratic polynomial cannot do so, even more so if some samples are on a peak. If the sampling is done properly (evenly distributed and reaching the peak points) Kriging shows far better accuracy compared to SRSM. Some comparative tests have been done to show this concept and results are reported in the following chapter.

If we are dealing with noisy data, Kriging can cope with this and noise removal techniques are effective [15] while we cannot remove noise with SRSM surrogate modelling.

Another important aspect is that using the quadratic polynomial we assume *a priori* a shape for the function we are trying to approximate, which is a k-dimensional quadratic polynomial surface. Kriging doesn't do that and leaves open the possibility to build whatever surface based on the samples we have.

In perspective of a Reliability Analysis to carry out on the surface we are building, the complexity of the expression which can result from Kriging is to be remarked as it may increase the computational effort for its evaluation and manipulation. Mainly the complexity increases with the square of the number of sample points since they explicitly appear in the Kriging prediction expression.

3.3.2 'Sombrero' test

Considering a 3D space where two variables act as inputs and the third one represents the output we can easily visualise and make some consideration on the effectiveness of the two surrogate modelling methods above mentioned.

Considering the situation where we have a model function similar to the one depicted in Figure 3.3 it is certainly difficult to represent it with a simplified function.

If we try to build a surrogate of this simple three-dimensional function using a SRSM quadratic polynomial technique what we obtain is in Figure 3.4, which is largely far from what we expect to reconstruct despite from the high number of reconstruction points used.

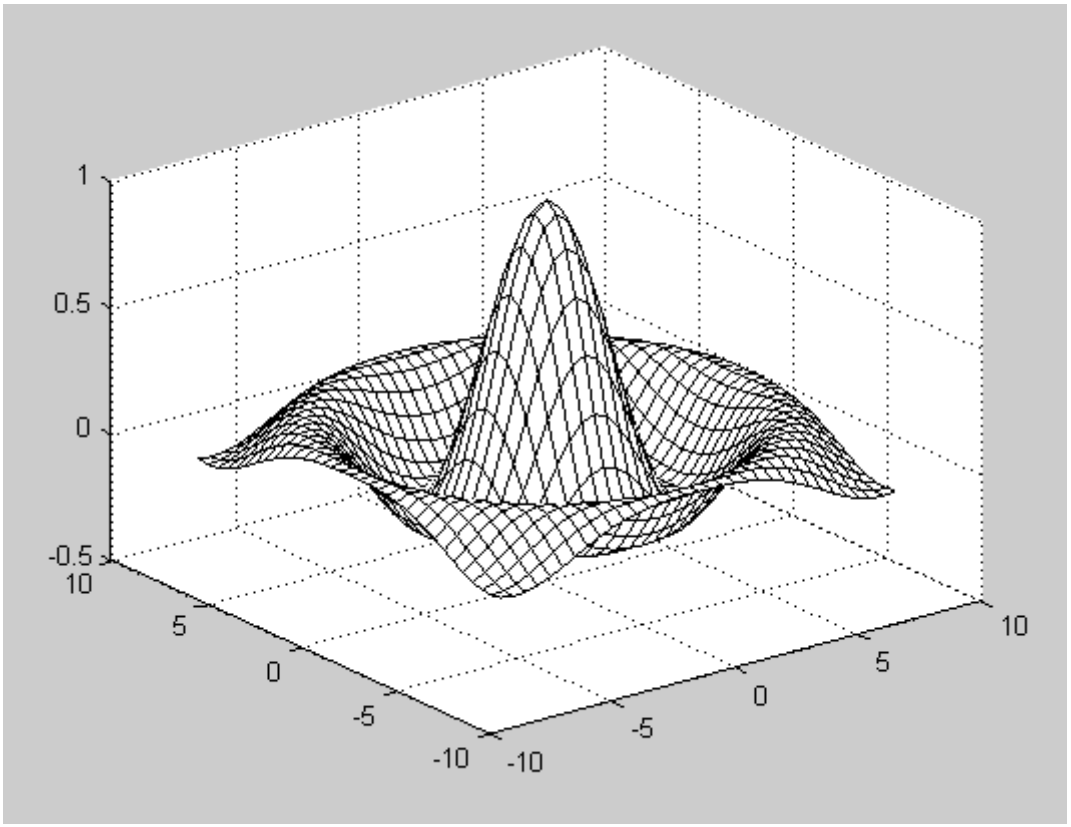


Figure 3.3. 'Sombrero' function

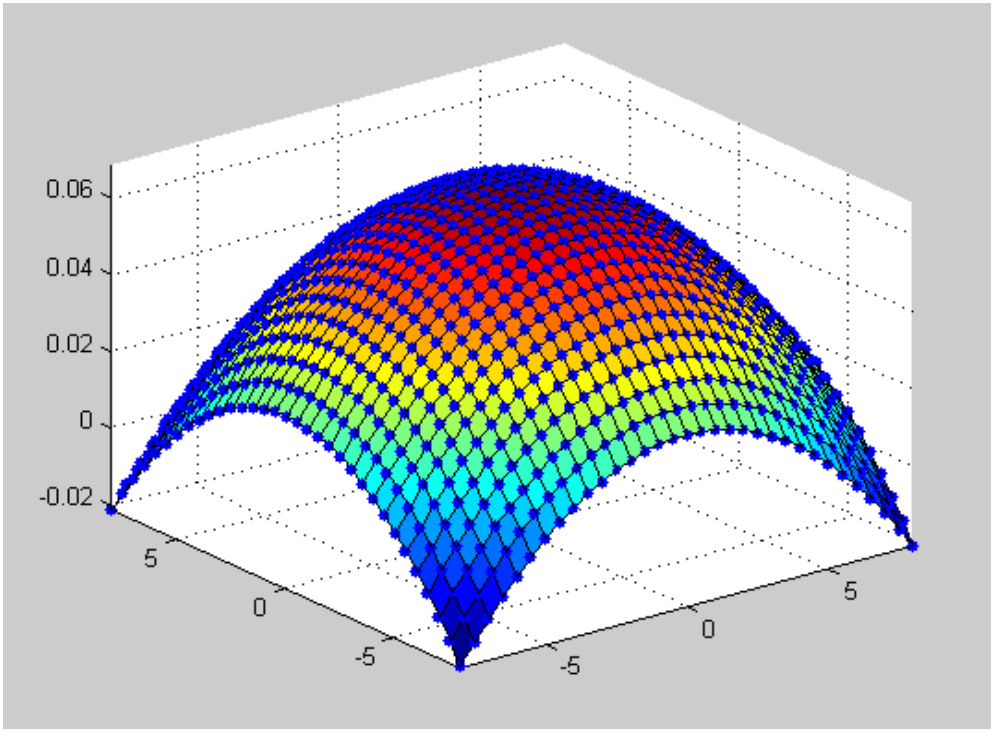


Figure 3.4. SRSM quadratic polynomial 'Sombrero' surrogate model

From what we can see the peak is highly underestimated and an almost flat curve is obtained. We could do the same using a fourth order polynomial expecting a better result, which is showed in Figure 3.5.

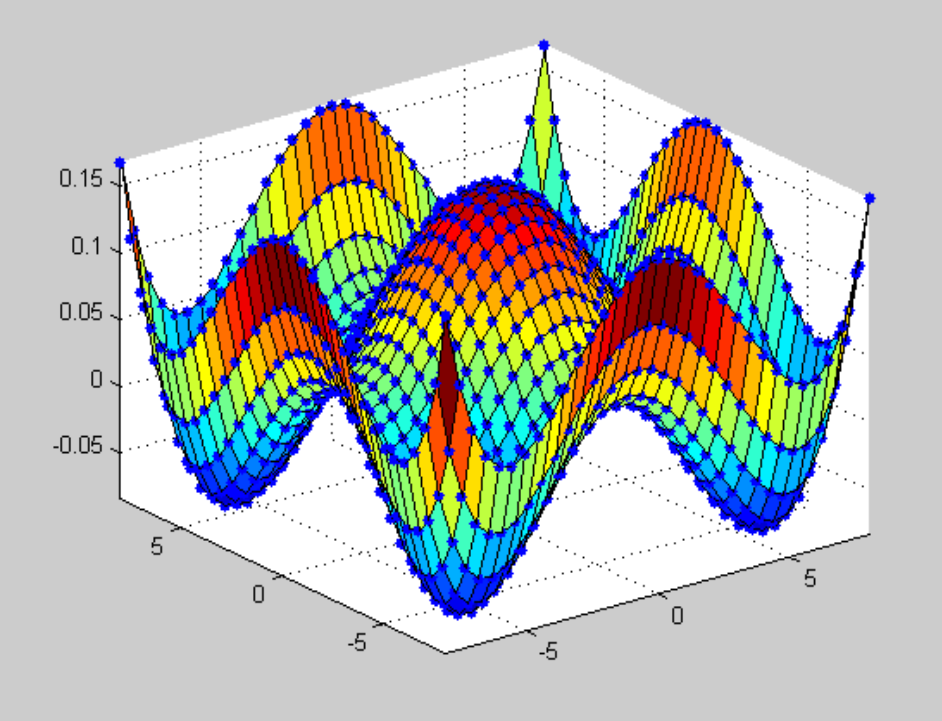


Figure 3.5. SRSM fourth order polynomial 'Sombrero' surrogate model

What we obtain is something slightly better but still largely wrong. Probably increasing the order of the polynomial better results could be obtained but because of the sudden changes in curvature of the original function the results obtained wouldn't always be satisfactory.

If we try the same test on Kriging, which by definition passes through all the samples points what we should obtain is something more similar to the original function. This is obtained only if the sampling is appropriate and results in sample points on the peak. What we obtain is showed in Figure 3.6 (few samples and few prediction points) and Figure 3.7 (lots of samples and lots of prediction points).

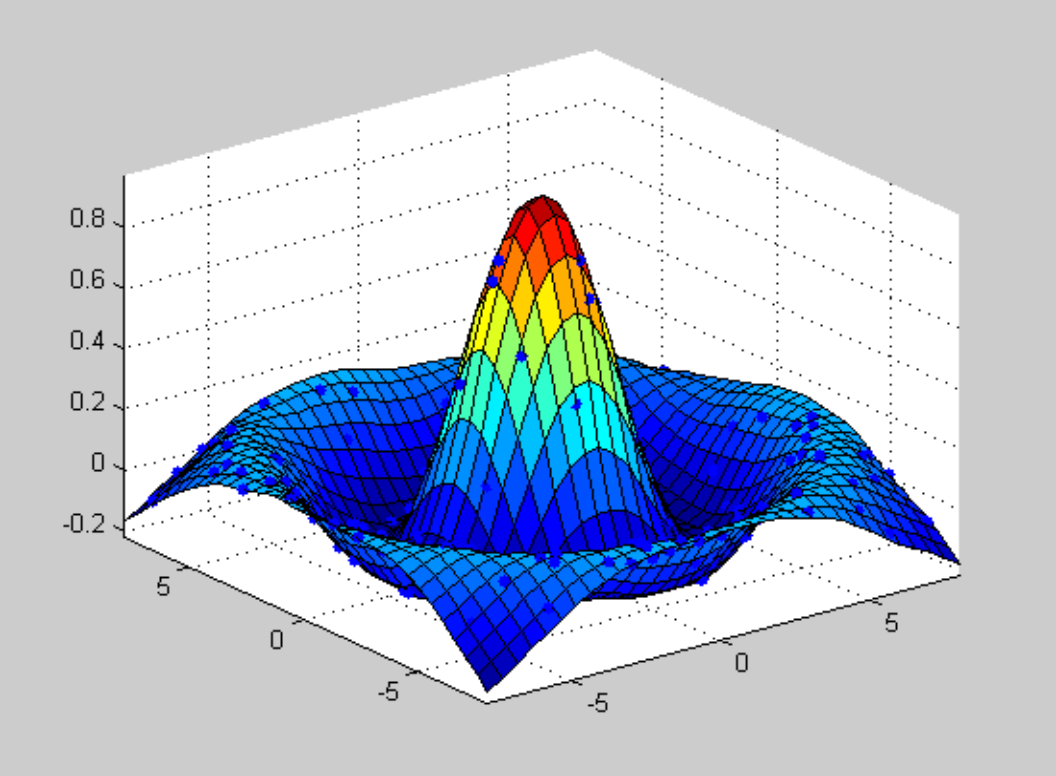


Figure 3.6. Kriging 'Sombrero' surrogate model, few points

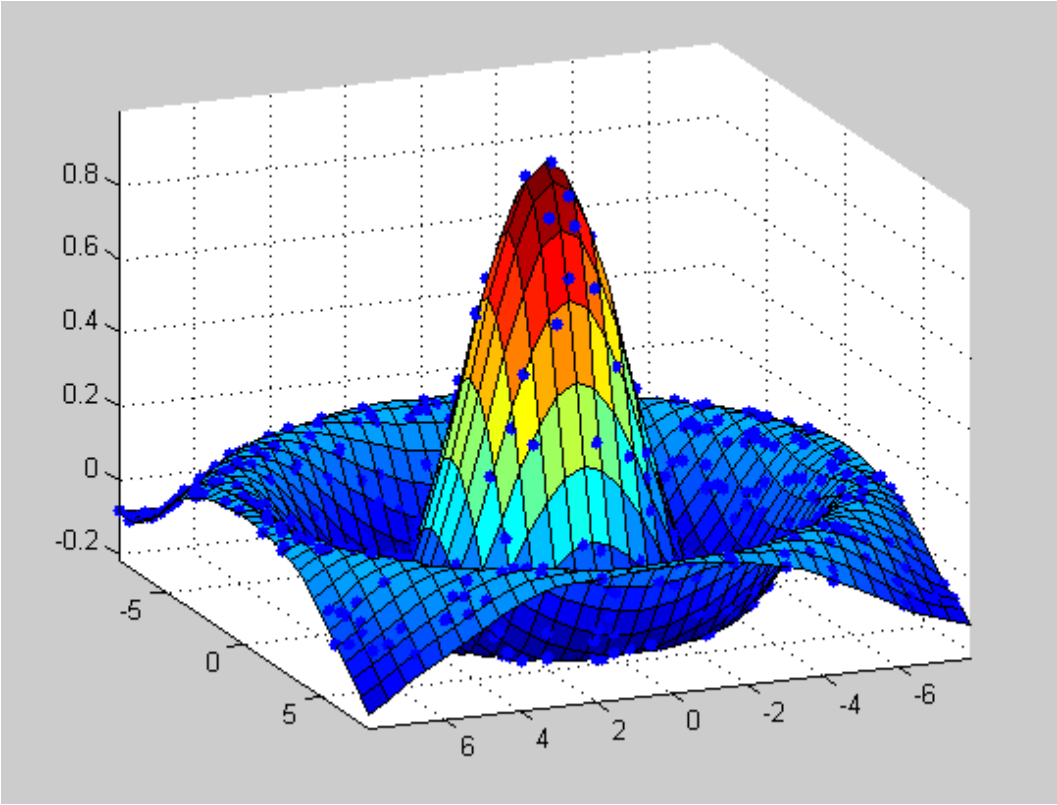


Figure 3.7. Kriging 'Sombrero' surrogate model, many points

As said the quadratic polynomial cannot cope with sudden changes in curvature and this is proved by the obtained results. Theoretically we could approximate the function using

the quadratic polynomial on the peak only, selecting samples just from the very tip of it. What we obtained is shown in Figure 3.8 and Figure 3.9 with two different sampling domains e.g. x_1 and x_2 belonging to intervals [-2, 2] and [-3, 3] respectively.

As we can see, even if the second considered domain is just slightly bigger than the first one the quadratic polynomial approximation cannot do the job because there is a small sign of curvature change.

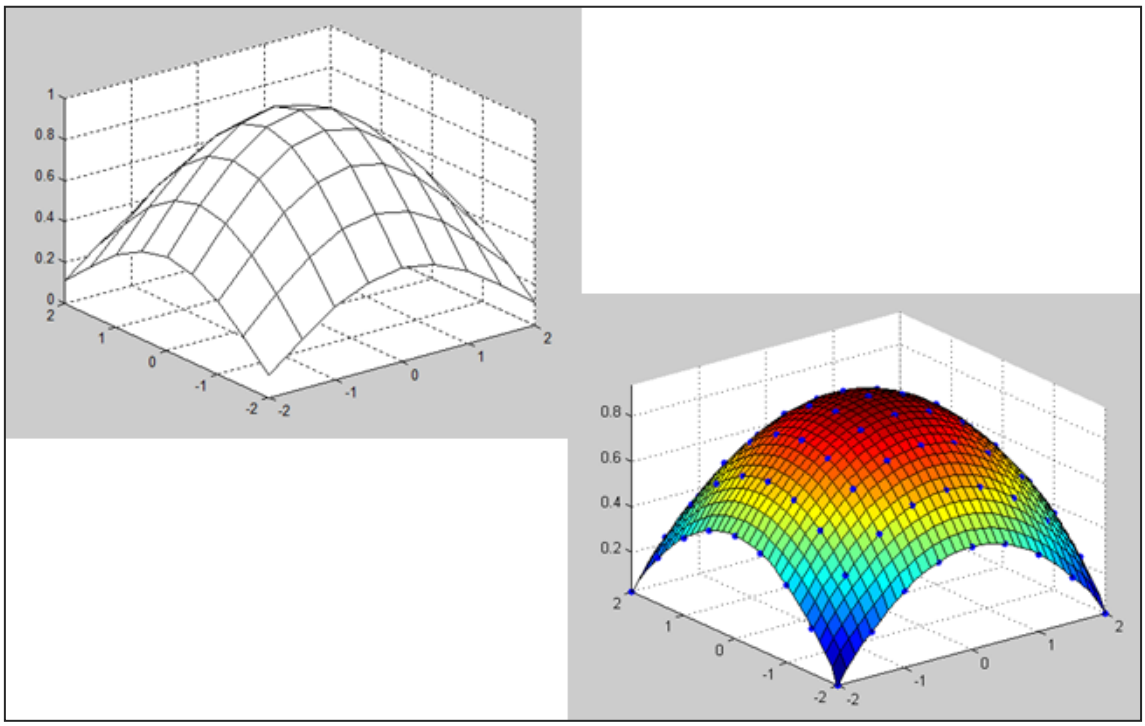


Figure 3.8. Local SRSM quadratic polynomial in the sampling domain [-2, 2]

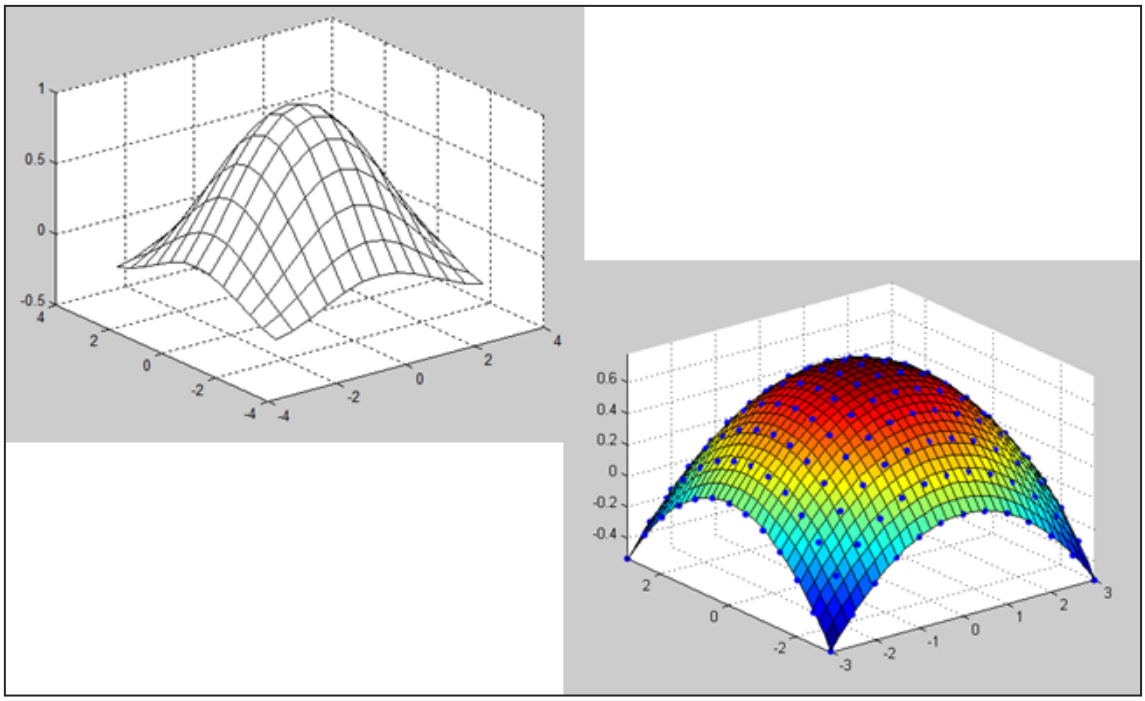


Figure 3.9. Local SRSM quadratic polynomial in the sampling domain [-3, 3]

3.4 **Techniques for Kriging accuracy**

Since the quantity and quality of samples in Kriging is so important, particular attention has been paid.

High quality sampling can be guaranteed through Latin Hypercube Sampling [9].

The minimum quantity of samples needed to obtain a good surface cannot be determined analytically and a trial-and-error procedure is followed. A particular algorithm for checking this has been developed. The considerations behind it are reported in 3.4.2 and the code used in Appendix.

3.4.1 Latin Hypercube Sampling

LHS is stratified sampling technique for which each variable in a design space is evenly sampled in a range. Knowing the range for the variable to change this can be divided in several non-overlapping 'bins' and a random sample in each bin can be taken. Using this particular sampling procedure we ensure that the variable is fully represented and, since the intervals have all the same size (e.g. the bins seeds are equally spaced), no local accumulation of samples occurs. [9].

This is performed for all the variables and then in order to obtain coordinates for points where to sample our function in our design space, sampling values for different variables are paired. This matching procedure is showed in Figure 3.10. Once these sets are obtained sample responses are generated.

To ensure this variables pairing is the best possible to fully represent the design space many techniques and codes are available in literature [15].

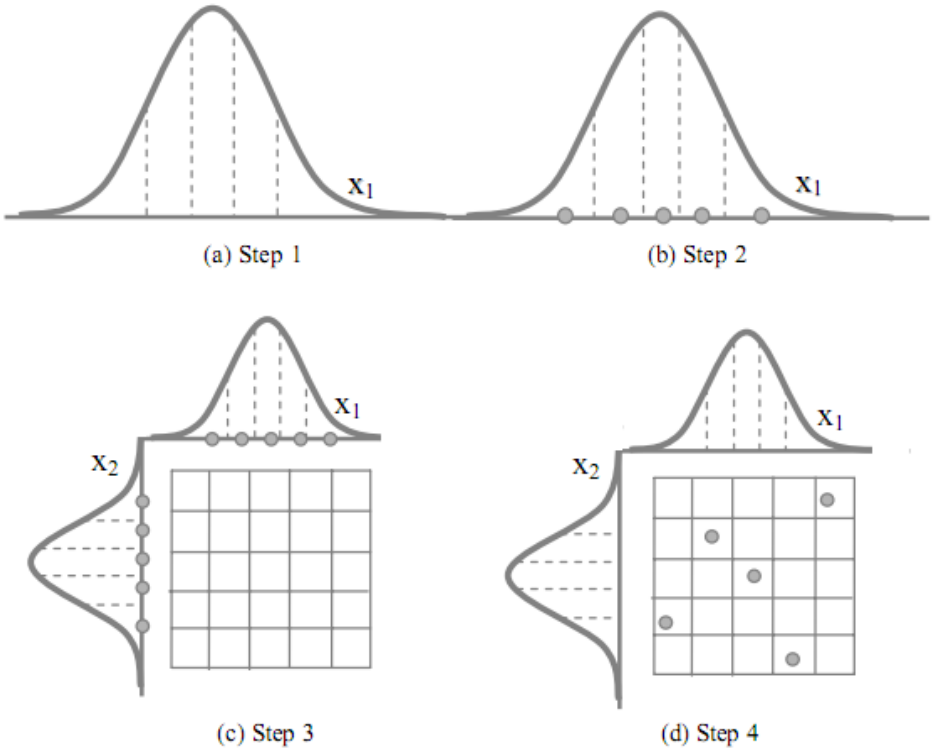


Figure 3.10. Latin Hypercube Sampling procedure

Usually samples generation routines create sets of *k* variables assuming they are ranging from 0 to 1. Then scaling of the same on the actual range is done.

Assuming the variables normally distributed we define a so-called cumulative probability for the i-th sample of the actual variable *x* we are considering.

$$P_i = \frac{x_i}{n} + \frac{m-1}{n} \tag{3.44}$$

where *n* is the size of the sample and *m* is a cardinality index for x_i. This is indicating the cardinal position of the actual sample in the range considered. In some references [9] is found that the index m is considered as a position index for x_i indicating its position in the sample array and not in the sampling range. Applying the slight modification we can obtain broader representation for the actual variable *x*.

Having the cumulative probability P_i we can obtain the scaled value for x_i as

$$x_{i,scaled} = \mu_x + \sigma_x \Phi(P_i) \tag{3.45}$$

3.4.2 Minimum number of samples determination

Since the functions we will deal with are not simply 3-dimensional, we cannot perform a comparative analysis between the real function and the surrogate model as we have done with the 'Sombrero' test because the n-dimensional space is not visualisable.

What we can do is to compare an array containing real values with one containing predicted values generated through Kriging.

Since the real function, as said, is not visualisable we can reconstruct it on the whole design space. To do so every variable is taken at some regularly spaced points (not randomly chosen as in LHS) which we can call seeds. So having the same number of seeds on every variable axis we can combine them to obtain all the possible points in the design space where to calculate values for the test function.

Using instead random sampling through LHS we can obtain some sample points and responses to perform Kriging and build the surrogate. Evaluating the surrogate in the whole design space (seeds of the real function) we obtain the value which would be predicted by Kriging in that point.

Comparing the values obtained by the prediction with the real values we can find the average and maximum errors of the 'kriged' surface. Setting a limit for this and repeating the procedure for different sample size we can assess which is the minimum number of samples needed to well represent our original surface. Typically, and also in this work, a limit of 1% [37] on the maximum and average error is considered.

4 RELIABILITY ANALYSIS

4.1 **Introduction to reliability engineering**

Traditionally, in the design of a piping system the uncertainty in some input variables is taken into account over-sizing components or installing more pumping power than required. This results in an increase in equipment and installing costs. Some other uncertainties which occur during the operational life of the system result in frequent maintenance actions and extra costs as well.

The actual engineering practice does not consider how variables analytically affect the output of a system and is simply recommending to the designer some guidelines to follow, which have been written up according to past experience in the field.

In some engineering fields, like structural engineering, a new so-called probabilistic approach through reliability analysis is spreading but it is still characterised by niche techniques, employed by few [23, 32].

Reliability analysis is nothing but the set of tools that allow us to estimate the probability of failure of a system, where failure has to be intended as non-fulfilment of a specific requirement.

Reliability is defined as "*the ability of a system to fulfil its design functions under designated operating and environmental conditions for a specified period of time*" [48]. Theoretically is defined as the complementary to 1 of the probability of failure.

$$Reliability = 1 - P_f \tag{4.1}$$

4.2 **Concepts of reliability engineering**

The probability of failure can be seen as the probability for which a limit state for our system is exceeded. This can be expressed using a Limit State Function as

$$g = L - V \tag{4.2}$$

where L is the limit and V the actual value of the limited variable.

Our problem, as said, is subjected to k stochastic input variables which can be written in a vector form as

$$X = \{x_1, x_2, x_3, \dots x_k\} \tag{4.3}$$

Both actual value V and limit value L can be dependent on the stochastic variables hence the LSF is

$$g(X) = L(X) - V(X) \tag{4.4}$$

According to the definition of Limit State Function given above, we can mathematically define the probability of failure as the probability for the limit state condition to be unsatisfied

$$P_f = P[g(X) < 0] \tag{4.5}$$

We can identify three different situations:

- failure region $g(X) < 0$
- failure surface $g(X) = 0$
- safe region $g(X) > 0$

This concept is depicted in Figure 4.1.

If the Limit State Function is linear, mean, standard deviation and reliability index can be defined as follows;

$$\mu_g = \mu_L - \mu_V \tag{4.6}$$

$$\sigma_g = \sqrt{{\sigma_L}^2 + {\sigma_V}^2 - 2\rho_{LV}\sigma_V\sigma_L} \tag{4.7}$$

$$\beta = \frac{\mu_g}{\sigma_g} \tag{4.8}$$

The reliability index is a measure of how far our PDF is from the zero or in other words the margin of safety that the expected value (mean) has. Following the notation given in Figure 15 the probability of failure can be rewritten as;

$$P_f = \int_{-\infty}^{0} f_g dg = \int_{g(X)<0} f_g dg \tag{4.9}$$

As the PDF moves left the reliability index decreases, if it moves right it increases. This leads us to intuitively define its Cumulative Distribution Function as the area below the PDF which is still in the positive half plane. Hence the probability of failure can be re-stated also as

$$P_f = 1 - \Phi(\beta) \tag{4.10}$$

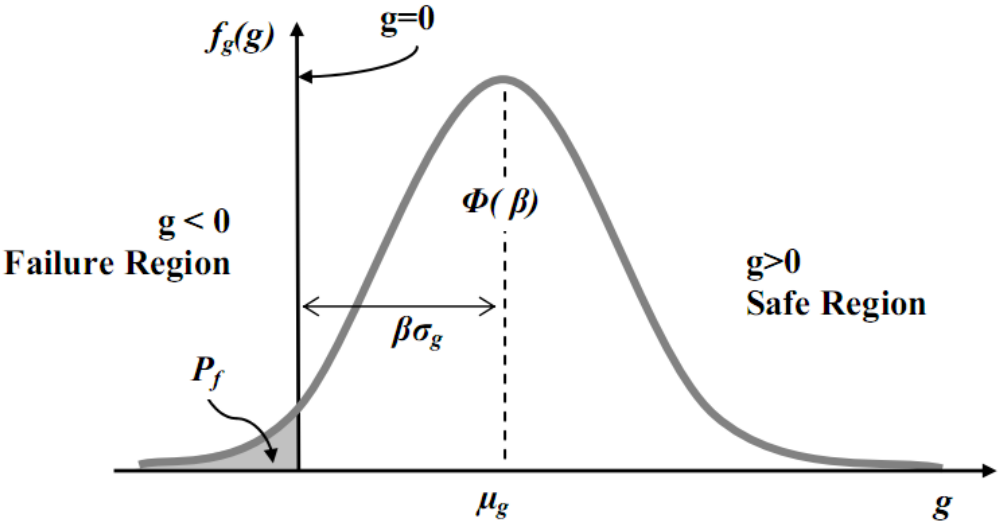


Figure 4.1. Probability density function of the Limit State Function g(X), [9]

The Probability Density Function of the LS *g(X)* is, as said, depending on all the stochastic variables which characterise our system. So considering the multi-dimensional case, the PDF is nothing but the joint probability of the design variables which integral has to be computed in order to obtain the probability of failure.

$$P_f = \int \dots \int f_X(x_1, x_2, x_3, \dots x_k) dx_1 dx_2 dx_3 \dots dx_k \tag{4.11}$$

The solution of this integral presents difficult or even impossible calculations to perform hence probabilistic methods each characterised by different computational effort and accuracy exist.

Many approximation techniques have been developed and applied. Basically, they all consist in an approximation of the stochastic variables through an equivalent geometrical representation of the same.

Stochastic variables are represented by their moments, or better the moments of area of the curves which represent their stochasticity (first moment→mean, second moment→variance, third moment→skewness, fourth moment→kurtosis).

Originally two main methods were adopted: First Order Second Moment (FOSM) and Second Order Second Moment (SOSM); which consisted in a Taylor expansion (up to first or second order) around the mean value under the failure condition $g(X) = 0$, see Figure 16.

An extensive dissertation about those can be found in [9]. From what can we see only low levels of accuracy can be reached even with second order approximation and a different formulation of the methods has been given by (Hasofer and Lind, 1973).

According to their formulation we can obtain a substantial improvement in accuracy performing a transformation of the design space (Figure 4.2) and choosing the Failure Point as the approximation point for the Taylor expansion. In this way the failure point is referred to as the Most Probable failure Point (MPP).

This last formulation is widely known as First Order Reliability Method (FORM) and an analytical derivation of the methodology is given in the next chapter.

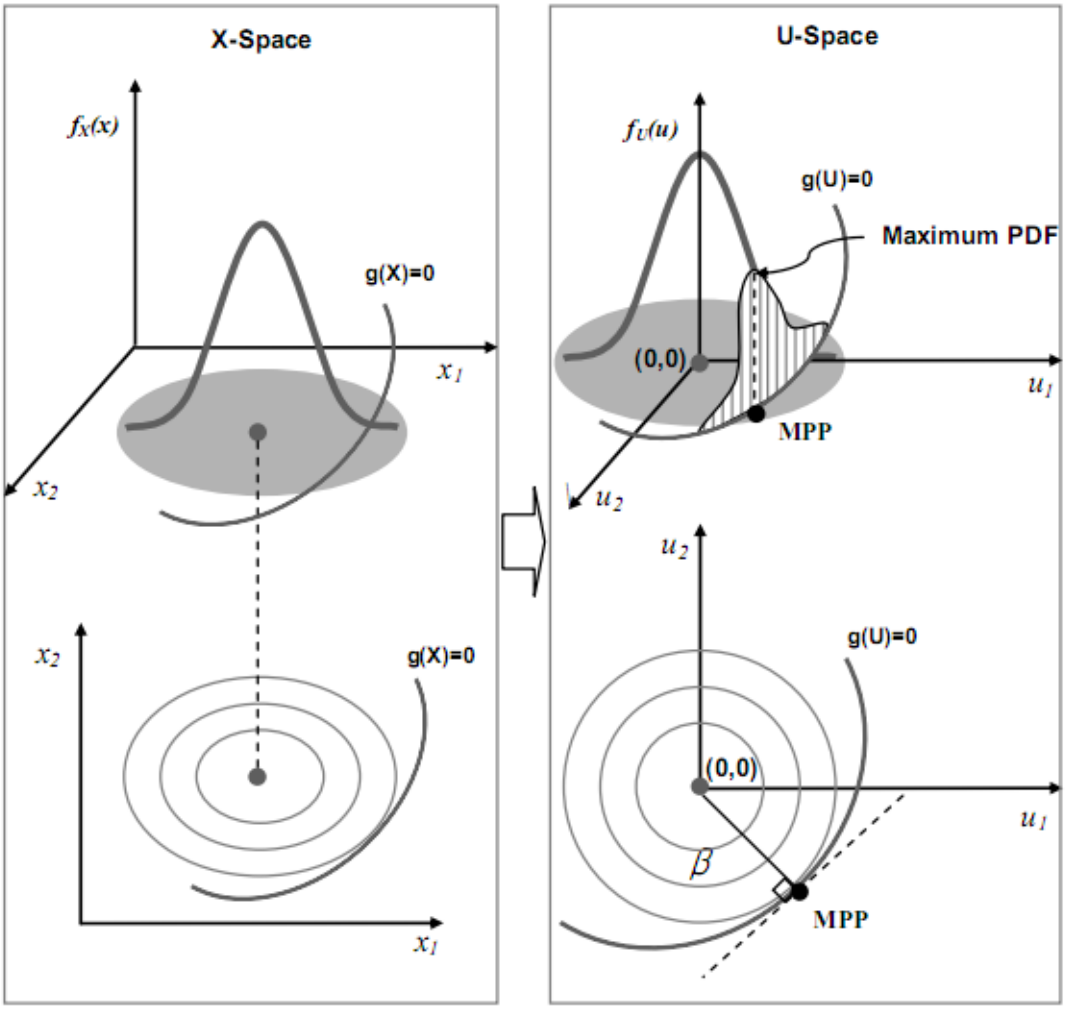


Figure 4.2. Hasofer and Lind transformation [9]

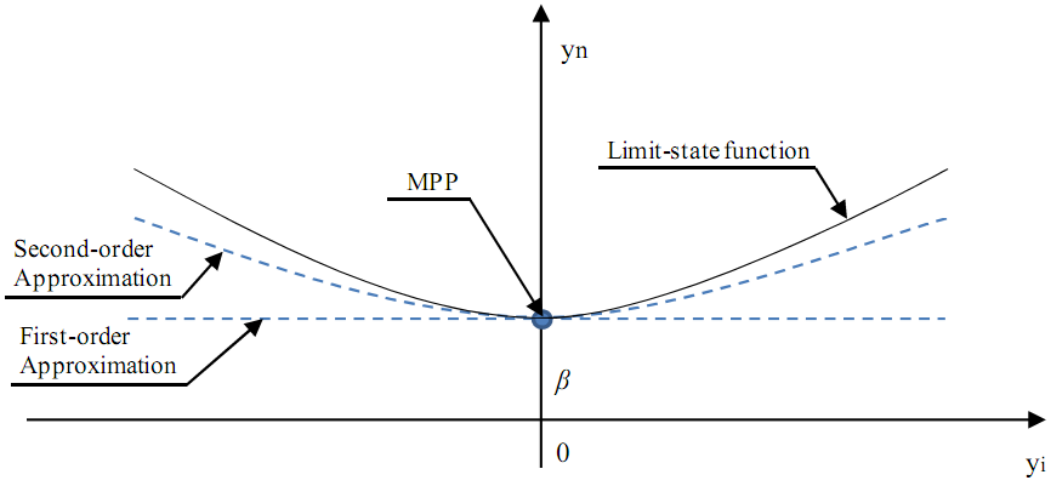


Figure 4.3. Limit State approximation

4.3 **First Order Reliability Method**

Considering L and V as the only variables (reasonable assumption since all the others combine in them) the 3D situation can be easily represented and visualised. The condition $g(X) = 0$ implies that $L(X) = V(X)$, hence the limit condition is represented by the bisector of the L-V plane. Projecting the limit state on this plane we can easily identify the failure area, the failure surface (that degenerates in a line in this simplified case), the Most Probable failure Point and the safe area (Figure 4.4).

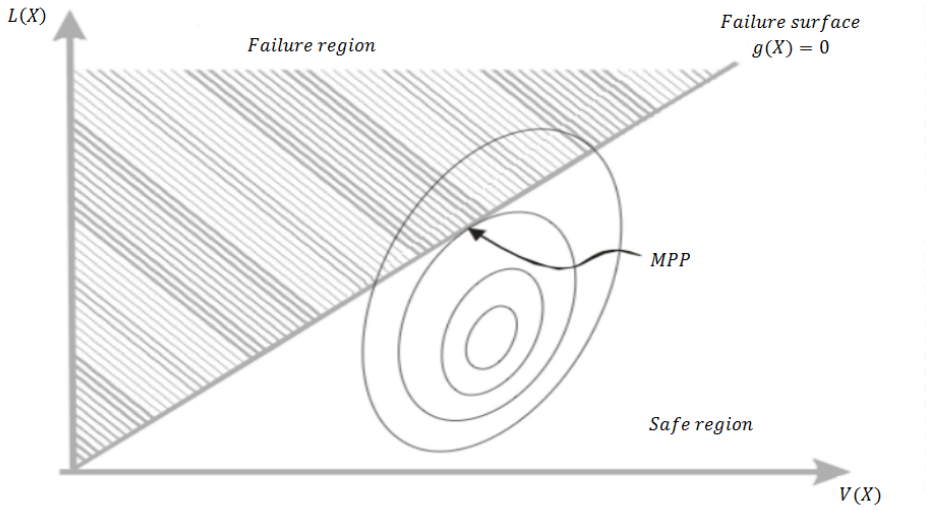


Figure 4.4. L-V plane failure region representation

Applying the transformation we have

$$Z_j = \frac{X_j - \mu_j}{\sigma_j} \tag{4.12}$$

$$\boldsymbol{X} \sim N(\mu_X, \sigma_X) \rightarrow \boldsymbol{Z} \sim N(0,1) \tag{4.13}$$

As we could expect both a transformation in the space and in the functions occurs, so that the limit condition won't be represented any more by the bisector of the axes but by a different curve.

Approximating the curve with a first order or a second order polynomial, its shape will be properly represented by

$$\tilde{g}(\boldsymbol{Z}) \cong g(\boldsymbol{Z}^*) + A \cdot (\boldsymbol{Z} - \boldsymbol{Z}^*) + B \cdot (\boldsymbol{Z} - \boldsymbol{Z}^*)^2 \tag{4.14}$$

Where A and B are generic terms, in FORM;

$$A = \nabla g | \boldsymbol{Z}^* \tag{4.15}$$

$$B = 0 \tag{4.16}$$

The failure point will be moved as well in the design space and what we want to obtain is its distance to the origin of the axes O(0,0). After the transformation what we obtain is a k-dimensional standard normal PDF, centred in the origin which has the property to be rotationally symmetrical.

What we are interested in now is the transformed condition

$$g(\boldsymbol{X}) = \boldsymbol{0} \rightarrow g(\boldsymbol{Z}) = \boldsymbol{0} \tag{4.17}$$

In our k-dimensional design space we will find more than one point satisfying this condition and we are particularly interested in the point that has the maximum probability density function value (e.g. the most prone point to failure). Identifying this point and measuring its distance to the origin gives us how much our system is reliable and we express this distance in standard deviation units. This is also well displayed in Figure 16).

This number of 'standard deviations' is defined with the reliability coefficient β which, as said, gives us a measure of how safe we are in the design and operation of the system.

Since we need a minimum distance our problem is minimisation problem [23].

$$g(\boldsymbol{Z}) = \boldsymbol{g}\left(\{Z_1\sigma_1 + \mu_1, Z_2\sigma_2 + \mu_2 \ldots Z_j\sigma_j + \mu_j \ldots Z_k\sigma_k + \mu_k\}\right) = \mathbf{0} \tag{4.18}$$

Approximating the curve with a First Order Taylor expansion in $\boldsymbol{Z}^*$ and writing it explicitly after the considerations done above

$$\tilde{g}(\boldsymbol{Z}) \cong g(\boldsymbol{Z}^*) + \sum \left.\frac{\partial g}{\partial Z_j}\right| (\boldsymbol{Z}^*) \cdot \left(Z_j - {Z^*}_j\right) \tag{4.19}$$

The derivative in the j-th direction, according to what stated in 4-17, can be rewritten as

$$\left.\frac{\partial g}{\partial Z_j}\right| (Z) = \left.\frac{\partial g}{\partial X_j}\right| (X) \cdot \sigma_{X_j} \tag{4.20}$$

According to the definition of β given in 4-7 and re-elaborating 4-18 the minimum distance from the origin to the surface can be given as

$$\beta = \frac{g(\boldsymbol{Z}^*) - \sum \left.\frac{\partial g}{\partial X_j}\right| (X) \cdot \sigma_{X_j} \cdot {Z^*}_j}{\sqrt{\sum \left(\left.\frac{\partial g}{\partial X_j}\right| (X) \cdot \sigma_{X_j}\right)^2}} \tag{4.21}$$

Defining the coordinates of the MPP in the normalised design space

$$Z_j = \beta \, cos\theta_{X_j} \tag{4.22}$$

where the cosine is the direction cosine. This trigonometric variable expresses the influence of the j-th variable on the total variation or in other words how the curve changes moving along the j-th axis. This, by definition of direction cosine, can be expressed as

$$\alpha_j = cos\theta_{X_j} = \frac{\frac{\partial g}{\partial X_j}\Big|(X) \cdot \sigma_{X_j}}{\sqrt{\sum(\frac{\partial g}{\partial X_j}\Big|(X) \cdot \sigma_{X_j})^2}} \tag{4.23}$$

These factors are so-called sensitivity factors because the value they take in the last iteration (e.g. final value after convergence) tells us the impact that the j-th variable has on failure.

The coordinates of the MPP can also be given in the original non-transformed design space.

$$X_j = Z_j\sigma_j + \mu_j = \beta\ cos\theta_{X_j} \cdot \sigma_j + \mu_j \tag{4.24}$$

Unlike other approximations mentioned before, for example MVFOSM, the Hasofer and Lind formulation of FORM is an iterative process and hence requires an algorithm to get to the solution. This is illustrated in Figure 4.5 below.

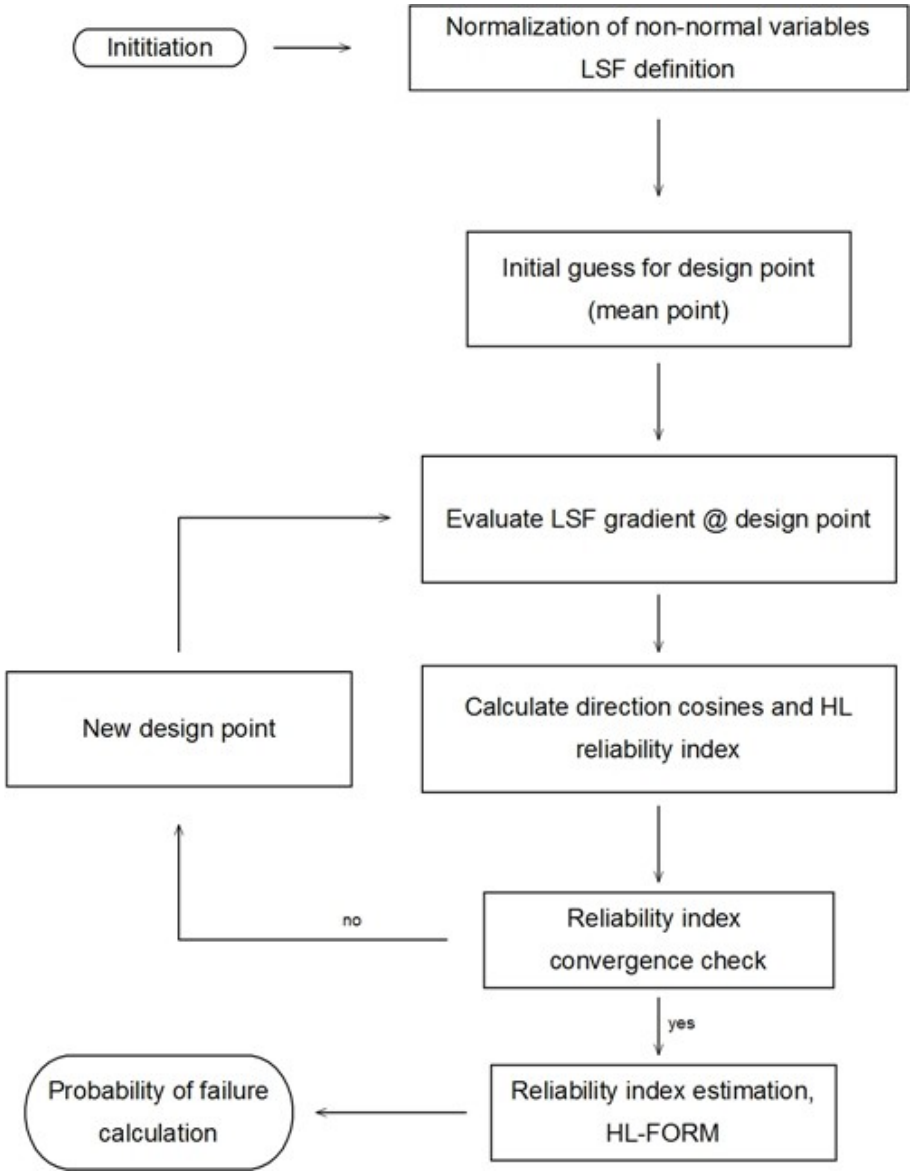


Figure 4.5. HL algorithm diagram

This procedure assumes that the input stochastic variables are normally distributed but in nature it may happen to have different distributions (i.e. log-normal, Weibull, t-Student, exponential). In order to handle these situations an extensive and deep coverage is given in [9] and is just a matter of transforming our variables into equivalent normal variables before starting with the algorithm iterations (normalisation in the block diagram above). In this work only normally distributed variables are considered because they are derived as shown in chapter 1.2. Moreover the main aim of this work is not to treat such non-normally distributed variables which only require basic statistical knowledge to be handled.

As we can see in the block diagram in Figure 4.5 the first iteration assumes some value from which to start. Basically we need the actual design point where to evaluate our function gradient but we have none. We can proceed mainly in two ways:

- assume the initial design point as the mean point (point in the design space identified by the mean value on all the coordinate axes)

- assume values for reliability index and sensitivity factors, bearing in mind that by definition of director cosine the following condition holds

$$\sum {\alpha_j}^2 = 1 \tag{4.25}$$

Having betas and alphas we can compute the coordinates of a design point and then proceed evaluating the gradient and with the first iteration of the algorithm.

4.3.1 SRSM for FORM

Among the most used techniques for quantitative assessment of systems, structures above all, there is a combination of SRSM and FORM early discussed [23, 32].

In this case the Limit State for FORM (e.g. the input function to the FORM-HL routine) is given by a second order polynomial in k variable which approximates our actual system. The approximation is given by SRSM as it has been discussed early on. So our function $\underline{g}$ will have a shape like

$$g = g_{lim} - a_0 + \sum_{i=1}^{k} b_i x_i \sum_{i=1}^{k} c_i {x_i}^2 \tag{4.26}$$

If the limit condition (upper) is deterministic or

$$\hat{g} = a_0 + \sum_{i=1}^{k} b_i x_i \sum_{i=1}^{k} c_i {x_i}^2 \tag{4.27}$$

if the limit condition is in some way affected by uncertainties, hence stochastic itself. In this case not the output from the system, but the difference between actual limit and actual limited variable is considered. This can be defined as a Margin of Safety and is given by formula 4-26

$$MoS = g = g_{lim} - \hat{g}_{actual} \tag{4.28}$$

So the quadratic polynomial straight bridges the gap between inputs and MoS.

4.4 **Monte Carlo Simulation**

Another method to compute the probability of failure is simulation method (distinguishing from the previously discussed numerical one).

Monte Carlo simulation is widely used in several engineering and non-engineering applications because it doesn't require much knowledge and statistical understanding of the problem. The algorithm is pretty easy to implement and relies on a massively high number of iterations.

It consists in launching several times the analytical model with different inputs. At each run the stochastic inputs take values according to their distribution and these values are generated randomly by a computer. Checking if the output we are interested in is above or below a certain threshold we can assess if in that particular run the system would fail or not. Thanks to a counter which increases every time we have a failure we can estimate the failure probability as

$$P_f = \frac{N_f}{N} \tag{4.29}$$

where N_f is the counter for the number of failures and N the total number of runs. A representation of the algorithm is given in Figure 4.6.

Typically to get an accurate result we should run at least 10^2 times more than the reciprocal of the probability of failure we are trying to estimate to get an accurate result [9].

$$N_{min} = \frac{1}{P_f} \cdot 10^2 \tag{4.30}$$

Not knowing the probability of failure we can start running MCS with low N just to have an idea about the order of magnitude of the number we are trying to approach. Then we can go further with more runs to ensure the number it is actually as more correct as possible.

In this way we are like sampling our problem and we are not always sure that the samples we take are effectively representing the situation we are dealing with. In particular, this sample suffers of sometimes high variance hence variance reduction techniques have been developed and are well illustrated in [9]. Moreover in [33] is found that this method is not suitable for low probabilities of failure.

Hence we can state that if we are not dealing with too low probabilities this is actually the most accurate method for probability of failure estimation. This is due to the fact that there is no system approximation.

One more drawback of the method is that, as said, it is computationally cumbersome and time consuming. Even if this can be limited applying variance reduction is still present and inacceptable for some applications.

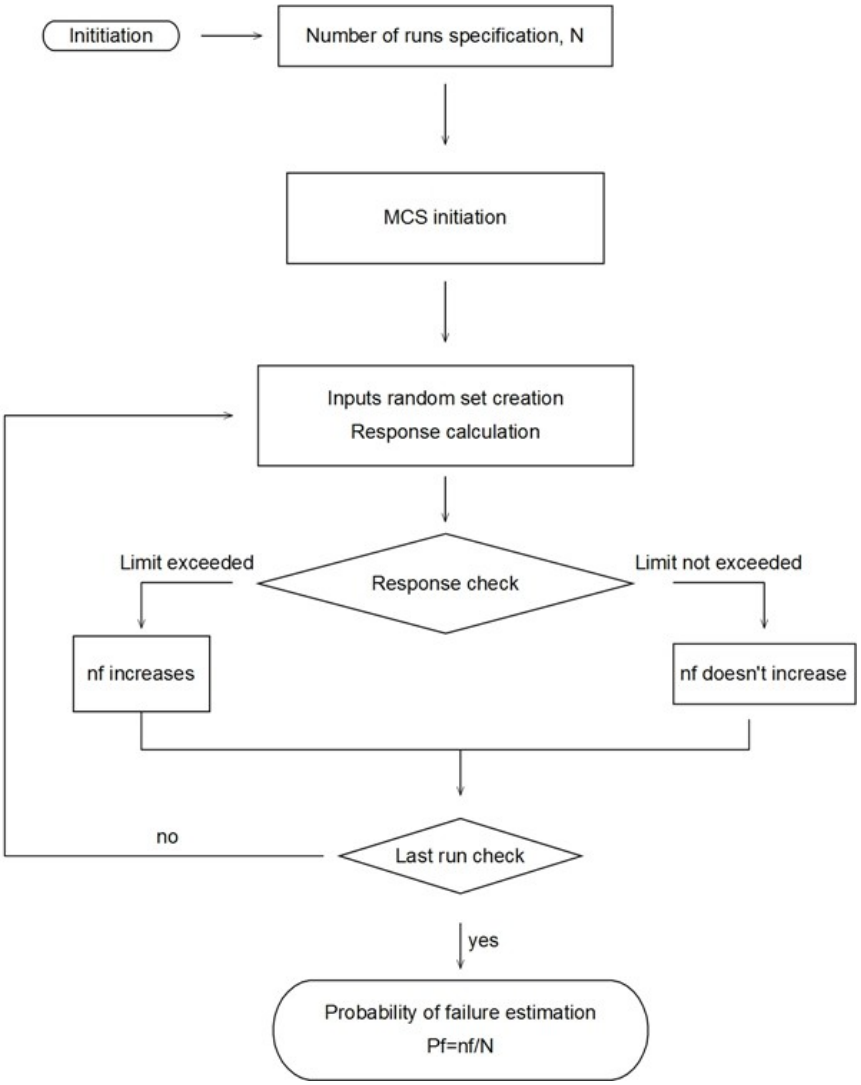


Figure 4.6. MCS algorithm

5 CONVENTIONAL METHODS FOR RELIABILITY ANALYSIS

A general overview of all the methodologies employed is given in this chapter, highlighting pros and cons, important aspects and remarks. Here only a block diagram of the code written in Matlab to implement each methodology is reported. The codes are then fully available in Appendix C.

5.1 **Direct MCS**

This direct simulation technique has just been applied to the first system because the second one doesn't permit to do so since it is modelled implicitly (e.g. with no explicit equations but by means of an interface tool, OLGA).

The way to proceed with this in order to use a number of runs large enough to obtain reliable results is to start from few runs (e.g. 10^3) and increase the number of runs by one order of magnitude until the result is stabilised. A rule of thumb suggests going two orders of magnitude higher than the reciprocal of the probability of failure to catch, following formula 4-29.

In this application no variance reduction techniques have been employed so as shown in [33] MCS is probably not able to get too low probabilities of failure. Anyway this methodology is considered as reference to benchmark the other ones.

This, as said in the previous chapters, takes a lot of time and a large computational effort often not acceptable. Simulation times up to 10'000 seconds have been experienced during the system analysis.

The random input sets are generated according to the distribution every variable shows. Random number generation functions are widely available in commercial software packages, in particular in Matlab this can be done with the function *normrnd.m*, giving as an input to the function the distribution parameters.

A block diagram explaining the code is given in Figure 5.1.

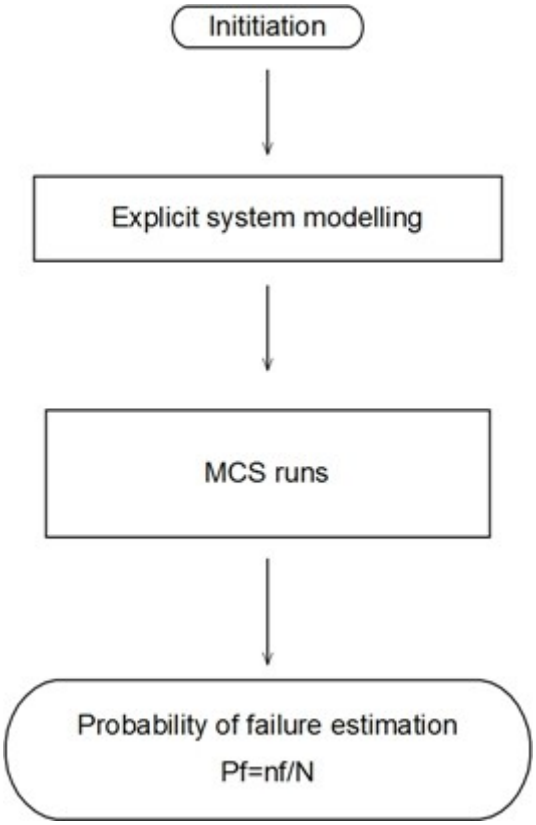


Figure 5.1. Direct MCS algorithm

5.2 **SRSM - MCS**

In order to speed up the analysis procedure approximation techniques can be applied to the system first, still using MCS as reliability analysis tool. What can happen applying this approximation procedure is that some failure points are not ‘detected’ . This effect is also overlapped to the one given by MCS which is an inaccurate system failure sampling technique for low probabilities of failure.

A block diagram showing the procedure followed by the code is given in Figure 5.2

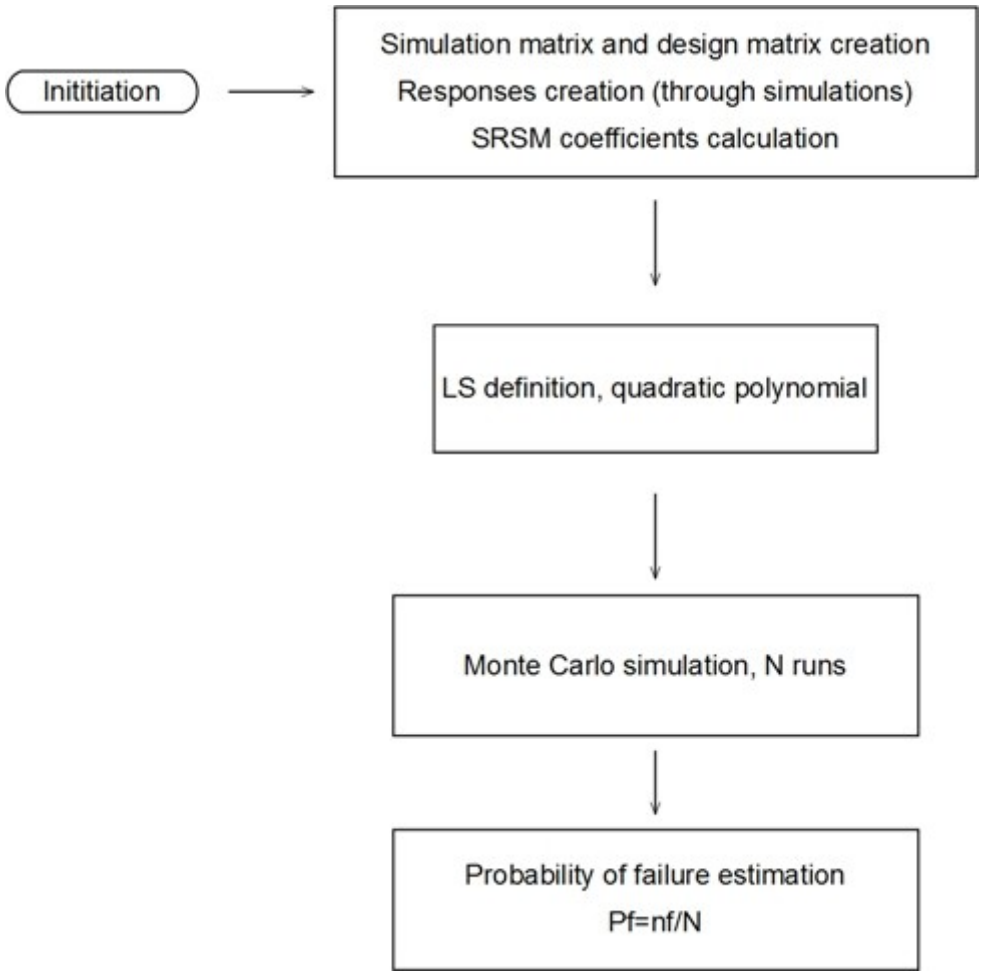


Figure 5.2. SRSM-MCS block diagram

5.3 **Kriging - MCS**

To have a better approximation of the system but still faster than the direct simulation Kriging can be applied. Then every MCS run a random prediction point can be created and the function predicted and evaluated. The Kriging function has to be built before running MCS according to evenly spread sample points. A block diagram of the procedure followed is given in Figure 5.3

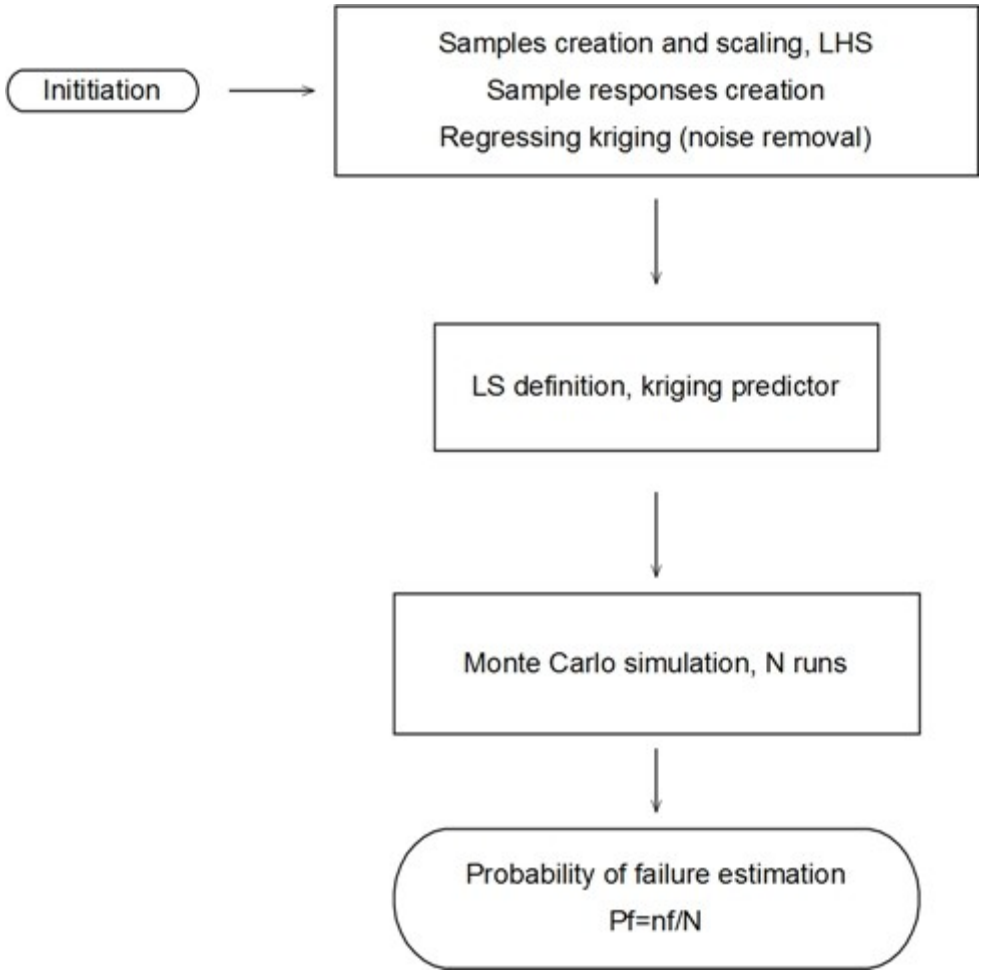


Figure 5.3. Kriging-MCS block diagram

5.4 **SRSM - FORM**

Introducing a further approximation on the reliability method the simulation time drastically drops to few seconds per each Limit State evaluation.

This is by all means the fastest procedure among the ones analysed but also the most approximated one. This is proven by the fact that sometimes the results are not accurate at all (e.g. far away from the ones obtained using direct MCS).

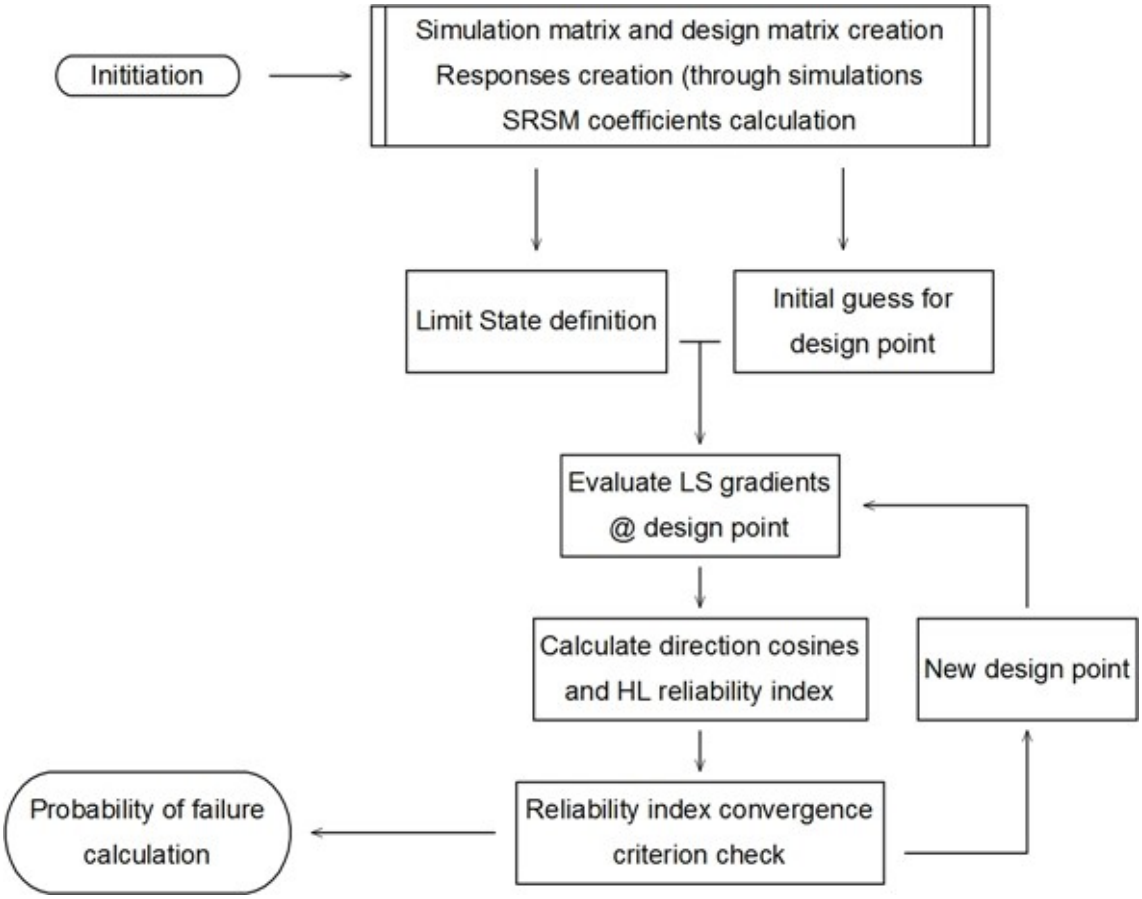


Figure 5.4. SRSM-FORM block diagram

6 ADVANCED SURROGATE MODELLING METHODS FOR RELIABILITY ANALYSIS

As seen in literature, there's now a movement towards new techniques for Reliability Assessment especially in the system approximation. An interesting study can be found in [37] where the system is approximated using kriging and Reliability Analysis is carried out using direct simulation method (i.e. Monte Carlo Simulation). A development of new techniques and a trial to couple kriging and FORM is done in this work.

Here the method is illustrated for normal variables only. Non normal variables can be incorporated using the same concepts in various conventional methods. In particular is possible to refer to [9, 53] where extensive discussion is reported.

Basics of the theory behind the process are given in this chapter.

6.1 Analytical Kriging for First Order Reliability Analysis

In actual applications the limit state is, mostly represented by a second order polynomial in *k* variables.

kriging parameters were obtained though kriging approximations performed on the reference system can be used to explicitly write our limit state expressing through the kriging predictor

As seen in Chapter 3, the kriging predictor can be written as

$$y^*(\boldsymbol{x}) = \hat{\mu} + \boldsymbol{\psi^T \Psi^{-1}}(\boldsymbol{y} - \mathbf{1}\hat{\mu}) \tag{6.1}$$

Elements in this equation are defined as:

$\boldsymbol{\psi^T}$ is a 1xn vector of correlations between the prediction point and the i-th sample.

$\boldsymbol{\Psi^{-1}}$is the inverse of the square $n_x n$ matrix of correlations between the samples.

$\boldsymbol{y} - \mathbf{1}\hat{\mu}$ is the nx1 vector of responses diminished of the MLE mean value.

What we obtain from this multiplication is a single value that represents the deviation from the mean value $\hat{\mu}$.

If we try to express this matrix multiplication we can distinguish between one part of the formula dependent on the new prediction point, which is ψ^T and a second part independent from this, $\boldsymbol{\mathcal{G}} = \boldsymbol{\Psi^{-1}}(\boldsymbol{y} - \mathbf{1}\hat{\mu})$.

$\boldsymbol{\mathcal{G}}$ is a nx1 matrix where each line can be expressed as

$$\boldsymbol{\mathcal{G}}(\boldsymbol{i}, \mathbf{1}) = \boldsymbol{\Psi^{-1}}(i,1)(y_1 - \hat{\mu}) + \boldsymbol{\Psi^{-1}}(i,2)(y_2 - \hat{\mu}) + \cdots + \boldsymbol{\Psi^{-1}}(i,n)(y_n - \hat{\mu}) \tag{6.2}$$

The multiplication $\boldsymbol{\psi^T \mathcal{G}}$ can also be expressed as

$$\boldsymbol{\psi^T \mathcal{G}} = \psi^T(1,1)\mathcal{G}(1,1) + \psi^T(1,2)\mathcal{G}(2,1) + \cdots + \psi^T(1,n)\mathcal{G}(n,1) \tag{6.3}$$

Where, as previously mentioned

$$\psi^T(1,i) = e^{-\{\theta_1|x_{1,i}-x_1|^2+\cdots+\theta_1|x_{k,i}-x_k|^2\}} \tag{6.4}$$

as the limit state is then expressed as

$$g = y_{lim} - y_{pred} = y_{lim} - (\hat{\mu} + \boldsymbol{\psi^T \mathcal{G}}) \tag{6.5}$$

If the limit condition is not stochastic; or simply

$$g = \hat{\mu} + \boldsymbol{\psi^T \mathcal{G}} \tag{6.6}$$

if it is; as explained for the case of quadratic polynomial SRSM.

The previously analysed procedure is what is later on mentioned as Limit State definition.

The algorithm followed for this analytical kriging implementation in FORM is shown in Figure 6.1.

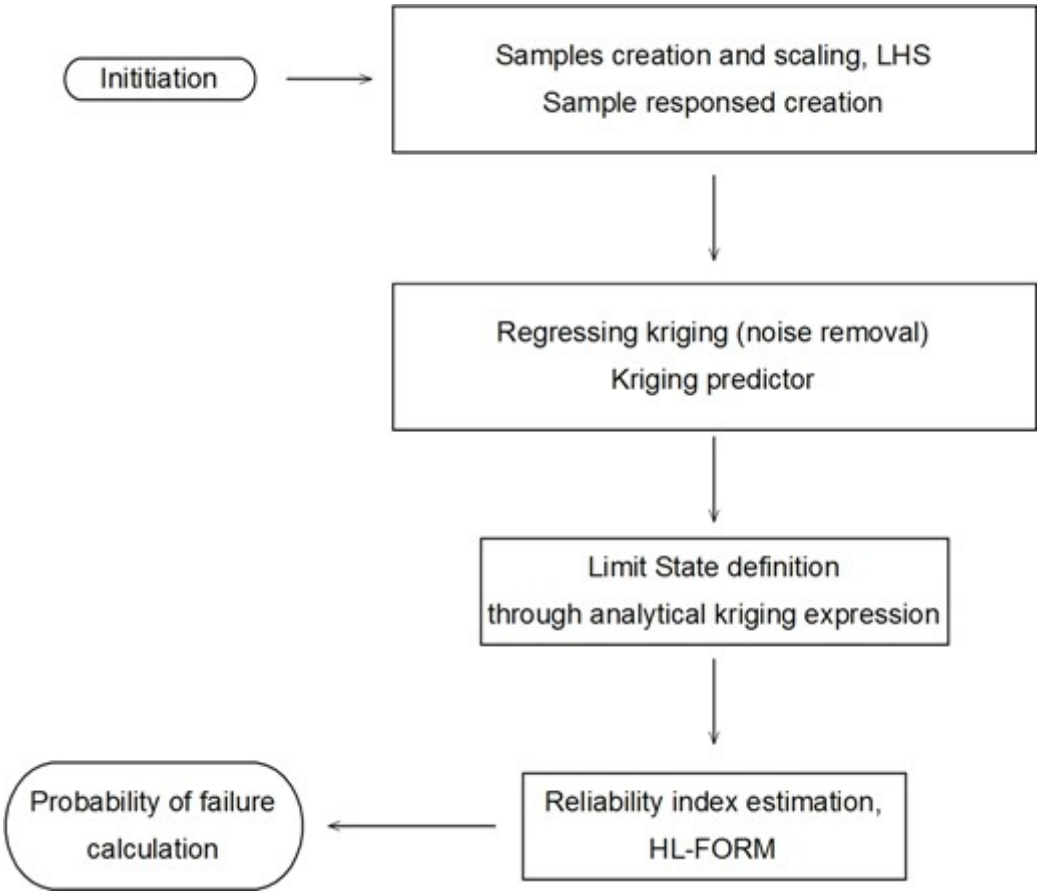


Figure 6.1. Analytical kriging-FORM algorithm

6.2 **Kriging and Dynamic Stochastic Response Surface Method for First Order Reliability Analysis**

The methodology here described is referred throughout this work to as ‘dynamic SRSM’ or ‘dynamically Kriged LS’. The name refers to the dynamicity of the method with regards to the continuous change in the LS through Kriging prediction and local approximation of the LSF.

As discussed earlier on in the chapter, where a comparison between kriging and SRSM is done, the ability of kriging is to reach all the sample points. This can’t be done using SRSM despite a local approximation is considered only. Moreover kriging is very good in predicting function values.

If a small domain around a peak point is considered then SRSM is also able to map the function well.

Furthermore, kriging function complexity increases with the number of samples and it is hard to manipulate it effectively when the number of points is high. Function manipulation (e.g. evaluation, differentiation etc...) is essential to perform Reliability Analysis.

SRSM is shown not to work well globally but can do a good job locally. Since kriging works well globally the positive features of the two methods can be coupled together, avoiding the drawbacks. A valuable example to show to the reader that the local approximation starting from a ‘kriged’ surface works is ‘Sombrero’ test (chapter 3.3.2) because the results can be visualised properly.

Here a kriging prediction of the function is done with more points than the sample ones. Particularly an evaluation of the predicted curve is done at equally spaced points. The result is reported in Figure 6.2.

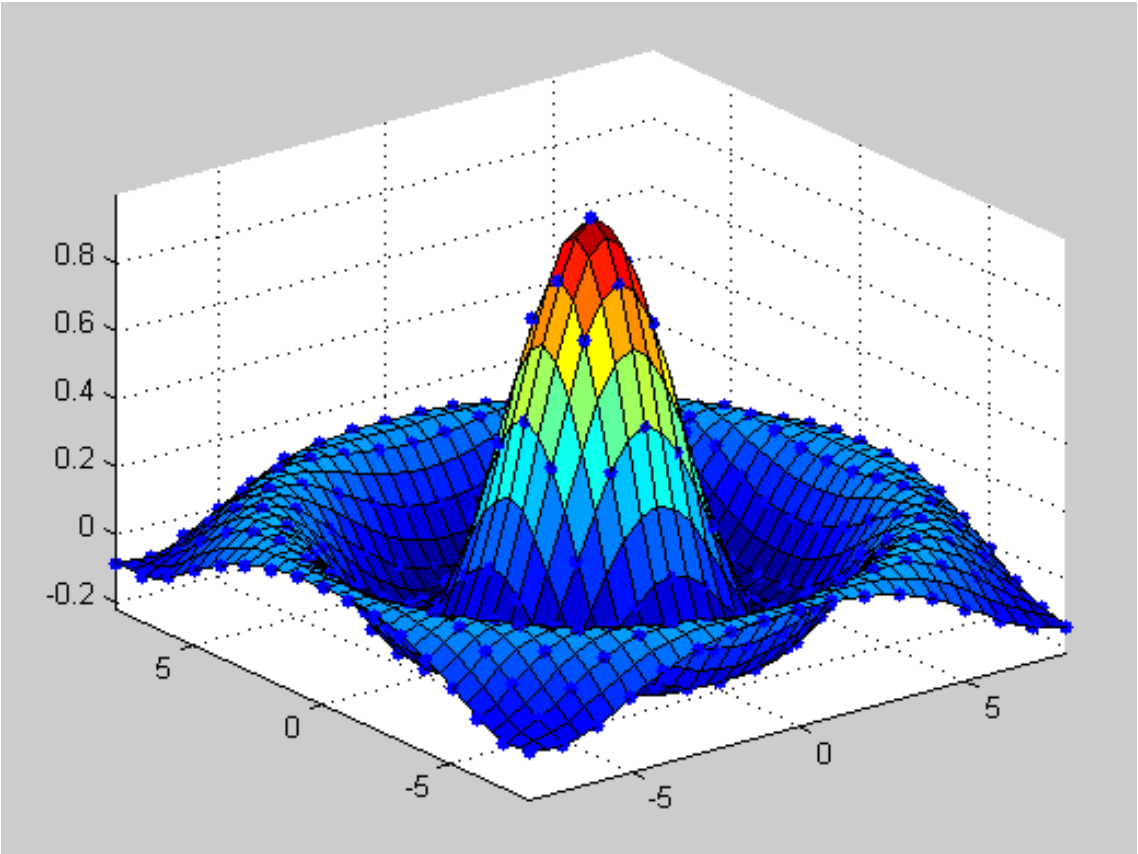


Figure 6.2. Kriged 'Sombrero' surface, equally spaced prediction points

If we consider this function on a restricted domain, let's say [-2, 2] performing SRSM and evaluating the obtained second order polynomial at equally spaced points we obtain;

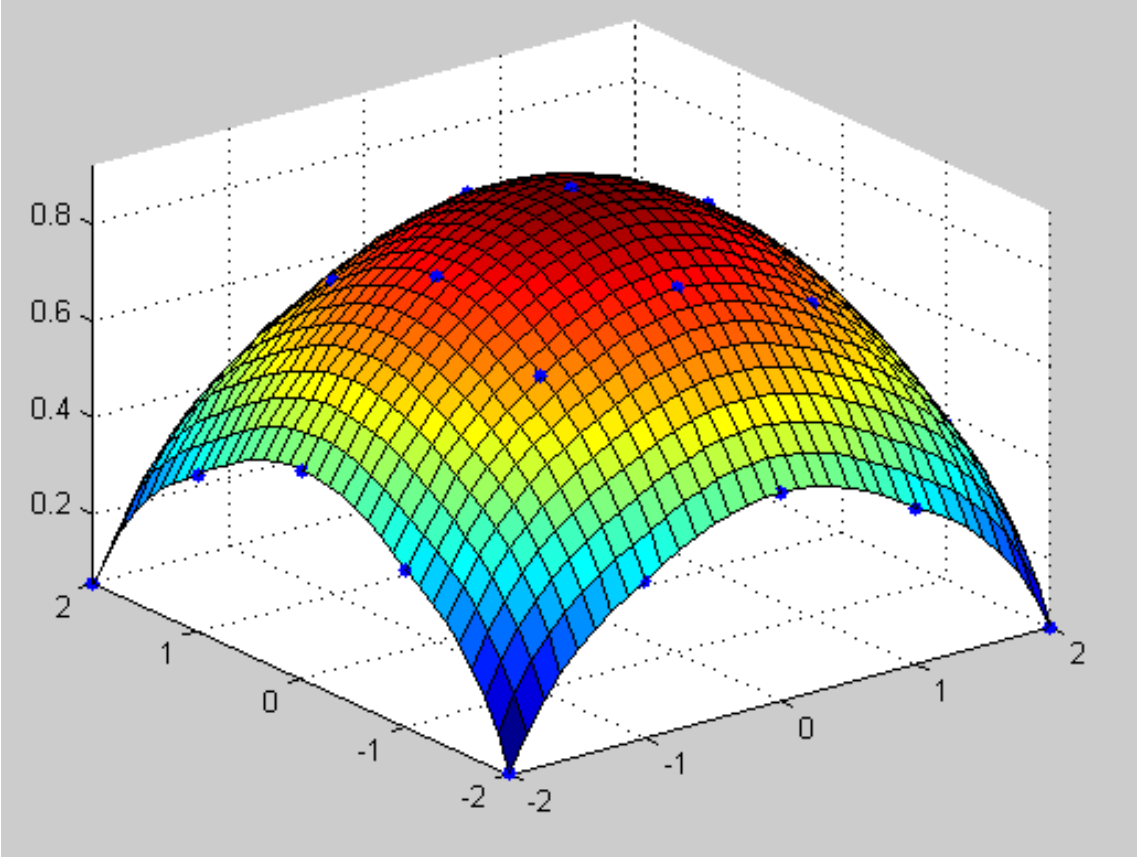


Figure 6.3. SRSM from 'kriged' prediction

which is far better than what achieved and showed in Figure 3.4.

As we can see the top of the peak achieved by this last function is pretty much the same as the initial function. Hence we can state that an extremely good and light local approximation of our initial function is achieved in this way.

To couple this accurate approximation with reliability analysis we can proceed as follows. We can predict some points around the current design point using kriging and approximate this 'local' curve using SRSM. Then a quick FORM iteration can be done using this quadratic polynomial as the LSF because it is much lighter than the analytical kriging one. Through the FORM iteration we can approach and calculate a new design point around which predict some values, build an SRSM and proceed for the next iteration.

This procedure algorithm is showed in Figure 6.4.

An intelligent point from which the algorithm can be initiated (as in a normal FORM-HL optimisation algorithm) is the mean point (i.e. the mean value point for each variable). Following the *2k+1* combination rule for sampling we can get the *n* samples to build the SRSM.

$$x_{j,\ ith\ sample} = \mu_{x_j} + f\sigma_{x_j} \tag{6.7}$$

The f factor can be termed as 'amplitude' of the domain mapped by kriging and approximated by SRSM. This also gives us the size of the area considered in the following FORM iteration. For this reason it can be considered as an 'horizon factor' giving us a measure of how far the FORM iteration goes to search for the optimum point. The bigger f the farther the research can go and the quicker we can get to the optimum point since we do 'bigger steps' iteration by iteration. The smaller the horizon factor the closer we stay to the design point, the more accurate the approximation of the function, the smaller and more the steps towards the optimum point. In this second case the simulation time will definitely increase but the accuracy of the result obtained will be higher. A sensitivity study on this factor has been carried out with values from 0.001 to 1.8 and good results have been obtained with values between 0.2 and 0.4.

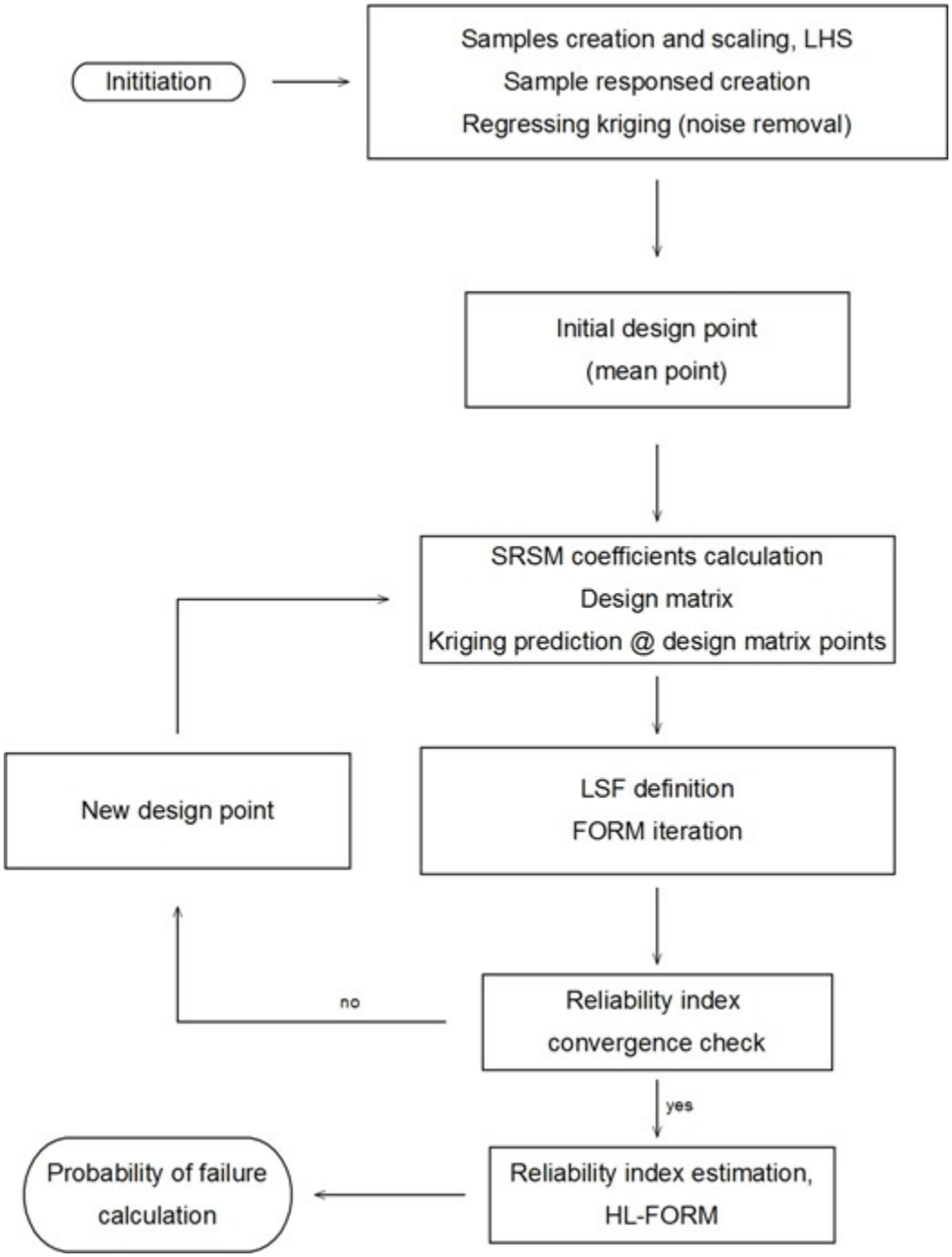


Figure 6.4. Dynamically kriged LS-FORM algorithm

7 APPLICATION AND DISCUSSION OF RESULTS

In this chapter applications of the discussed methodologies and the aspects under which the system analysis is carried out are reported. First the models built are described and visualised through clarifying 3D views.

Then, after describing the functioning of the systems, some criticalities are pointed out and critical scenarios are built.

A resume of the available methodologies is given analysing how to match the different techniques described in the previous chapters. Here also pros and cons of each methodology are discussed, giving the expected behaviour of each one regarding accuracy and simulation time.

Finally results are presented and discussed, commenting what obtained and comparing the effectiveness of different analysis strategies.

All the methodologies have been applied to the first system. Because the model is explicit even direct simulation techniques could be applied.

This gives us the opportunity to evaluate and benchmark the performance of different methodologies so as to highlight pros and cons of each. Having this information is also easier which to choose for future applications.

From the analysis carried out on the first system we can say that the most accurate tool, after the direct MCS is Kriging-MCS. Since as said direct MCS is not applicable because no explicit model is present this second option is considered.

Then, being SRSM-FORM the quickest and most diffused simulation technique, it's as well applied. Because of the high number of elements (segments of pipeline) and the need to perform an analysis over time all the computationally expensive techniques other than Kriging-MCS are put aside. Because of the quickness showed and the global accuracy achieved, Dynamically Kriged LS is also applied.

In the second case study a complete analysis on the whole system is performed using SRSM-FORM, which is the quickest one. Then, in order to save time further investigation is performed on critical points only for which the results are confirmed using other techniques. In particular, Kriging-MCS and Dynamically Kriged LS are applied.

7.1 **Analysis scenarios**

7.1.1 Limit states

Our system output is as said limited either by a deterministic limit or by a stochastic one. It has already been shown in equation 4-25 and 4-26 how to formulate limit states in the two different cases.

The functioning mode of the system has some intrinsic limits that the analyser has to detect and explore with the presented methods.

Many failure modes can be represented quantitatively and a qualitative study to identify them before performing a quantitative analysis is suggested.

7.1.2 One-phase system analysis scenarios

For what concerns the one-phase system 4 different scenarios are analysed. In the first three 6 stochastic variables are considered: pressure in the tank, tank water level, roughness of the pipe, temperature, gate valve component loss and pump characteristic. In the last one more is considered, the flow rate, since the pump is assumed to work at a certain regime (rotating speed) so delivering a certain flow rate independently from the head loss to overcome.

The first one, named scenario 1A, is formulated so that the limit is represented by a maximum pressure in the pipe. The aim of this analysis is to assess, through the methodologies presented, what is the probability of failure of the system considering as a failure mode the exceedance of the burst pressure. The limit value is in this case deterministic and taken from the piping [1]. Even this value is dependent from wall thickness and yield strength of the material (Equation 2-13) which are stochastic variables a deterministic value is taken instead. An advanced analysis which takes into consideration also this is performed in the second case study. In particular for this scenario the pressure limit is considered just downstream the pump, in the point called A in the 3D schematic given in Figure A.4.

The second analysis scenario for the first application, named scenario 1B, is considering the maximum pressure allowable in a flange. Since no particular flange is selected and no manufacturers' indications are given for the one in the real rig, a maximum value for the pressure is assumed. In this scenario the failure mode leakage due to high pressure is investigated.

Scenario 2 analyses the energy efficiency of the system assuming that the power supply is limited. The pump, which is the only energy consuming device in the system, has to work in a regime so that its efficiency is higher than a certain assumed limit value.

In the last analysis, scenario 3, the difference in pressure is taken as a limit. The flow has to be ensured because of this difference so at least atmospheric pressure has to be present in the tank. As said the pump is assumed to work in a constant regime delivering a given flow rate.

The uncertainties are expressed in terms of CoV and given in Table 7.1.

Table 7.1. One-phase flow case study uncertainties

	CoV
Pipe roughness [mm]	10,9%
Inlet pressure [Pa]	0,5%
Temperature [K]	0,2%
Elevation [m]	0,5%
Globe valve *k*	15,4%
Pump characteristic *a* [m]	10,1%
Flow rate [m3/s]	5,0%

7.1.3 Oil-gas pipeline analysis scenario

As said the maximum pressure allowed in a pipe so that the pipe itself withstands the load is given by the Barlow's formula. This is mainly depending on yield strength, thickness and diameter. The last one can be considered as deterministic but the first two have to be considered stochastic so that stochasticity is introduced also in the limit. So, the limit state is formulated in terms of *MoS* as the limit value minus the actual value. A surrogate model is then built directly from stochastic inputs (stochasticity coming both from system inputs and system limit) to the *MoS* values.

Since the environment where the pipe has to operate is harsh, corrosion is an important factor to be considered. Only internal corrosion is assumed and modelled according to equation 2.12. External corrosion is prevented by means of coating materials and because frequent external inspection and maintenance can be carried out. Regarding internal corrosion, thickness deterioration occurs over-time so that the pressure the pipe can withstand is diminishing. The safe operability level, represented by the safety index, is also changing over-time and in particular decreasing. An estimate of the time needed before inspection or maintenance can in this way be done choosing some target safety levels. Typical safety levels for pipelines [5].

In our application limits at 10^{-4} and 10^{-5} are chosen as maximum danger and warning levels (Ultimate Limit State and Serviceability Limit State).

The uncertainties level considered for this particular case expressed in terms of *CoVs* are reported in Table 7.3 below.

Table 7.2. Target Reliability levels [5]

Limit states	Safety classes		
	Low	Normal	High
SLS	10^{-1}-10^{-2}	10^{-2}-10^{-3}	10^{-2}-10^{-3}
ULS	10^{-2}-10^{-3}	10^{-3}-10^{-4}	10^{-4}-10^{-5}
FLS	10^{-3}	10^{-4}	10^{-5}
ALS	10^{-4}	10^{-5}	10^{-6}

Table 7.3. Two-phase flow case study uncertainties

	CoV
Well pressure [bar]	1,0%
Temperature [°C]	0,2%
Flow rate [kg/s]	0,3%
Pipe roughness [mm]	10,7%
Sea temperature [°C]	0,8%
Heat transfer coefficient [$W/m^2.°C$]	10,0%
Yield strength [MPa]	1,0%

7.2 **Results**

The results obtained with different methodologies are reported in terms of safety index and compared on that scale. Sometimes very low probabilities of failure are obtained and it doesn't make much sense to compare them on a scale ranging from 10^{-5} to 10^{-6}.

Simulation times are also reported to show the benefits in terms of saved computational effort using some approximation techniques.

Moreover, for those scenarios where analysis over-time is carried out (e.g. two-phase flow system) prediction of the safety index trend is also given.

Finally, for each case the impact of the single stochastic variable on failure is discussed through the analysis of FORM sensitivity coefficients α_j and Kriging activity parameters θ_j.

The numerical results are first reported in tables and then summarised in graphs so that the concepts that inspired the analysis are easily spotted.

7.2.1 One-phase flow system

First of all benchmark data are collected from computer experiments using MCS. The procedure to determine the minimum number of runs to get accurate values is the one discussed in chapter 4.

Values for the reliability index are reported in Table 7.4 and a graph showing the convergence of the result value for scenario 1B is shown in Figure 7.1.

Table 7.4. MCS results for one-phase system

	# runs	1.00E+03		1.00E+04		1.00E+05		1.00E+06		1.00E+07		1.00E+08	
beta	1A	2.75	2.88	3.01	3.06	3.09	3.11	3.09	3.09				
	1B	1.73	1.80	1.69	1.67	1.68	1.67	1.67	1.67				
	2	1.64	1.66	1.61	1.63	1.64	1.64	1.64	1.65				
	3			3.72	3.93	3.89	3.78	3.82	3.85	3.84	3.83	3.83	3.83

Where the beta value is not reported that means that the probability to catch is too low with the actual number of iterations.

It's easy to notice that the higher beta the lower the probability of failure hence the higher the maximum number of runs needed.

The same investigations are then carried out using SRSM-FORM standard procedure, approximating the LSF with a quadratic polynomial as shown in the previous chapters. In table 7-6 are reported the results.

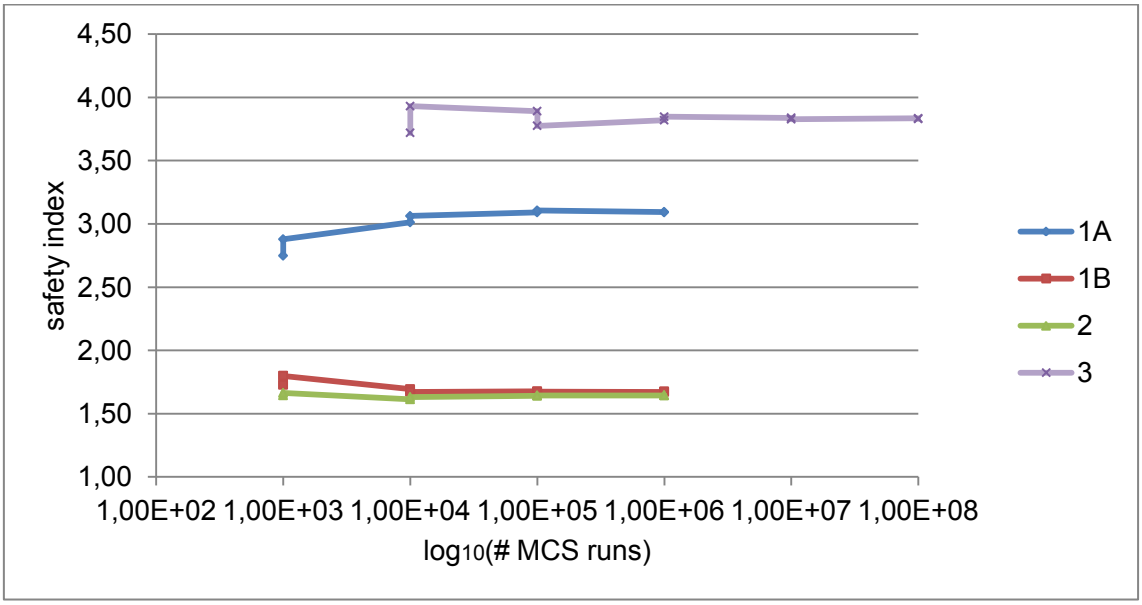


Figure 7.1. MCS results convergence

Table 7.5. MCS simulation times

	runs	1.00E+03		1.00E+04		1.00E+05		1.00E+06		1.00E+07		1.00E+08	
sim time [s]	1	2.99	2.73	26.7	26.5	261.6	257.5	2606.9	2575.5				
	2	8.05	8.96	85.2	86.3	885.7	869.3	6563.9	6867.0				
	3	4.03	4.08	40.1	40.5	400.1	404.5	3445.9	3444.4	32657	31865	315982	317497

Table 7.6. SRSM-FORM results for one-phase system

	1A	1B	2	3
P_f	0.000698	0	0.0447	0.000367
beta	3.1954	10.9436	1.6984	3.3767
time [s]	0.79	0.69	0.65	0.49

Table 7.7. SRSM-MCS results for one-phase system

	# runs	1.00E+03		1.00E+04		1.00E+05		1.00E+06		1.00E+07	
beta	1A		3.09	3.43	3.12	3.23	3.12	3.21	3.18		
	1B					3.94	3.89	3.89	3.90	3.88	3.88
	2	1.79	1.70	1.68	1.66	1.70	1.71	1.71	1.72		
	3	3.09			3.72	3.81	4.01	3.93	3.92		

As we can see we don't get high accuracy in the results for all scenarios so further analysis is required. SRSM-MCS which is another common technique is applied next. Results obtained are reported below in Table 7.7.

As we can see the method is still inaccurate in estimating the probability of failure for scenario 1B even if the simulation time is definitely decreased if we compare it with direct MCS. Simulation times reported in Table 7.8 .

What we can conclude up to now is that SRSM is not giving an accurate approximation of the system. Following these considerations the analysis is then conducted using a different surrogate modelling technique, kriging.

The first analysis where kriging surrogate modelling is used is kriging-MCS. Results are reported in Table 7.9 below;

Table 7.8. SRSM-MCS simulation times

	# runs	1.00E+03		1.00E+04		1.00E+05		1.00E+06		1.00E+07	
sim time [s]	1	0.63	0.67	3.34	3.43	32.12	32.06	313.59	318.95	3251.00	3079.00
	2	0.32	0.50	2.13	2.13	19.82	19.92	203.18	208.49		
	3	0.20	0.20	2.00	2.03	19.93	20.61	202.73	201.02		

Table 7.9 Kriging-MCS results for one-phase system

	# runs	1.00E+03		1.00E+04		1.00E+05		1.00E+06	
beta	1A	2.88		3.29	3.19	3.20	3.18	3.12	3.13
	1B			2.17	2.37	2.26	2.24	2.26	2.27
	2	1.64	1.55	1.61	1.58	1.58	1.57	1.58	1.58
	3	3.09	2.88	2.93	3.04	3.09	2.97	3.06	2.99

Table 7.10. Kriging-MCS simulation times

	# runs	1.00E+03		1.00E+04		1.00E+05		1.00E+06	
sim time [s]	1	115	224	220	380	540	513	4879	5394
	2	148	152	157	161	301	290	2971	2866
	3	198	180	210	221	588	564	3715	3694

After this analysis we see some improvement in the results and the simulation times are still reasonably low. Carefully looking at the results it can be realised that probably also using MCS a bad sampling is done. For this reason coupling of Kriging and FORM is done in the two different newly explored techniques.

First implementation of analytical kriging in LSF is done and results are reported below.

Table 7.11. Analytical kriging-FORM results for one-phase system

	1A	1B	2	3
beta	3.154	1.730	1.686	3.415
Pf	0.000806	0.041838	0.045863	0.000319
time [s]	120.411	124.753	130.22	124.374

Then the Dynamically Kriged LS algorithm is applied.

Table 7.12. Dynamically Kriged LS results for one-phase system

	1A	1B	2	3
#samples	30	75	30	100
f	0.1	0.1	0.3	0.2
Pf	0.00086	0.03912	0.04946	0.00050
beta	3.135486	1.761015	1.650127	3.288858
time [s]	223	389	212	467
#iterations	117	238	87	168

Since the results reported in this way are not readable and they have to be carefully analysed to get some meaning out of them a better visualisation of the same is given in Figure 7.2. Also a comparison of the simulation times is given, in Figure 7.3.

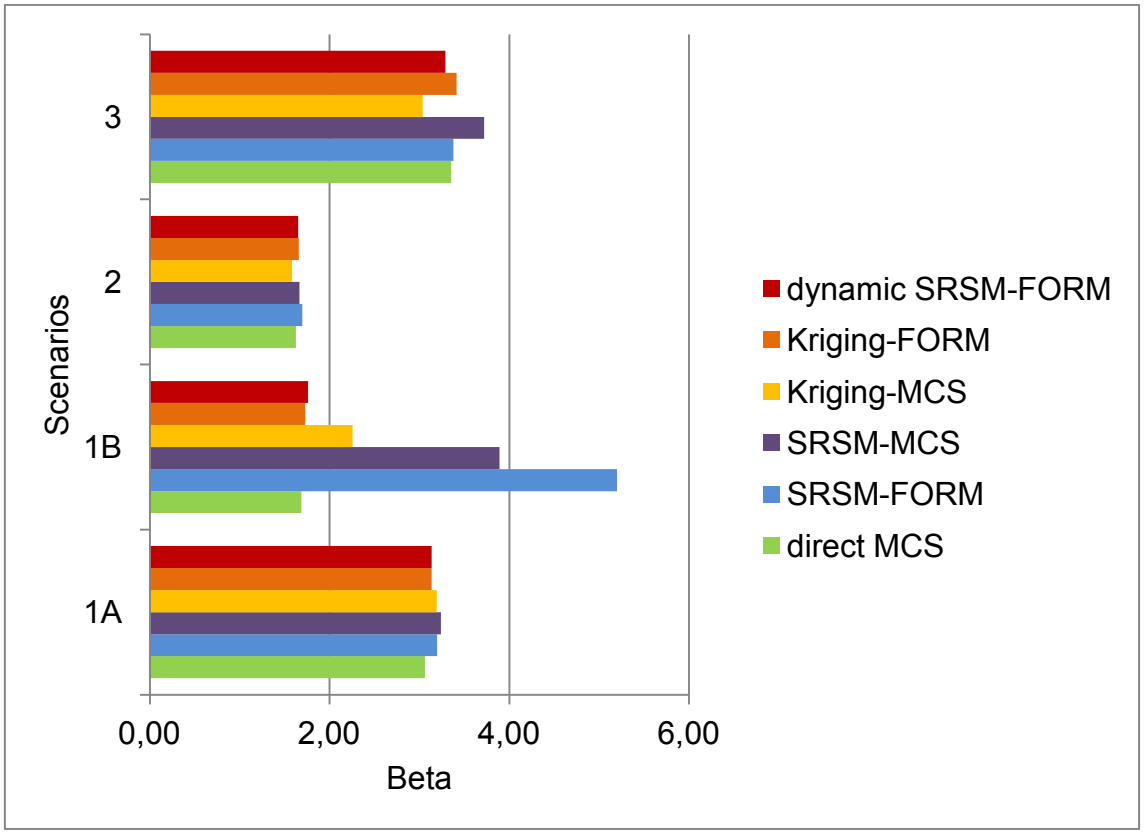


Figure 7.2. Safety index results for one-phase system, comparative graph

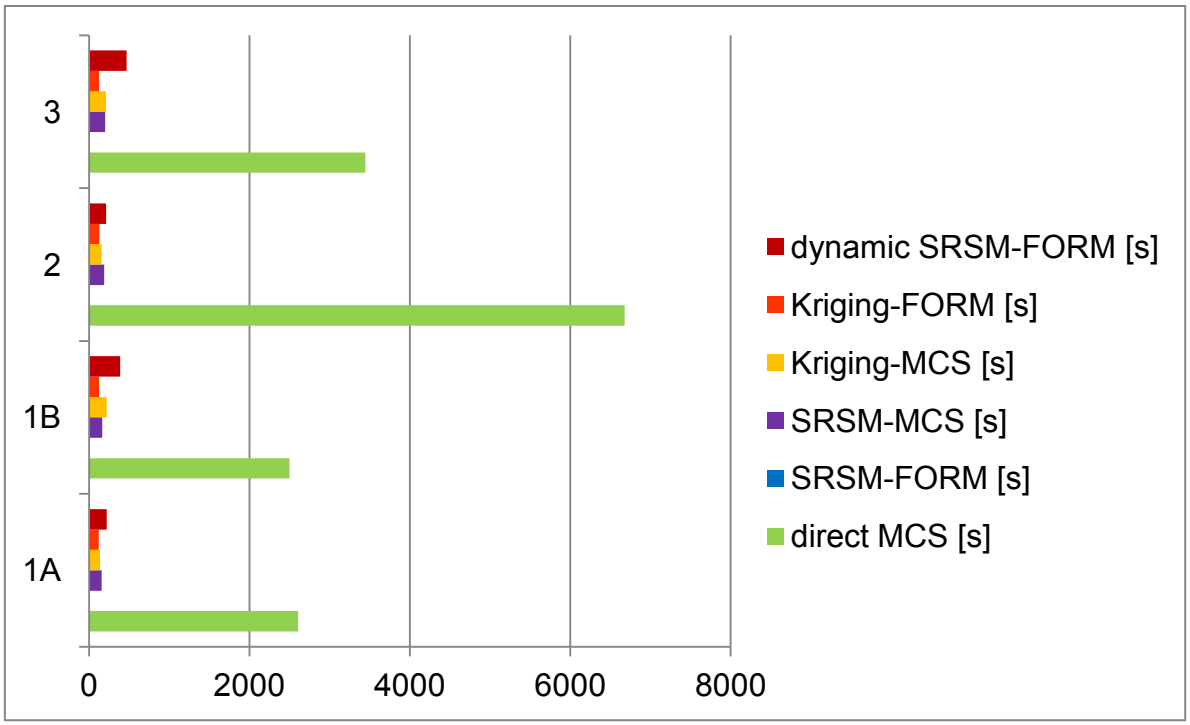


Figure 7.3. Simulation times for one-phase system, comparative graph

7.2.2 Two-phase flow system

As said no direct simulation can be implemented in this scenario and the first analysis is carried out using SRSM-FORM. Target reliability levels to maintain over-time are 10^{-4} and 10^{-5} corresponding to 3.71 and 4.26 safety index respectively. The analysis conducted spotted out which points are likely to fail first and the time range when this can happen.

To have an idea, simulations have been run for time instants ranging from 1 to 2.7 years. High probability of failure is obtained in a so short time because of the operating condition assumed. The operating pressure is quite high and for this reason even the smallest thickness deterioration affects a lot the safety of the system.

The results are reported below in Figure 7-4, Figure 7.5 and Figure 7.6 where visualisation of safety levels is done through colours mapping. Red means low reliability (e.g. high probability of failure) and blue the opposite.

Detailed numerical results are reported in Appendix D.

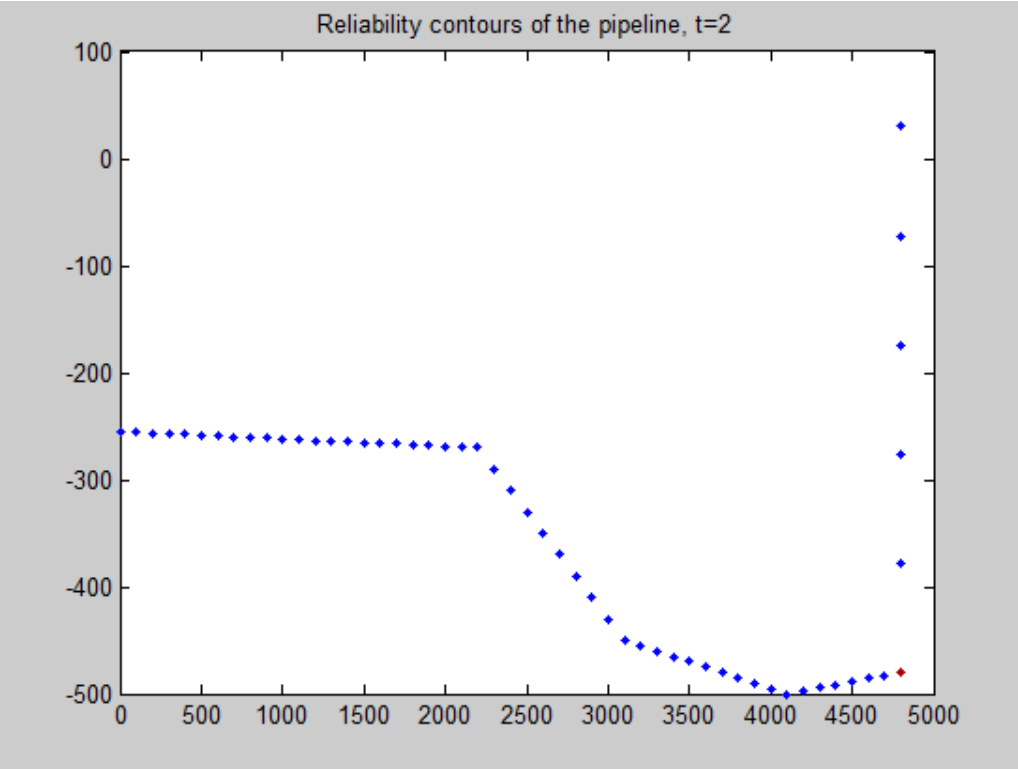


Figure 7.4. Reliability contours of the pipeline for t=2 years

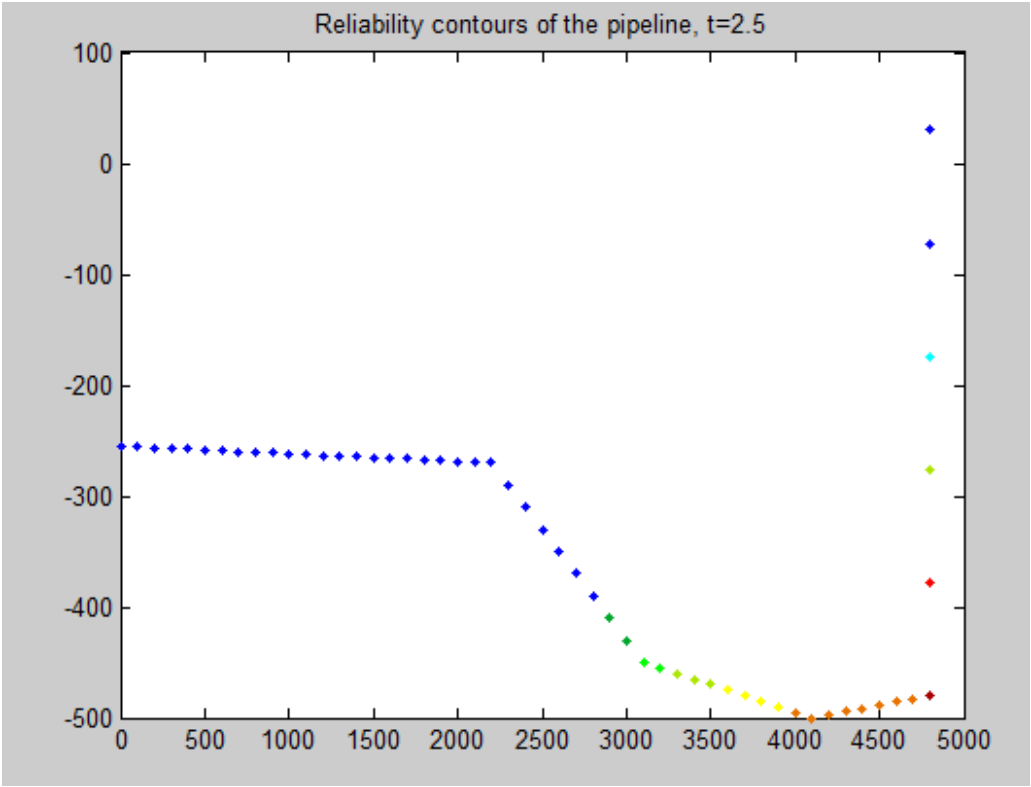


Figure 7.5. Reliability contours of the pipeline for t=2.5 years

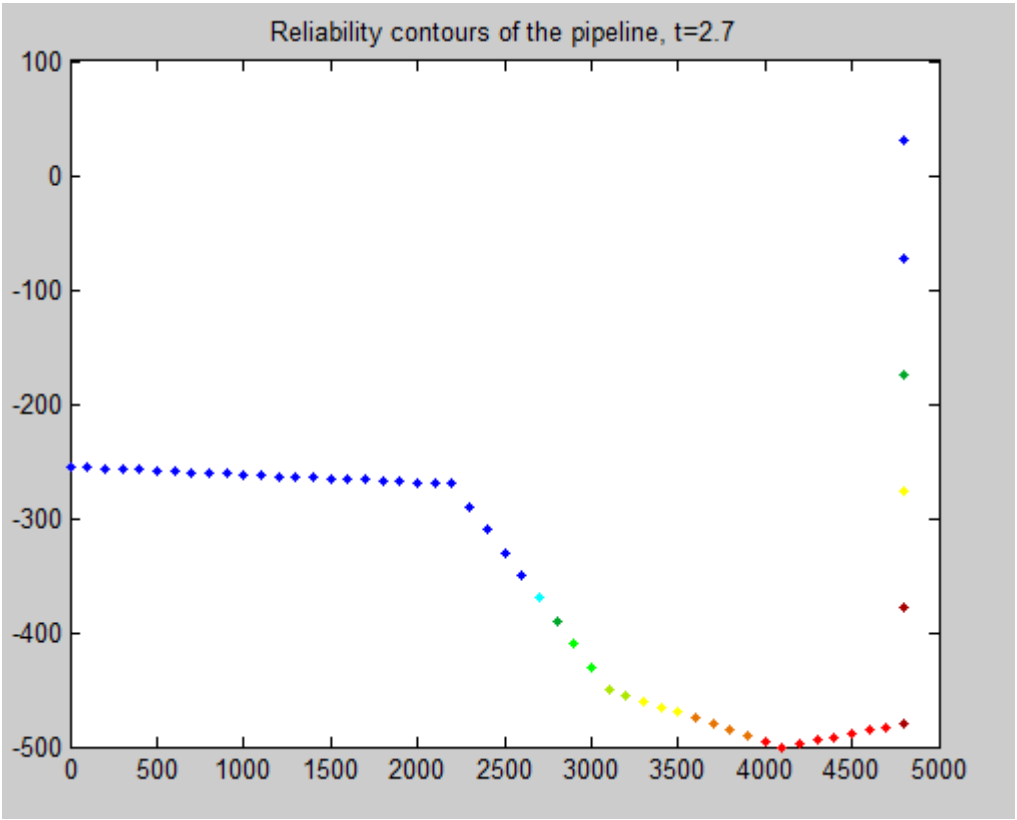


Figure 7.6. Reliability contours of the pipeline for t=2.7 years

Commenting the results displayed above we can say that they make perfectly sense because the highest corrosion rate is shown to happen for high pressures and the margin of safety is intuitively lower for the same pipe segments. In particular segments that have higher operating pressure conditions are between pipe segment #40 and #50.

Another interesting visualisation of the results can be given if only the exceedance of USL and SLS is considered. A figure to visualise this is produced for t=2.5 years only. The legend is the same used in Appendix D.

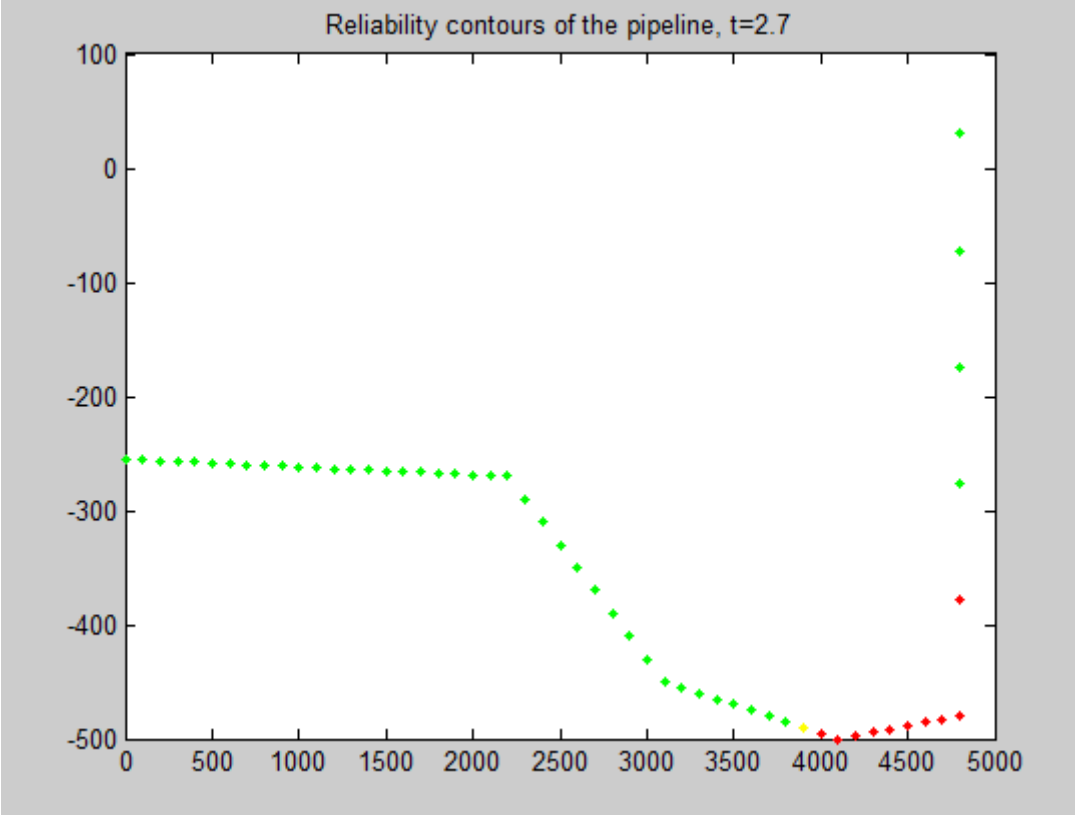


Figure 7.7. Safety Levels exceedance map for t=2.7 years

Considering only segment #49 which is the first that over-time shows safety levels below target SLS and ULS we can plot his behaviour over-time in Figure 7.8.

For some low probability of failure values we cannot represent the safety index in the scale considered. During years from 1 to 2 beta is so high (Pf so low) that we are not interested in representing it properly. Setting as a limit below which represent the safety index trend the beta value of 8 we realise that just after the second year this is overcome.

To confirm these results other techniques have been applied. As discussed previously Kriging-MCS and Dynamically Kriged LS are applied. Because of the relatively high number of segments and the low interest in investigating those who don't show failure first, further analysis techniques are just applied to the critical segment #49. The aim of this further investigation is to show the applicability of the methodologies also in a reliability analysis over-time.

Comparative results are reported in Figure 7.9.

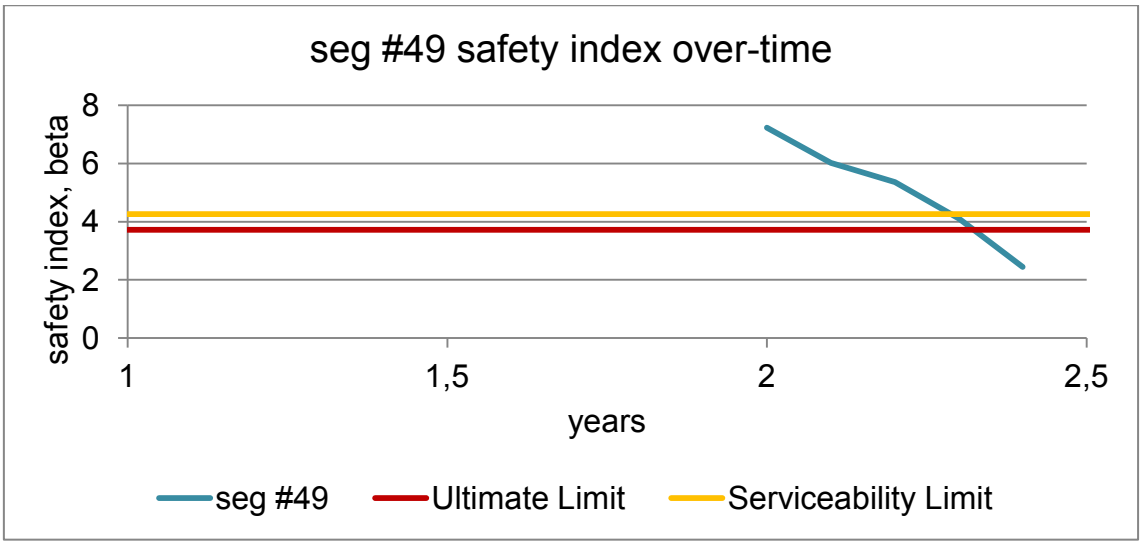


Figure 7.8. Safety index over-time for critical segment

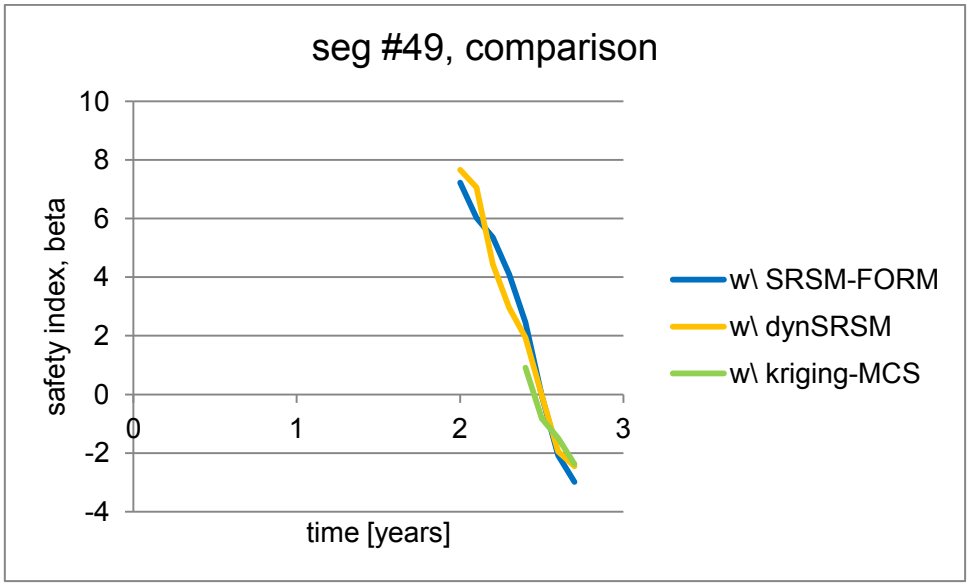


Figure 7.9. Methodologies comparison on critical segment safety index over-time

Setting some limits to the safety index we can predict when troubles may arise hence plan inspection and maintenance. What we would expect after such intervention is to pull the safety index back far above the safety thresholds.

One more thing that can be done without simulating for too many discrete events in time is to run simulations just for few time instants (at least 3) and interpolate them with a curve. What the analyst can expect is that this curve will accurately predict the trend over-time. This concept associated to maintenance planned according to predictive engineering is well shown in Figure 7.10.

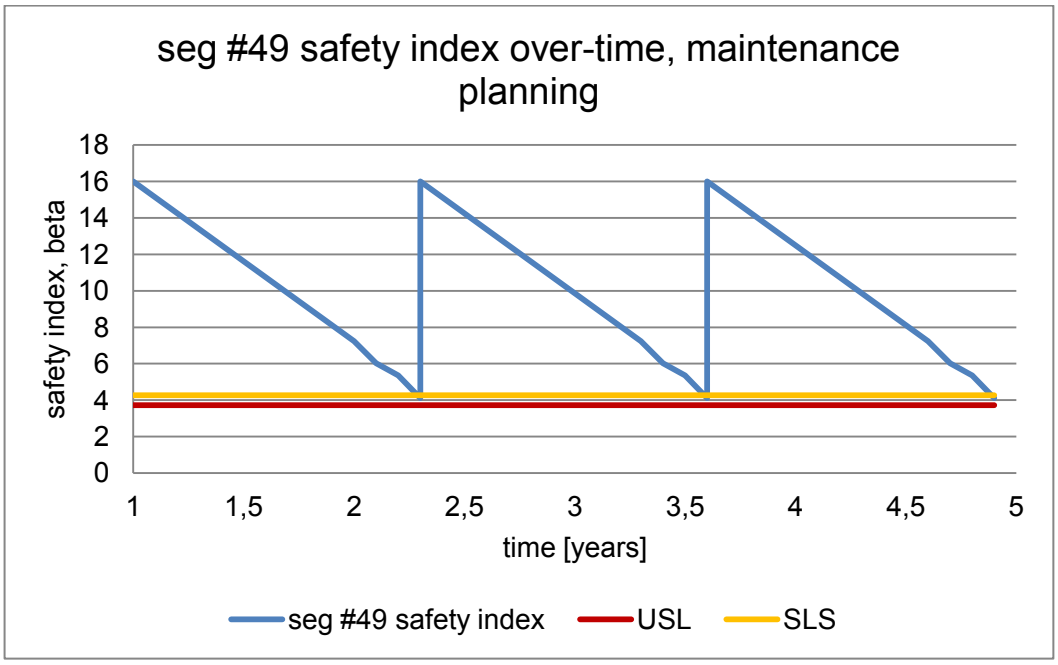


Figure 7.10. Maintenance planning effect, seg#49

Thanks to the spatial and time integration (such terms are inappropriate since every analysis has been carried out for different spatial points on different time instants, results have been then interpolated to have a continuous trend) we can have values of probability of failure for each segment of the geometry over time hence determine where and when failure will occur first.

An analysis on sensitivity factors (direction cosines of HL-FORM) can be done in order to spot out which ones have the largest impact on the system failure. Some pie charts at different discrete time are created to represent the relative contribution to failure and they are reported below. In the charts the value for the angles corresponding to the direction cosine is given. The direction cosine itself can vary between -1 and 1 and gives the steepness of the curve close to the MPP, hence how a variation in an input variable can move the MPP far from the actual one.

Analysing the charts we can assess that initially just yield and well pressure are affecting failure most, while other variables have minor impact. Time passing the well pressure is more and more participating in failure as a cause of it. Also fluid temperature and heat transfer coefficient start having large slices of the pie of failure. In particular, the last pie chart shows that temperature and heat transfer coefficient. This makes perfectly sense since the thickness is decreasing and it's not appearing as a considered variable. In the *MoS* what appears is the pressure allowable in the pipe, direct proportional to the thickness and inverse proportional to the fluid temperature.

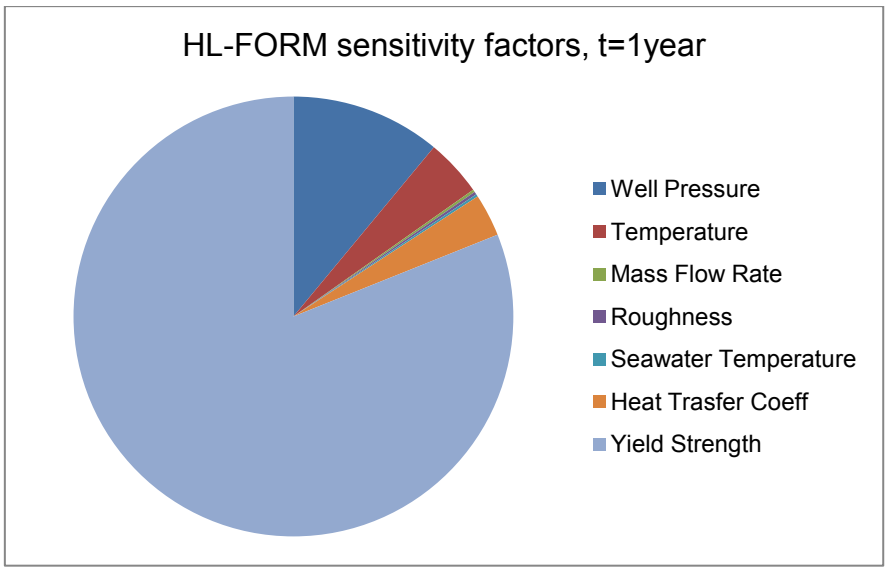


Figure 7.11. Sensitivity factors, t=1 year

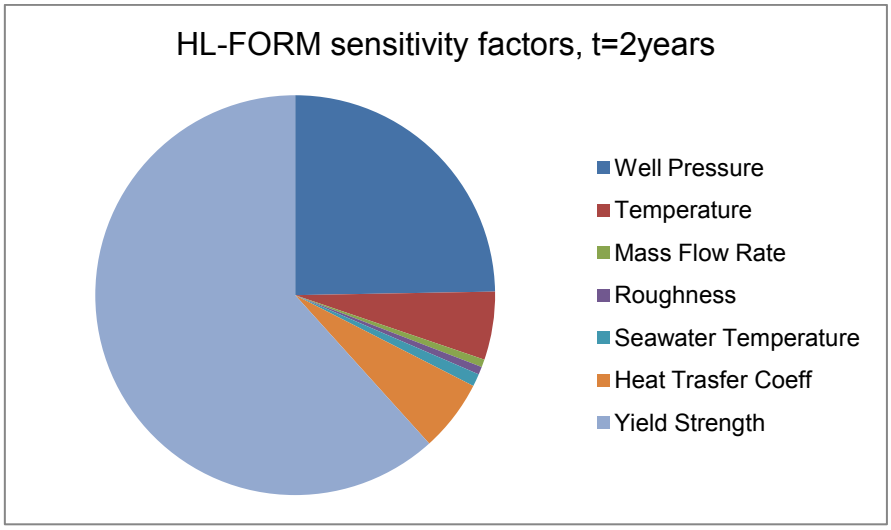


Figure 7.12. Sensitivity factors, t=2 years

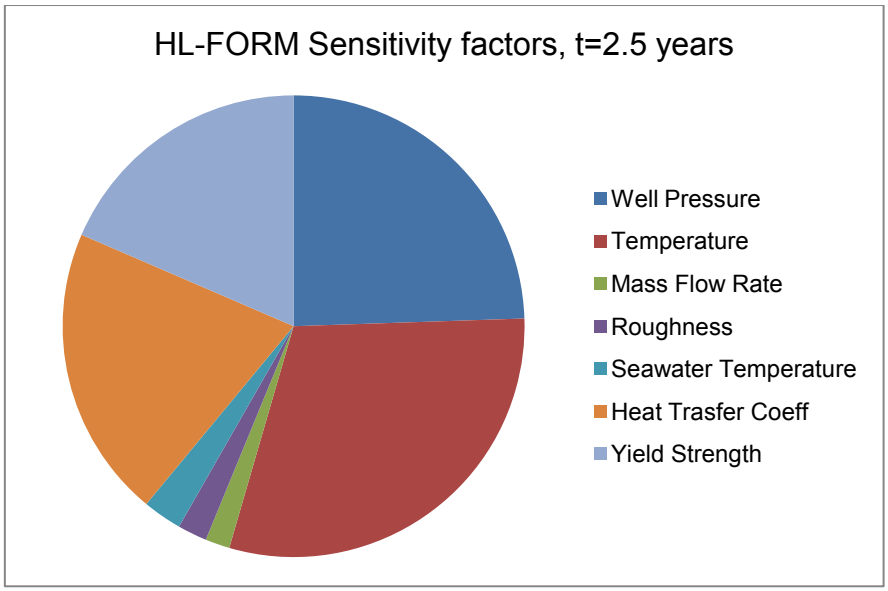
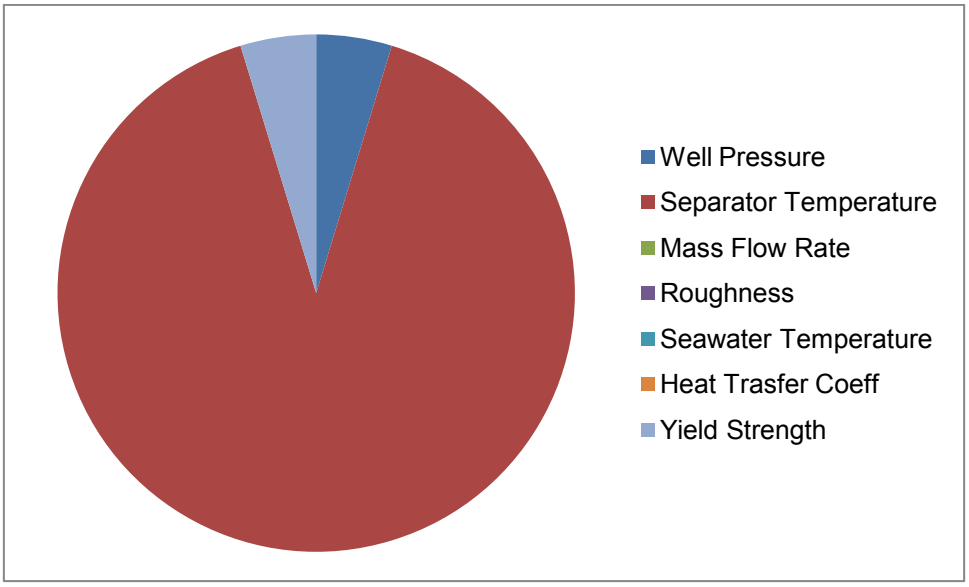


Figure 7.13. Sensitivity factors, t=2.5 years

An interesting comparison can be done with kriging activity parameter which just gives an idea of how much a variable is active on the output (MoS). In kriging no stochastic analysis is included hence no stochasticity of the variables is considered. What we obtain for the same time instants is shown in the graphs below. We can observe that the variable importance position is completely mixed compared with the ones given in the charts above. That's because no stochasticity is considered there hence the width of the distribution doesn't really matter. Meanwhile this largely matters in the failure of the system because the larger the distributions the more randomness is introduced in the process. Kriging parameters don't give us any information on the failure but only on the surrogate built.


Well Pressure
Separator Temperature
Mass Flow Rate
Roughness
Seawater Temperature
Heat Trasfer Coeff
Yield Strength

8 CONCLUSIONS

8.1 **Summary**

Since no quantitative probabilistic assessments have ever been applied to piping systems and no systematic analysis has ever been carried out, the present work attempted to cover this.

The analysis has been focused both on one-phase systems and two-phase oil-gas flow, where uncertainties, especially in harsh operating environments, are higher and their impact is critical. Moreover the risk related to failure in such conditions is extremely high because of the impact of the failure event itself on the environment.

Conventional tools, already employed in the structural reliability world, have been applied to pipe flow systems and, in order to perform a more accurate analysis, some new tools have been developed. Particularly surrogate modelling using kriging has been coupled to probabilistic analysis tools which included direct simulation by means of MCS and optimisation algorithm to seek for the MPP distance from the failure region (e.g. FORM-HL).

Due to the focus on corrosion and the many works present in literature where corrosion impact assessment is carried out, particular attention has been given to this aspect. Furthermore, a discrete events analysis of the system behaviour has been carried out. This leads to predictive engineering considerations about where and when the failure is going to happen. The accuracy of such prediction is subjected to the approximations introduced by system surrogate modelling and by probabilistic analysis, which are generally considered acceptable.

8.2 **Achievements**

Quantitative reliability assessment of pipe flow systems has been introduced and a systematic approach to analyse them has been developed.

Conventional reliability analysis tools have been transferred successfully from structural design world (where they have been first developed) to pipe flow system world.

Accurate and innovative analysis tools have been developed and proved to be applicable to pipe flow systems.

Both one-phase flow and two-phase flow systems have been analysed. In particular, corrosion impact on the latter has been assessed and a discrete events analysis has been performed.

8.3 **Future work**

The present work opens many possibilities of analysis regarding pipe flow systems and systems in general.

Firstly, the new analysis tools developed should be tested on other types of systems to show their broad applicability.

Further studies should be carried out to analyse a real system where different failure modes for pipe systems are investigated. Having real system geometries and field data,

from which to derive variables distributions and generalised corrosion models, a prediction estimate of the real behaviour of the system can be done. Moreover deeper and more realistic analysis can be done obtaining more meaningful results.

APPENDICES

Appendix A Models details

A.1 Model validation data

A.1.1 Pump data

A typical pump characteristic which is also varied throughout the validation procedure is given below. It is intended to serve just as a reference for the reader since the defining parameters are changed to obtain different validation responses.

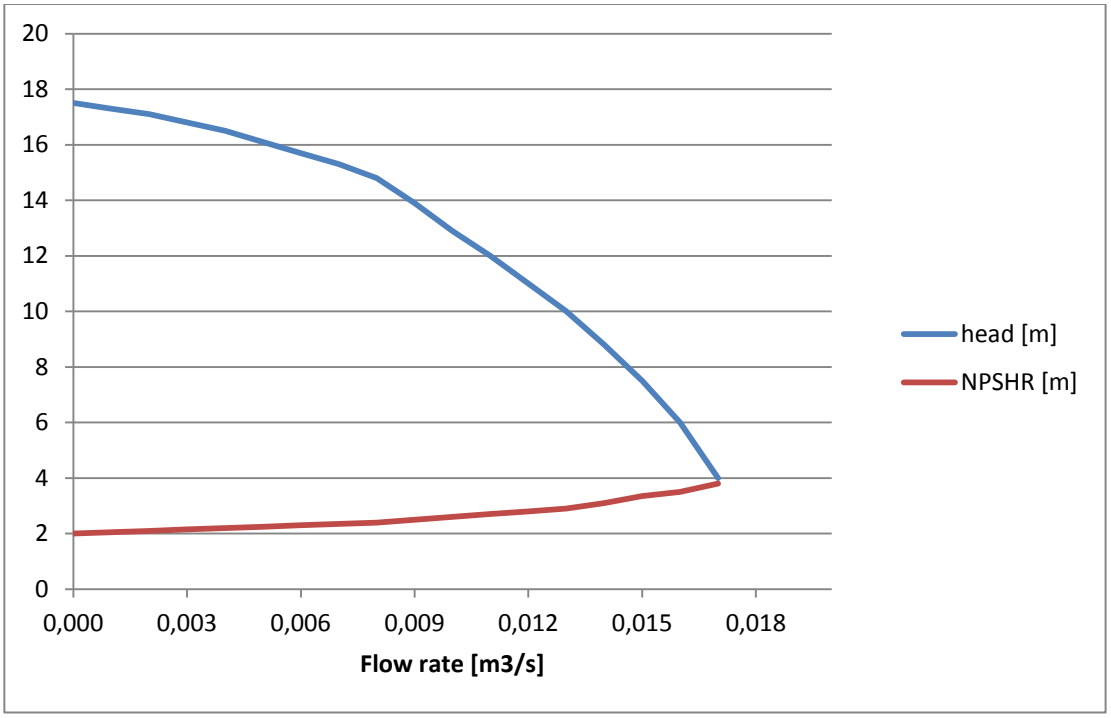


Figure A.1. Pump characteristic

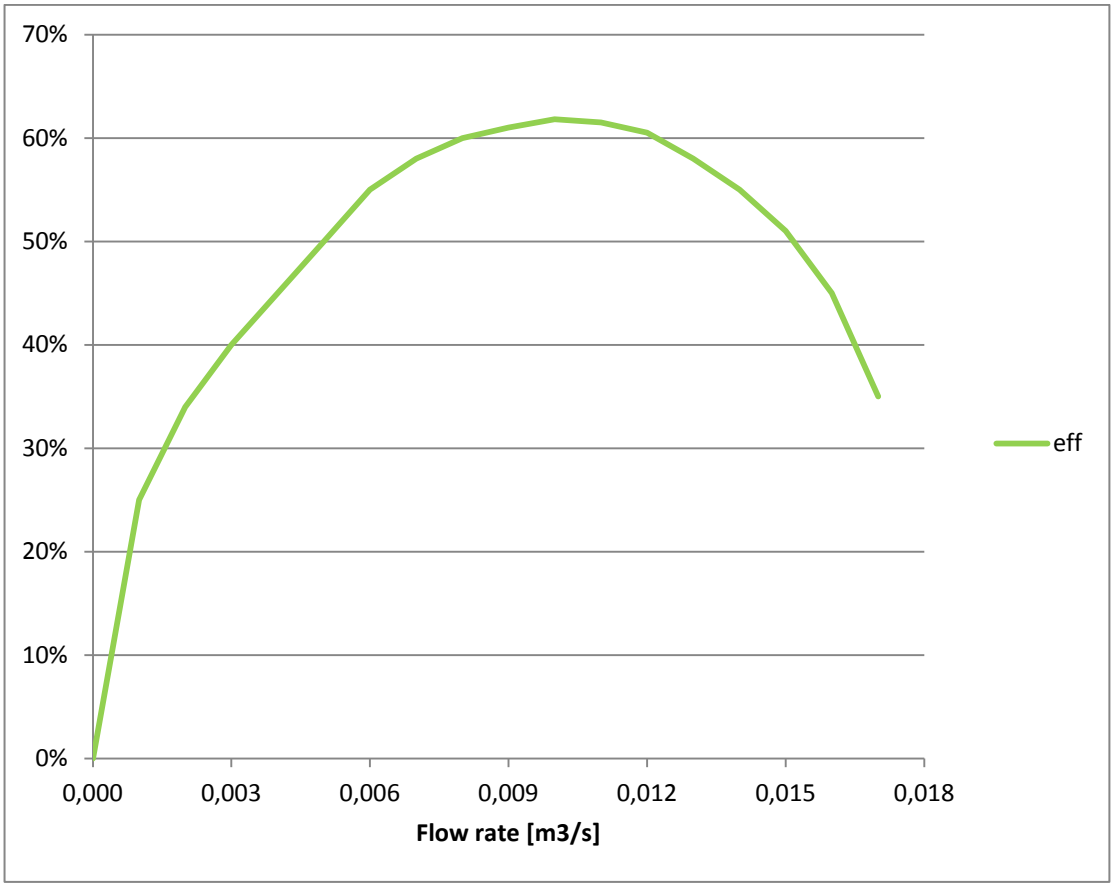


Figure A.2. Pump efficiency

This real curve has been approximated differently from time to time depending on the flow rate with an equation similar to the one below and just changing the coefficient *b*. For a flow rate of 0.010 m^3/s

$$H_{pump} = 17.5 - 46000\,\dot{V}^2$$

A.1.2 Validation data: *dp2fr.m*

Table A.1. Validation data for *dp2fr.m*

System	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5
L [m]	10.00	15.00	20.00	2.00	200.00
D [cm]	5.2502	7.7927	5.2502	2.6645	10.2260
ε [mm]	0.0460	0.0460	0.0460	0.0150	0.0460
h [m]	0.00	2.00	1.00	0.00	0.00
bends	[1,1,1]	[7,0,0]	[1,0,0]	[0,0,0]	[0,0,0]
valves	[1,1,1]	[1,1,0]	[0,0,1]	[0,0,0]	[0,0,0]
a	17.50	17.50	17.50	17.50	17.50
b	47,000	44,378.5	42,200	42,200	47,000
T [K]	293	293	293	293	293
Matlab					
V' [m3/s]	0.010	0.013	0.008	0.008	0.010
p_in [Pa]	400,000	300,000	300,000	400,000	200,000
p_out	376,530	325,310	361,800	404,660	298,350
PFE2010					
L_in [m]	40.87	30.653	30.653	40.87	20.435
L_out [m]	38.472	33.239	36.967	41.347	30.484
V' [m3/s]	0.010	0.013	0.008	0.008	0.010

e	***0%***	***0%***	***0%***	***0%***	***0%***

A.1.3 Validation data: *fr2pout.m*

Table A.2. Validation data for *fr2pout.m*

System	Scenario 1	Scenario 2	Scenario 3	Scenario 4	Scenario 5
L [m]	10.00	15.00	20.00	2.00	200.00
D [cm]	5.2502	7.7927	5.2502	2.6645	10.2260
ε [mm]	0.0460	0.0460	0.0460	0.0150	0.0460
h [m]	0.00	2.00	1.00	0.00	0.00
bends	[1,1,1]	[7,0,0]	[1,0,0]	[0,0,0]	[0,0,0]
valves	[1,1,1]	[1,1,0]	[0,0,1]	[0,0,0]	[0,0,0]
a	17.50	17.50	17.50	17.50	17.50
b	47,000	44,378.5	42,200	42,200	47,000
T [K]	293	293	293	293	293
Matlab					
p_out	376,530	325,310	361,800	404,660	298,350
p_in [Pa]	400,000	300,000	300,000	400,000	200,000
V' [m3/s]	0.010	0.013	0.008	0.008	0.010
PFE2010					
L_in [m]	40.87	30.653	30.653	40.87	20.435
L_out [m]	38.472	33.239	36.967	41.347	30.484
V' [m3/s]	0.010	0.013	0.008	0.008	0.010

e	***0%***	***0%***	***0%***	***0%***	***0%***

A.2 Systems geometry

A.2.1 One-phase system

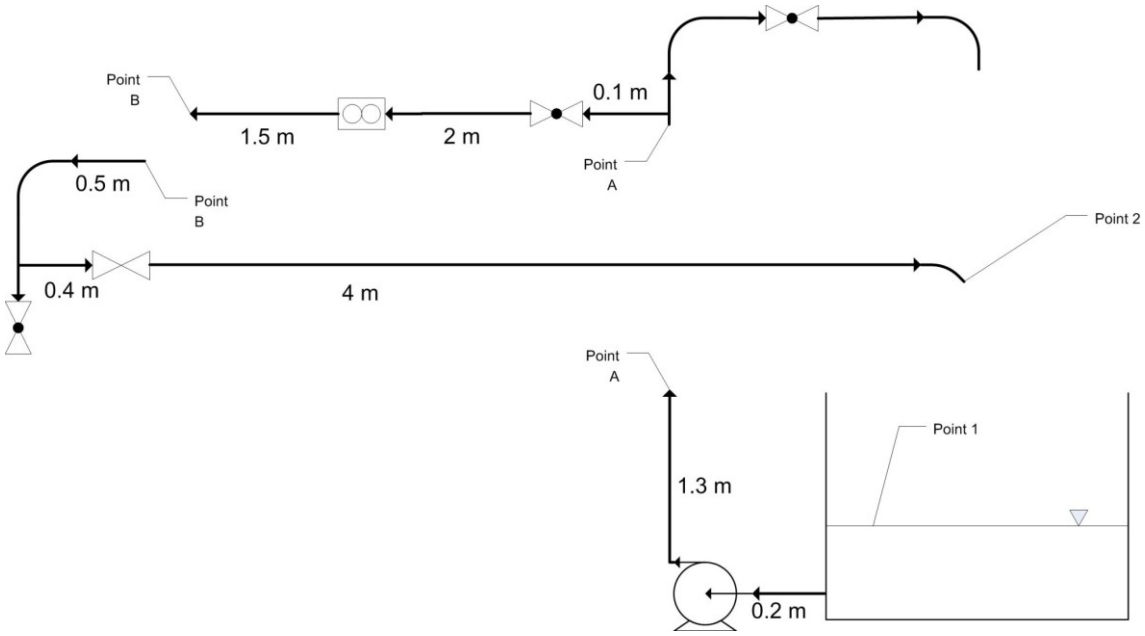


Figure A.3. One-phase system geometry

Table A.3. One-phase system operating data

Elevation [m]	1
Temperature [K]	293
Pipe length [m]	10
Diameter [cm]	5.2502
Pipe roughness [mm]	0.046
Inlet pressure [Pa]	101325
Outlet pressure [Pa]	101325
Pressure in A [Pa]	233960.3
Pressure in B [Pa]	141025.57
Friction coefficient	0.203
Flow rate [m3/s]	0.01

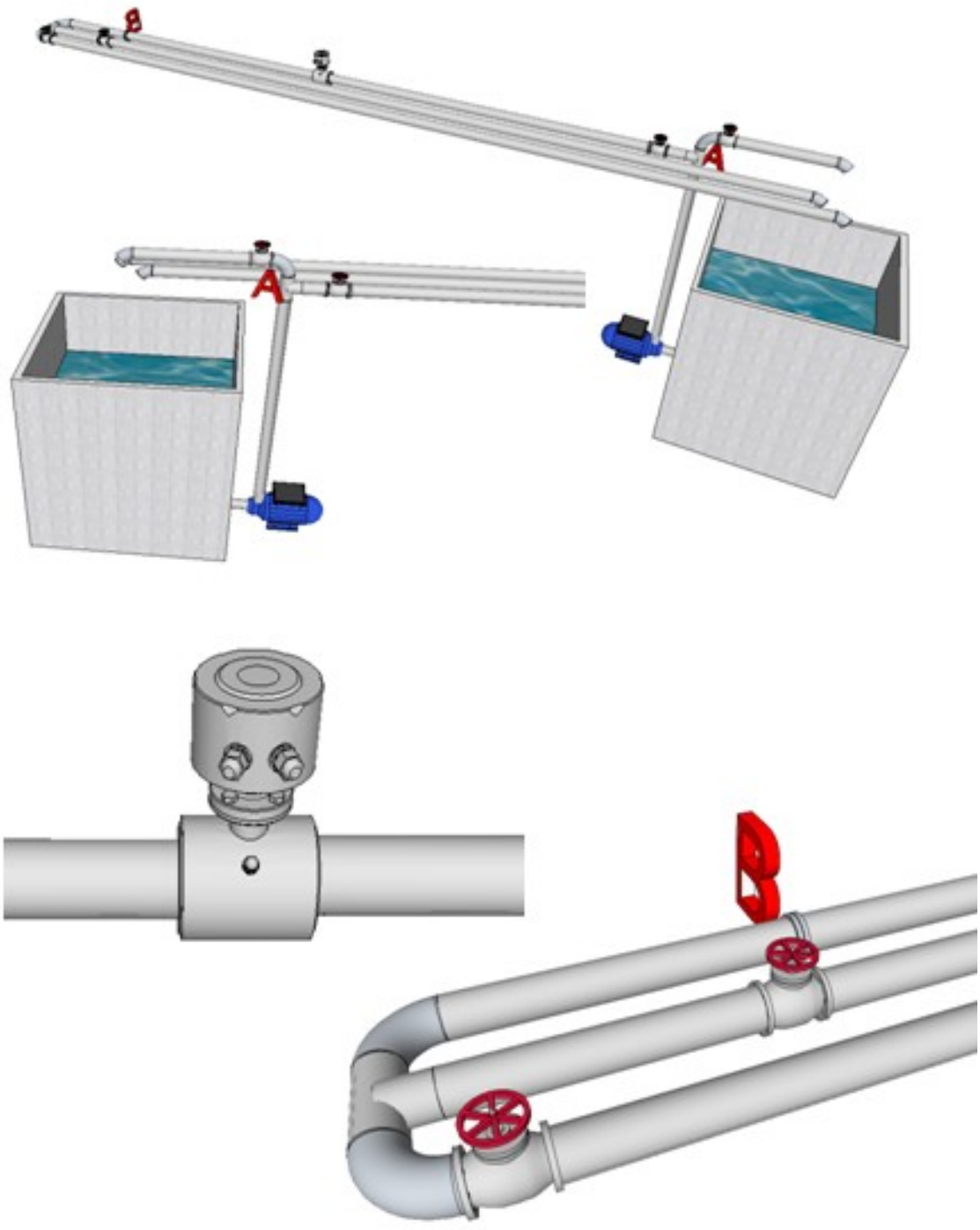


Figure A.4. One-phase system 3D snapshots

A.2.2 Two-phase system

Table A.4. Two-phase system geometrical data

Pipe	x [m]	y [m]	Length [m]	Elevation [m]	# Sections	Diameter [m]	Roughness [m]	Wall thickness [m]
Start Point	0	-255						
PIPE-1	2200	-270	2200,05	-15	22	0,12	2,80E-05	0,009
PIPE-2	3100	-450	917,824	-180	9	0,12	2,80E-05	0,009
PIPE-3	4100	-500	1001,25	-50	10	0,12	2,80E-05	0,009
PIPE-4	4800	-480	700,286	20	7	0,12	2,80E-05	0,009
PIPE-5	4800	30	510	510	5	0,1	2,80E-05	0,0075
PIPE-6	4900	30	100	0	1	0,1	2,80E-05	0,0075
			5429,41	285				

Appendix B Matlab codes

Some pieces of code used to perform Kriging were not developed by the author and they were just taken from http://www.southampton.ac.uk/~aijf197/academic.htm, where they are supplied as learning material from [15]. The codes employed in this work are listed below and they have been used and sometimes modified according to the attached GNU GPL license. No copyright infringement is intended and it belongs to the original authors Alexander IJ Forrester, András Sóbester and Andy Keane.

List of codes used:

- *ga.m*
- *bestlh.m*
- *jd.m*
- *mm.m*
- *mmlhs.m*
- *mmphi.m*
- *mmsort.m*
- *perturb.m*
- *reglikelihood.m*
- *regpredictor.m*
- *rlh.m*

B.1 Models

B.1.1 fr2pout.m

```
% calculates output pressure from a given (sub)system specified by
length of straight pipe [m], diameter [cm], pipe roughness [mm],
elevation [m], number of standard 90° bends, long radius 90° bends,
45° elbow bends, number of gate valves, globe valves and check valves,
pump characteristic a-bV^2. needs 'density.m', 'viscosity.m' to run
(to calculate respectively density and viscosity of the fluid).
operating conditions of the system are given by flow rate [m3/s],
inlet pressure [Pa] and temperature [K].

flag=0;

close all

clear all

clc

% data aquisition: geometry of the system

L=input('total straight pipe length [m]: ');

D=input('pipe diameter [cm]: ')/100;

eps=input('pipe roughness [mm]: ')/1000;

h=input('elevation (defined positive if z_out>z_in) [m]: ');

n_bends=input('total # bends: 90° standard, 90° long radius, 45°
elbows: ');

n_valves=input('total # valves: gate, globe, check valve: ');
```

```
comp_loss=input('other total component loss: ');
disp('pump characteristic, Hp=a-bQ^2 [m]');
a=input('a [m]:' );
b=input('b [s/m^2]: ');
% data aquisition: operating conditions
V=input('flowrate [m3/s]: ');
p_in=input('inlet pressure [Pa]: ');
t=input('temperature [K]: ');
% circuit solution %
As=D^2*3.1415/4;
V=v*As;
    %incompressible fluid assumption, to be checked later
rho=density(t,p_in);
mi=viscosity(t);
gamma=rho*9.81;
Re=rho*v*D/mi;
eD=eps/D;
    %purely turbulent flow assumption, to be checked later
f=colebrook_white(eD, Re);

H_distr=f*L*v^2/(2*9.81*D);
k_tot=n_bends(1)*0.57+n_bends(2)*0.30+n_bends(3)*0.30+n_valves(1)*0.15
+n_valves(2)*6.50+n_valves(3)*2.40+comp_loss;
H_loc=k_tot*v^2/(2*9.81);
p_out=gamma*(p_in/gamma-h+a-b*V^2-H_loc-H_distr);
    %check assumptions
if Re<4000
    flag=1;
end
rho1=density(t,p_out);
if abs((rho-rho1))/rho>0.01
    flag=1;
end
if flag==1
    disp('Assumption(s) not acceptable');
end
%presentation of results
disp('p_out=');
disp(p_out);
%% Daniele Casali_s162468_Cranfield University, MSc PSE, 2012
```

B.1.2 dp2fr.m

```
% calculates volumetric flow rate [m3/s] from a given (sub)system
specified by length of straight pipe [m], diameter [cm], pipe
roughness [mm], elevation [m], number of standard 90° bends, long
ragius 90° bends, 45° elbow bends, number of gate valves, globe valves
and check valves, pump characteristic a-bV^2. needs 'density.m',
'viscosity.m' to run (to calculate respectively density and viscosity
of the fluid). operating conditions of the system are given by inlet
and outlet pressure [Pa] and operating temperature [K]. (no heat loss
assumed)
flag=0;
close all
clear all
clc
% data aquisition: geometry of the system
L=input('total straight pipe length [m]: ');
D=input('pipe diameter [cm]: ')/100;
eps=input('pipe roughness [mm]: ')/1000;
h=input('elevation (defined positive if z_out>z_in) [m]: ');
n_bends=input('total # bends: 90° standard, 90° long radius, 45°
elbows: ');
n_valves=input('total # valves: gate, globe, check valve: ');
comp_loss=input('other total component loss: ');
disp('pump characteristic, Hp=a-bQ^2 [m]');
a=input('a [m]:' );
b=input('b [s/m^2]: ');
disp('pump efficiency characteristic, eff=cQ-dQ^2 %');
c=120;
d=5800;
% data aquisition: operating conditions
p_in=input('inlet pressure [Pa]: ');
p_out=input('outlet pressure [Pa]: ');
t=input('temperature [K]: ');


% circuit solution %
As=D^2*3.1415/4;
    %incompressible fluid assumption, to be checked later
rho=density(t,p_in);
mi=viscosity(t);
gamma=rho*9.81;
eD=eps/D;
```

```
f=1/(2*log10(eD/3.71))^2;
f_e=1;
    %purely turbulent flow assumption, to be checked later
while abs(f_e-f)/f_e>0.01
    f_e=f;

k_tot=n_bends(1)*0.57+n_bends(2)*0.30+n_bends(3)*0.30+n_valves(1)*0.15
+n_valves(2)*6.50+n_valves(3)*2.40+comp_loss;
    v=sqrt(((p_out-p_in)/gamma+h-a)/(-b*As^2-k_tot/(2*9.81)-
f_e*L/(2*9.81*D)));
    Re=rho*v*D/mi;
    f=colebrook_white(eD, Re);
end
V=v*As;
eff=c*V-d*V^2;
    %check assumptions
rho1=density(t,p_out);
if abs((rho-rho1))/rho>0.01
    flag=1;
end
if flag==1
     disp('Assumption(s) not acceptable');
end
%presentation of results
disp('V= ');
disp(V);
disp('eff= ');
disp(eff);
%% Daniele Casali_s162468_Cranfield University, MSc PSE, 2012
```

B.1.3 density.m

```
%calculates density of water in a range 0-100°C temperature and
applies a correction for pressure values. input in [K] and in [Pa].
output in[kg/m3]
function [rho]=density(t1,p1)
temp=[273.15; 278.15; 283.15; 293.15; 303.15; 313.15; 323.15; 333.15;
343.15; 353.15; 363.15; 373.15];
dens=[999.9; 1000; 999.7; 998; 995.2; 992.2; 988.1; 983.2; 977.8;
971.8; 965.3; 958.4];
array=[temp, dens];
for i=1:1:12,
```

```
    if abs(array(i, 1)-t1)<= 5, rho=array(i,2);
    end
end
%correction for pressure
if p1>1.013*10^5
    rho_c=rho/(1-((p1-1.013*10^5)/(2.15*10^9)));
    rho=rho_c;
elseif p1<1.013*10^5
    disp ('warning: pressure below p_atm at this point');
elseif p1==1.013*10^5
    disp('warning: p_atm at this point');
end
end
```

B.1.4 viscosity.m

```
%calculates viscosity of water in a range 0-100°C temperature and
applies a correction for pressure values. input in [K]. output in
%[Pa.s]. assume independent from pressure.
function [mi]=viscosity(t1)
temp=[273.15; 278.15; 283.15; 293.15; 303.15; 313.15; 323.15; 333.15;
343.15; 353.15; 363.15; 373.15];
visc=10^(-3)*[1.78;
1.52;1.31;1;0.798;0.653;0.547;0.467;0.404;0.355;0.314;0.281];
array=[temp, visc];
for i=1:1:12,
    if abs(array(i, 1)-t1)<= 5, mi=array(i,2);
    end
end
end
```

B.1.5 colebrook_white.m

```
function [f]=colebrook_white(eD,Re)
cw_eq=inline('1.0/sqrt(f) + 2.0*log10( eD/3.71 + 2.51/(Re*sqrt(f))
)',...
'f','eD','Re');
f_i=1/(-1.8*log10((eD/3.7)^1.11+6.9/Re))^2;
dfTol=5e-6;
f=fzero(cw_eq,f_i,optimset('TolX',dfTol,'Display','off'),eD,Re);
end
```

B.2 Reliability analysis

B.2.1 MCS.m

```
%direct MCS on scenario (explicit model)


%acquisition of random variables through a vector input, in this
order:
%limit states
function Pf=MCS(mi,sigma,y_lim)
n_max=input('maximum number of iterations: ');
nf=0;
tic
for i=1:n_max
    clc
    in(1)=normrnd(mi(1),sigma(1));
    in(2)=normrnd(mi(2),sigma(2));
    in(3)=normrnd(mi(3),sigma(3));
    in(4)=normrnd(mi(4),sigma(4));
    in(6)=normrnd(mi(6),sigma(6));
    y=scenario(in);
    if y>y_lim
        nf=nf+1;
    end

end
Pf=nf/n_max
toc
%%  Daniele Casali_s162468_Cranfield University, MSc PSE, 2011
```

B.2.2 SRSM.m

```
%needs FORM 1,2,3,4,5,6,7,8 variables and scenario simulation to run
%the code builds the simulation matrix, runs simulations, builds the
design
%matrix, calculates the vector of coefficients, and launches FORM
depending
%on the number of stochastic variables considered
```

```
function [beta, Pf]=SRSM(mi,sigma,lim,typelim)
tic
clc
clear all
close all
n_m=size(mi);
n_s=size(sigma);
if n_m~=n_s
    return
end
n=n_m;
if (n>8)
    return
end
%simulation matrix
S=zeros(2*n+1,n);
for i=1:2*n+1
    S(i,:)=mi;
end
for i=1:2*n+1
     for j=1:n
         if i==1
             S(i,j)=mi(j);
         elseif j+1==i && i<=n+1
             S(i,j)=mi(j)+3*sigma(j);
         elseif j+n+1==i && i>n+1
             S(i,j)= mi(j)-3*sigma(j);
         end
     end
 end
 %design matrix
 X=ones(2*n+1,2*n+1);
 i=1;
 j=1;
 for i=2:2*n+1
     if mod(i,2)==0
         X(:,i)=S(:,j);
     else
         X(:,i)=S(:,j).^2;
```

```
        j=j+1;
    end
end
%SRSM samples creation
Y=ones(2*n+1,1);
for i=1:2*n+1
    in=S(i,:);
    Y(i,1)=scenario(in);
end
a=((X'*X))\(X'*Y);
switch n
    case 1
beta=FORM_1(mi,sigma,a,lim,typelim);
    case 2
beta=FORM_2(mi,sigma,a,lim,typelim);
    case 3
beta=FORM_3(mi,sigma,a,lim,typelim);
    case 4
beta=FORM_4(mi,sigma,a,lim,typelim);
    case 5
beta=FORM_5(mi,sigma,a,lim,typelim);
    case 6
beta=FORM_6(mi,sigma,a,lim,typelim);
    case 7
beta=FORM_7(mi,sigma,a,lim,typelim);
    case 8
beta=FORM_8(mi,sigma,a,lim,typelim);
end
Pf=1-normcdf(beta);
toc
end
%% Daniele Casali_s162468_Cranfield University, MSc PSE, 2012
```

B.2.3 SRSM_FORM.m

```
%% Daniele Casali, 2012, Cranfield University
Script applied to the two-phase flow case study where direct
simulation through Matlab cannot be performed.
```

```
% before running the code specify: mi, sigma values. 'data.xlsx' is
the spreadsheets contained in the same working directory from which
the script acquires data. Two sheets are present in 'data.xlsx', 'X'
where the design matrix is contained and Y where the matrix of
responses 'Y' is reported. 'Y' is structured so that different 2k+1
cases responses are reported in different columns and by row we have
the response in all the integration points of the system.

Column vectors of probability of failure and safety index per each
segment are obtained as an output.

clc

clear all

tic

mi=[mean values];

sigma=[std deviation values];

X=xlsread('data.xlsx', 'X');

Y=xlsread('data.xlsx', 'Y');

Y=Y';

[n_s,n_el]=size(Y);

n_v=(n_s-1)/2;

beta=zeros(n_el,1);

Pf=zeros(n_el,1);

for i=1:n_el

    y=Y(:,i);

    a=(X'*X)\(X'*y);

    beta(i,1)=FORM_7(mi,sigma,a);

    Pf(i,1)=1-normcdf(beta(i,1));

end

toc

%% Daniele Casali, 2012, Cranfield University
```

B.2.4 FORM_7.m

Just the 7 variables case is reported here to show how the algorithm works and how the author has implemented it. The same code can be easily written for more or less variables maintaining the same structure.

```
%FORM/HL for 6 variables case, just this one is reported here to show
the algorithm and how it has been implemented by the author
% if upper limit typelim=1, else lower limit
function [beta]=FORM_7(mi,sigma,a,lim,typelim)
tic
syms x1 x2 x3 x4 x5 x6 x7;
L=[x1 x2 x3 x4 x5 x6 x7];
% 1 - definition of the limit state function
if typelim==1
    g=lim-
(a(1)+a(2)*x1+a(3)*x1^2+a(4)*x2+a(5)*x2^2+a(6)*x3+a(7)*x3^2+a(8)*x4+a(
9)*x4^2+a(10)*x5+a(11)*x5^2+a(12)*x6+a(13)*x6^2+a(14)*x7+a(15)*x7^2);
else
g=(a(1)+a(2)*x1+a(3)*x1^2+a(4)*x2+a(5)*x2^2+a(6)*x3+a(7)*x3^2+a(8)*x4+
a(9)*x4^2+a(10)*x5+a(11)*x5^2+a(12)*x6+a(13)*x6^2+a(14)*x7+a(15)*x7^2
)-lim;
end
eta=1;
        iter=1;
        while eta>=0.001;
            if iter==1
        g_v=[double(subs(diff(g,L(1)),L,mi))
double(subs(diff(g,L(2)),L,mi)) double(subs(diff(g,L(3)),L,mi))
double(subs(diff(g,L(4)),L,mi)) double(subs(diff(g,L(5)),L,mi))
double(subs(diff(g,L(6)),L,mi)) double(subs(diff(g,L(7)),L,mi))
double(subs(g,L,mi))];
```

```
squaroot=sqrt((g_v(1)*sigma(1))^2+(g_v(2)*sigma(2))^2+(g_v(3)*sigma(3)
)^2+(g_v(4)*sigma(4))^2+(g_v(5)*sigma(5))^2+(g_v(6)*sigma(6))^2+(g_v(7
)*sigma(7))^2);

        beta=g_v(8)/squaroot;

        eta=1;

            else

        g_v=[double(subs(diff(g,L(1)),L,X))
double(subs(diff(g,L(2)),L,X)) double(subs(diff(g,L(3)),L,X))
double(subs(diff(g,L(4)),L,X)) double(subs(diff(g,L(5)),L,X))
double(subs(diff(g,L(6)),L,X)) double(subs(diff(g,L(7)),L,X))
double(subs(g, L, X))];


squaroot=sqrt((g_v(1)*sigma(1))^2+(g_v(2)*sigma(2))^2+(g_v(3)*sigma(3)
)^2+(g_v(4)*sigma(4))^2+(g_v(5)*sigma(5))^2+(g_v(6)*sigma(6))^2+(g_v(7
)*sigma(7))^2);

        beta_p=beta;

        beta=(g_v(8)-
(g_v(1)*sigma(1)*u(1)+g_v(2)*sigma(2)*u(2)+g_v(3)*sigma(3)*u(3)+g_v(4)
*sigma(4)*u(4)+g_v(5)*sigma(5)*u(5)+g_v(6)*sigma(6)*u(6)+g_v(7)*sigma(
7)*u(7)))/sqrt((g_v(1)*sigma(1))^2+(g_v(2)*sigma(2))^2+(g_v(3)*sigma(3
))^2+(g_v(4)*sigma(4))^2+(g_v(5)*sigma(5))^2+(g_v(6)*sigma(6))^2+(g_v(
7)*sigma(7))^2);

        eta=abs(beta_p-beta)/abs(beta_p);

            end


         alfa=[-sigma(1)*g_v(1)/squaroot -sigma(2)*g_v(2)/squaroot -
sigma(3)*g_v(3)/squaroot -sigma(4)*g_v(4)/squaroot -
sigma(5)*g_v(5)/squaroot -sigma(6)*g_v(6)/squaroot -
sigma(7)*g_v(7)/squaroot];

         X=[mi(1)+beta*sigma(1)*alfa(1) mi(2)+beta*sigma(2)*alfa(2)
mi(3)+beta*sigma(3)*alfa(3) mi(4)+beta*sigma(4)*alfa(4)
mi(5)+beta*sigma(5)*alfa(5) mi(6)+beta*sigma(6)*alfa(6)
mi(7)+beta*sigma(7)*alfa(7)];

         u=[(X(1)-mi(1))/sigma(1) (X(2)-mi(2))/sigma(2) (X(3)-
mi(3))/sigma(3) (X(4)-mi(4))/sigma(4) (X(5)-mi(5))/sigma(5) (X(6)-
mi(6))/sigma(6) (X(7)-mi(7))/sigma(7)];

         iter=iter+1;

        end

Pf=1-normcdf(beta)

toc

    end


%%  Daniele Casali_s162468_Cranfield University, MSc PSE, 2011
```

B.2.5 SRSM_MCS.m

```
%MCS on scenario, its approximation using SRSM quadratic polynomial
function Pf=SRSM_MCS(mi,sigma,lim, typelim)
n_m=size(mi);
n_s=size(sigma);
if n_m~=n_s
    return
end
n=n_m;
if (n>8)
    return
end
n_max=input('maximum number of iterations: ');
tic
%simulation matrix
S=zeros(2*n+1,n);
for i=1:2*n+1
    S(i,:)=mi;
end
for i=1:2*n+1
    for j=1:n
```

```
        if i==1
            S(i,j)=mi(j);
        elseif j+1==i && i<=n+1
            S(i,j)=mi(j)+3*sigma(j);
        elseif j+n+1==i && i>n+1
            S(i,j)= mi(j)-3*sigma(j);
        end
    end
end
%design matrix
X=ones(2*n+1,2*n+1);
i=1;
j=1;
for i=2:2*n+1
    if mod(i,2)==0
        X(:,i)=S(:,j);
    else
        X(:,i)=S(:,j).^2;
        j=j+1;
    end
end
%SRSM samples creation
Y=ones(2*n+1,1);
for i=1:2*n+1
    in=S(i,:);
    Y(i,1)=scenario(in);
end
a=((X'*X))\(X'*Y);
% 3 - definition of the limit state function
nf=0;
tic
for i=1:n_max
    x(1)=normrnd(mi(1),sigma(1));
    x(2)=normrnd(mi(2),sigma(2));
    x(3)=normrnd(mi(3),sigma(3));
    x(4)=normrnd(mi(4),sigma(4));
    x(5)=normrnd(mi(5),sigma(5));
    x(6)=normrnd(mi(6),sigma(6));
    if typelim==1
```

```
    g=lim-
(a(1)+a(2)*x(1)+a(3)*x(1)^2+a(4)*x(2)+a(5)*x(2)^2+a(6)*x(3)+a(7)*x(3)^
2+a(8)*x(4)+a(9)*x(4)^2+a(10)*x(5)+a(11)*x(5)^2+a(12)*x(6)+a(13)*x(6)^
2+a(14)*x(7)+a(15)*x(7)^2);
else

g=(a(1)+a(2)*x(1)+a(3)*x(1)^2+a(4)*x(2)+a(5)*x(2)^2+a(6)*x(3)+a(7)*x(3
)^2+a(8)*x(4)+a(9)*x(4)^2+a(10)*x(5)+a(11)*x(5)^2+a(12)*x(6)+a(13)*x(6
)^2+a(14)*x(7)+a(15)*x(7)^2)-lim;
end
if g<0
    nf=nf+1;
end
end
Pf=nf/n_max
toc
```

B.2.6 KriMCS.m

```
% kriging + MCS
function Pf=KriMCS(mi,sigma,lim,typelim)
clear all
clc
%sample points, LHS
k=input('number of design variables: ');
n=input('number of samples: ');
n_max=input('number of MCS iterations: ');
tic
%samples creation
S=bestlh(n,k,100,40);
Ss=zeros(size(S));
y=zeros(n,1);
% samples positions, cardinality for scaling
binsize=1/n;
pos=ones(size(S));
for i=1:k
    for j=1:n
        for l=1:n
    lowlim=binsize*(l-1);
     uplim=binsize*l;
         if lowlim<S(j,i)&& S(j,i)<=uplim
```

```
        pos(j,i)=1;
        end
        end
    end
end

% variables scaling
    % norminv on scaled probability matrix
    NrmInv_Pm=zeros(size(S));
for i=1:k
    for j=1:n
        NrmInv_Pm(j,i)=norminv(S(j,i)/n+(pos(j,i)-1)/n);
        if isinf(NrmInv_Pm(j,i))==1 && S(j,i)==0
            NrmInv_Pm(j,i)=-5;
        end
        if isinf(NrmInv_Pm(j,i))==1 && S(j,i)==1
            NrmInv_Pm(j,i)=5;
        end
    end
end
    % scaled samples matrix creation
for i=1:k
    for j=1:n
Ss(j,i)=mi(i)+sigma(i)*NrmInv_Pm(j,i);
    end
end
%generation of sample responses
for i=1:n
    inputs=Ss(i,:);
    y(i,1)=scenario1A(inputs); %insert scenario number!!!
end
%Kriging correlation
global ModelInfo
ModelInfo.X=Ss;
ModelInfo.y=y;
% Start iterating infill points
for I=1:4
% Tune regressing Kriging model
```

```
UpLim=ones(1,k)*4;
LowLim=ones(1,k)*-25;
UpLim(end+1)=0;
LowLim(end+1)=-6;
[Params,MaxLikelihood]=ga(@reglikelihood,k+1,[],[],[],[],LowLim,UpLim)
;
% Extract model parameters
ModelInfo.Theta(1:k)=Params(1:k);
ModelInfo.Lambda=Params(end);
% Put Choleski factorization of Psi
% into ModelInfo
[NegLnLike,ModelInfo.Psi,ModelInfo.U,ModelInfo.mu]=reglikelihood([Mode
lInfo.Theta ModelInfo.Lambda]);
ModelInfo.Option='Pred';
end
% Kriging prediction
randX=zeros(1,k);
nf=0;
% MCS iteration, every iteration a new prediction at random points.
for j=1:n_max
    %random inputs creation
    for i=1:k
    randX(i)=normrnd(mi(i),sigma(i));
    end
    pred_y(i)=regpredictor(randX);
    if pred_y>limit
        nf=nf+1;
    end
end
Pf=nf/n_max
toc
```

B.2.7 KriMCS_2ph.m

```
% kriging and MCS script for the two-phase system where simulation
data have to be acquired. 'data.xlsx' contains in sheet 'Ss' the
scaled sampling points from which simulation responses are obtained.
Simulation responses are post processed to obtain MoS values for each
segment using MoS.m.
```

```
tic
clear all
clc
N=input('number of MC iterations: ');
[MoS,p_all]=MoS();
MoS=MoS';
mi=[mean values];
sigma=[std deviation values];
[n_s,n_el]=size(MoS);
k=7; %number of variables, uncorrelated from number of samples now
Ss=xlsread('data.xlsx','Ss','A1:G15');
%Ss=xlsread('data20.xlsx','Ss','A1:G20');
%Ss=xlsread('data40.xlsx','Ss','A1:G40');
%Ss=xlsread('data60.xlsx','Ss','A1:G60');
 %   for counter=1:n_el
 for counter=49
        y=MoS(:,counter);
        %Kriging correlation
        global ModelInfo
        ModelInfo.X=Ss;
        ModelInfo.y=y;
       % Start iterating infill points
       for I=1:4
       % Tune regressing Kriging model
       UpLim=ones(1,k)*2;
       LowLim=ones(1,k)*-25;
       UpLim(end+1)=0;
       LowLim(end+1)=-15;

[Params,MaxLikelihood]=ga(@reglikelihood,k+1,[],[],[],[],LowLim,UpLim)
;
       % Extract model parameters
       ModelInfo.Theta(1:k)=Params(1:k);
       ModelInfo.Lambda=Params(end);
       % Put Choleski factorization of Psi
       % into ModelInfo

%[NegLnLike,ModelInfo.Psi,ModelInfo.U,ModelInfo.mu]=reglikelihood([Mod
elInfo.Theta ModelInfo.Lambda]);
```

```
[NegLnLike,ModelInfo.Psi,ModelInfo.mu]=reglikelihood([ModelInfo.Theta
ModelInfo.Lambda]);
        end



        %kriging LS definition
        lambda=ModelInfo.Lambda;
        Psi=ModelInfo.Psi;
        Psi=inv(Psi);
        x=ModelInfo.X;
        y=ModelInfo.y;
        mu=(ones(n_s,1)'/Psi*y)/(ones(n_s,1)'/Psi*ones(n_s,1));
        theta=10.^ModelInfo.Theta;
        Psi=inv(Psi);
        [n_s,k]=size(x);
        T=zeros(n_s,1);
        for i=1:n_s
            for j=1:n_s
            T(i,1)=Psi(i,j)*(y(j,1)-mu)+T(i,1);
    end
end
syms x1 x2 x3 x4 x5 x6 x7 psi
L=[x1 x2 x3 x4 x5 x6 x7];

for i=1:n_s
    s=0;
    for j=1:k
        s=theta(j)*(abs(x(i,j)-L(j)))^2+s;
    end
    psi(i,1)=exp(-s);
end
C=psi'*T;
g=mu+C;
% MCS algorithm
clc
Nf=0;
for p=1:N
```

```
        for i=1:k
            randX(i)=normrnd(mi(i),sigma(i));
        end
        predg=subs(g,L,randX);
        if predg<0
            Nf=Nf+1;
        end
    end
    Pf(counter,1)=Nf/N;
end
toc
%%  Daniele Casali_s162468_Cranfield University, MSc PSE, 2011
```

B.2.8 MoS.m

```
% post processing of data from simulations, output values are
converted to MoS values because stochasticity is also present in the
output limit value.


function [Mos,p_all]=MoS()
clear
clc
yield=xlsread('data.xlsx','Ss','G1:G15');
%yield=xlsread('data20.xlsx','Ss','G1:G60');
%yield=xlsread('data40.xlsx','Ss','G1:G60');
%yield=xlsread('data60.xlsx','Ss','G1:G60');
M=xlsread('data.xlsx','out','D3:AG56');
%M=xlsread('data20.xlsx','out','D3:AQ56');
%M=xlsread('data40.xlsx','out','D3:CE56');
%M=xlsread('data60.xlsx','out','D3:DS56');
nCO2=0.0145;
riserbase=49;
year=2.3;
[n_el,col]=size(M);
n_s=col/2;
k=1;
for j=1:2:col
    Y(:,1)=M(:,j);%pressure
    Y(:,2)=M(:,j+1);%temperature
```

```
for i=1:n_el
    vc=exp(5.8-1710/(273+Y(i,2))+0.67*log10(nCO2*Y(i,1)));
    cd=vc*year;
    if i<riserbase
        D=0.12;
        tp0=0.009;
    else
        D=0.1;
        tp0=0.0075;
    end
    tp=tp0-cd/1000;
    p_all(i,k)=2*yield(k)*10*tp*0.72/D;
    Mos(i,k)=p_all(i,k)-Y(i,1);
end
k=k+1;
end
end
```

B.2.9 KriFORM.m

```
% before running check: stoch variables#, mi and sigma arrays, lim and
% typelim, scenario#, sym variables

% kriging correlation analytically implemented as FORM limit state
function Pf=KriFORM(mi,sigma,lim,typelim)
clear all
clc

%inputs
k=input('number of design variables: ');
n=input('number of samples: ');
tic
%samples creation, LHS
S=bestlh(n,k,100,40);
%samples scaling
Ss=zeros(size(S));
y=zeros(n,1);
% samples positions, cardinality for scaling
```

```
binsize=1/n;
pos=ones(size(S));
for i=1:k
    for j=1:n
        for l=1:n
    lowlim=binsize*(l-1);
    uplim=binsize*l;
        if lowlim<S(j,i)&& S(j,i)<=uplim
        pos(j,i)=l;
        end
        end
    end
end

% variables scaling
    % norminv on scaled probability matrix
    NrmInv_Pm=zeros(size(S));
for i=1:k
    for j=1:n
        NrmInv_Pm(j,i)=norminv(S(j,i)/n+(pos(j,i)-1)/n);
        if isinf(NrmInv_Pm(j,i))==1 && S(j,i)==0
            NrmInv_Pm(j,i)=-5;
         end
         if isinf(NrmInv_Pm(j,i))==1 && S(j,i)==1
            NrmInv_Pm(j,i)=5;
         end
     end
 end
     % scaled samples matrix creation
 for i=1:k
     for j=1:n
 Ss(j,i)=mi(i)+sigma(i)*NrmInv_Pm(j,i);
     end
 end
 for i=1:n
     inputs=Ss(i,:);
     y(i,1)=scenario(inputs); %insert scenario number!!!
 end
```

```
%Kriging correlation
global ModelInfo
ModelInfo.X=Ss;
ModelInfo.y=y;
% Start iterating infill points
for I=1:4
% Tune regressing Kriging model
UpLim=ones(1,k)*2;
LowLim=ones(1,k)*-25;
UpLim(end+1)=0;
LowLim(end+1)=-15;
[Params,MaxLikelihood]=ga(@reglikelihood,k+1,[],[],[],[],LowLim,UpLim)
;
% Extract model parameters
ModelInfo.Theta(1:k)=Params(1:k);
ModelInfo.Lambda=Params(end);
% Put Choleski factorization of Psi
% into ModelInfo
[NegLnLike,ModelInfo.Psi,ModelInfo.U,ModelInfo.mu]=reglikelihood([Mode
lInfo.Theta ModelInfo.Lambda]);
end
```

```
%kriging LS definition
lambda=ModelInfo.Lambda;
Psi=ModelInfo.Psi;
Psi=inv(Psi);
x=ModelInfo.X;
y=ModelInfo.y;
mu=(ones(n,1)'/Psi*y)/(ones(n,1)'/Psi*ones(n,1));
theta=10.^ModelInfo.Theta;
Psi=inv(Psi);
[n,k]=size(x);
T=zeros(n,1);
for i=1:n
    for j=1:n
```

```
    T(i,1)=Psi(i,j)*(y(j,1)-mu)+T(i,1);
    end
end
syms x1 x2 x3 x4 x5 x6 x7 psi
L=[x1 x2 x3 x4 x5 x6 x7];

for i=1:n
    s=0;
    for j=1:k
        s=theta(j)*(abs(x(i,j)-L(j)))^2+s;
    end
    psi(i,1)=exp(-s);
end
C=psi'*T;



if typelim==1
    g=lim-(mu+C);
else
    g=(mu+C)-lim;
end
% HL algorithm
clc
eta=1;
        iter=1;
        while eta>=0.001;
            if iter==1
        g_v=[double(subs(diff(g,L(1)),L,mi))
double(subs(diff(g,L(2)),L,mi)) double(subs(diff(g,L(3)),L,mi))
double(subs(diff(g,L(4)),L,mi)) double(subs(diff(g,L(5)),L,mi))
double(subs(diff(g,L(6)),L,mi)) double(subs(diff(g,L(7)),L,mi))
double(subs(g,L,mi))];

squaroot=sqrt((g_v(1)*sigma(1))^2+(g_v(2)*sigma(2))^2+(g_v(3)*sigma(3)
)^2+(g_v(4)*sigma(4))^2+(g_v(5)*sigma(5))^2+(g_v(6)*sigma(6))^2+(g_v(7
)*sigma(7))^2);
        beta=g_v(8)/squaroot;
        eta=1;
            else
        g_v=[double(subs(diff(g,L(1)),L,X))
double(subs(diff(g,L(2)),L,X)) double(subs(diff(g,L(3)),L,X))
```

```
double(subs(diff(g,L(4)),L,X)) double(subs(diff(g,L(5)),L,X))
double(subs(diff(g,L(6)),L,X)) double(subs(diff(g,L(7)),L,X))
double(subs(g, L, X))];


squaroot=sqrt((g_v(1)*sigma(1))^2+(g_v(2)*sigma(2))^2+(g_v(3)*sigma(3)
)^2+(g_v(4)*sigma(4))^2+(g_v(5)*sigma(5))^2+(g_v(6)*sigma(6))^2+(g_v(7
)*sigma(7))^2);

        beta_p=beta;

        beta=(g_v(8)-
(g_v(1)*sigma(1)*u(1)+g_v(2)*sigma(2)*u(2)+g_v(3)*sigma(3)*u(3)+g_v(4)
*sigma(4)*u(4)+g_v(5)*sigma(5)*u(5)+g_v(6)*sigma(6)*u(6)+g_v(7)*sigma(
7)*u(7)))/sqrt((g_v(1)*sigma(1))^2+(g_v(2)*sigma(2))^2+(g_v(3)*sigma(3
))^2+(g_v(4)*sigma(4))^2+(g_v(5)*sigma(5))^2+(g_v(6)*sigma(6))^2+(g_v(
7)*sigma(7))^2);

        eta=abs(beta_p-beta)/abs(beta_p);

            end

         alfa=[-sigma(1)*g_v(1)/squaroot -sigma(2)*g_v(2)/squaroot -
sigma(3)*g_v(3)/squaroot -sigma(4)*g_v(4)/squaroot -
sigma(5)*g_v(5)/squaroot -sigma(6)*g_v(6)/squaroot -
sigma(7)*g_v(7)/squaroot];

         X=[mi(1)+beta*sigma(1)*alfa(1) mi(2)+beta*sigma(2)*alfa(2)
mi(3)+beta*sigma(3)*alfa(3) mi(4)+beta*sigma(4)*alfa(4)
mi(5)+beta*sigma(5)*alfa(5) mi(6)+beta*sigma(6)*alfa(6)
mi(7)+beta*sigma(7)*alfa(7)];

         u=[(X(1)-mi(1))/sigma(1) (X(2)-mi(2))/sigma(2) (X(3)-
mi(3))/sigma(3) (X(4)-mi(4))/sigma(4) (X(5)-mi(5))/sigma(5) (X(6)-
mi(6))/sigma(6) (X(7)-mi(7))/sigma(7)];

          u=[(X(1)-mi(1))/sigma(1) (X(2)-mi(2))/sigma(2) (X(3)-
mi(3))/sigma(3) (X(4)-mi(4))/sigma(4) (X(5)-mi(5))/sigma(5) (X(6)-
mi(6))/sigma(6) (X(7)-mi(7))/sigma(7)];

          iter=iter+1;

         end

Pf=1-normcdf(beta);


toc


end
%%  Daniele Casali_s162468_Cranfield University, MSc PSE, 2011
```

B.2.10 dynSRSM.m

```
%inputs
typelim=0;
lim=limit value;
mi=[mean values];
sigma=[std deviation values];
%sampling
clear all
```

```
clc


%inputs
k=input('number of design variables: ');
n=input('number of samples: ');
f=input('specify f, scaling factor: ');
tic
%samples creation, LHS
S=bestlh(n,k,100,40);

%samples scaling
Ss=zeros(size(S));
y=zeros(n,1);
% samples positions, cardinality for scaling
binsize=1/n;
pos=ones(size(S));
for i=1:k
    for j=1:n
        for l=1:n
    lowlim=binsize*(l-1);
    uplim=binsize*l;
        if lowlim<S(j,i)&& S(j,i)<=uplim
        pos(j,i)=l;
        end
        end
    end
end

% variables scaling
    % norminv on scaled probability matrix
    NrmInv_Pm=zeros(size(S));
for i=1:k
    for j=1:n
        NrmInv_Pm(j,i)=norminv(S(j,i)/n+(pos(j,i)-1)/n);
        if isinf(NrmInv_Pm(j,i))==1 && S(j,i)==0
            NrmInv_Pm(j,i)=-5;
        end
```

```
        if isinf(NrmInv_Pm(j,i))==1 && S(j,i)==1
            NrmInv_Pm(j,i)=5;
        end
    end
end
    % scaled samples matrix creation
for i=1:k
    for j=1:n
Ss(j,i)=mi(i)+sigma(i)*NrmInv_Pm(j,i);
    end
end
for i=1:n
    inputs=Ss(i,:);
    y(i,1)=scenario(inputs); %insert scenario number!!!
end



%Kriging correlation
global ModelInfo
ModelInfo.X=Ss;
ModelInfo.y=y;
% Start iterating infill points
for I=1:4
% Tune regressing Kriging model
UpLim=ones(1,k)*2;
LowLim=ones(1,k)*-25;
UpLim(end+1)=0;
LowLim(end+1)=-12;
[Params,MaxLikelihood]=ga(@reglikelihood,k+1,[],[],[],[],LowLim,UpLim)
;
% Extract model parameters
ModelInfo.Theta(1:k)=Params(1:k);
ModelInfo.Lambda=Params(end);
% Put Choleski factorization of Psi
% into ModelInfo
[NegLnLike,ModelInfo.Psi,ModelInfo.U,ModelInfo.mu]=reglikelihood([Mode
lInfo.Theta ModelInfo.Lambda]);
end
```

```
%FORM, first iteration
X=mi;
clc
eta=1;
iter=1;
beta=10;%whatever large value for beta is ok
%build design matrix, X
while eta>0.001
%simulation matrix
S=zeros(2*k+1,k);
for i=1:2*k+1
    S(i,:)=X;
end
for i=1:2*k+1
    for j=1:k
        if i==1
            S(i,j)=mi(j);
        elseif j+1==i && i<=n+1
            S(i,j)=mi(j)+f*sigma(j);
        elseif j+k+1==i && i>n+1
             S(i,j)= mi(j)-f*sigma(j);
         end
     end
end
%design matrix
D=ones(2*k+1,2*k+1);
i=1;
j=1;
for i=2:2*k+1
    if mod(i,2)==0
        D(:,i)=S(:,j);
    else
        D(:,i)=S(:,j).^2;
        j=j+1;
    end
end
```

```
)^2+(g_v(4)*sigma(4))^2+(g_v(5)*sigma(5))^2+(g_v(6)*sigma(6))^2+(g_v(7
)*sigma(7))^2);

        beta_p=beta;

        beta=(g_v(8)-
(g_v(1)*sigma(1)*u(1)+g_v(2)*sigma(2)*u(2)+g_v(3)*sigma(3)*u(3)+g_v(4)
*sigma(4)*u(4)+g_v(5)*sigma(5)*u(5)+g_v(6)*sigma(6)*u(6)+g_v(7)*sigma(
7)*u(7)))/sqrt((g_v(1)*sigma(1))^2+(g_v(2)*sigma(2))^2+(g_v(3)*sigma(3
))^2+(g_v(4)*sigma(4))^2+(g_v(5)*sigma(5))^2+(g_v(6)*sigma(6))^2+(g_v(
7)*sigma(7))^2);

        eta=abs(beta_p-beta)/abs(beta_p);

            end

         alfa=[-sigma(1)*g_v(1)/squaroot -sigma(2)*g_v(2)/squaroot -
sigma(3)*g_v(3)/squaroot -sigma(4)*g_v(4)/squaroot -
sigma(5)*g_v(5)/squaroot -sigma(6)*g_v(6)/squaroot -
sigma(7)*g_v(7)/squaroot];

         X=[mi(1)+beta*sigma(1)*alfa(1) mi(2)+beta*sigma(2)*alfa(2)
mi(3)+beta*sigma(3)*alfa(3) mi(4)+beta*sigma(4)*alfa(4)
mi(5)+beta*sigma(5)*alfa(5) mi(6)+beta*sigma(6)*alfa(6)
mi(7)+beta*sigma(7)*alfa(7)];

         u=[(X(1)-mi(1))/sigma(1) (X(2)-mi(2))/sigma(2) (X(3)-
mi(3))/sigma(3) (X(4)-mi(4))/sigma(4) (X(5)-mi(5))/sigma(5) (X(6)-
mi(6))/sigma(6) (X(7)-mi(7))/sigma(7)];

         iter=iter+1;

        end

Pf=1-normcdf(beta)

toc

%%  Daniele Casali_s162468_Cranfield University, MSc PSE, 2011
```

B.2.11 dynSRSM_2ph.m

```
% before running check: stoch variables#, mi and sigma arrays,
'data.xlsx' where scaled sample points 'Ss' and outputs 'out' are
reported from the simulations run (procedure completed using the
simulation tool and completely external to Matlab)



%sampling
clear all
clc


%inputs
k=7;
f=input('specify f, horizon factor: ');
tic
MoS=MoS();
MoS=MoS';
mi=[mean values];
sigma=[std deviations values];
[n,n_el]=size(MoS);
for counter=49
y=MoS(:,counter);
        Ss=xlsread('data.xlsx','Ss','A1:G15');
        %Ss=xlsread('data20.xlsx','Ss','A1:G20');
        %Ss=xlsread('data40.xlsx','Ss','A1:G40');
        %Ss=xlsread('data60.xlsx','Ss','A1:G60');
%Kriging correlation
global ModelInfo
ModelInfo.X=Ss;
ModelInfo.y=y;
% Start iterating infill points
for I=1:4
% Tune regressing Kriging model
UpLim=ones(1,k)*2;
LowLim=ones(1,k)*-25;
UpLim(end+1)=0;
LowLim(end+1)=-12;
```

```
[Params,MaxLikelihood]=ga(@reglikelihood,k+1,[],[],[],[],LowLim,UpLim)
;
% Extract model parameters
ModelInfo.Theta(1:k)=Params(1:k);
ModelInfo.Lambda=Params(end);
% Put Choleski factorization of Psi
% into ModelInfo
[NegLnLike,ModelInfo.Psi,ModelInfo.U,ModelInfo.mu]=reglikelihood([Mode
lInfo.Theta ModelInfo.Lambda]);
%[NegLnLike,ModelInfo.Psi,ModelInfo.mu]=reglikelihood([ModelInfo.Theta
ModelInfo.Lambda]);
        end



%FORM, first iteration
X=mi;
clc
eta=1;
```

```
iter=1;
beta=10;%whatever large value for beta is ok
%build design matrix, X
while eta>0.001
%simulation matrix
S=zeros(2*k+1,k);
for i=1:2*k+1
    S(i,:)=X;
end
for i=1:2*k+1
    for j=1:k
        if i==1
            S(i,j)=mi(j);
        elseif j+1==i && i<=n+1
            S(i,j)=mi(j)+f*sigma(j);
        elseif j+k+1==i && i>n+1
            S(i,j)= mi(j)-f*sigma(j);
        end
    end
end
%design matrix
D=ones(2*k+1,2*k+1);
i=1;
j=1;
for i=2:2*k+1
    if mod(i,2)==0
        D(:,i)=S(:,j);
    else
        D(:,i)=S(:,j).^2;
        j=j+1;
    end
end

% for each matrix row-->a prediction value using kriging, Y
n_max=2*k+1;
Y=ones(n_max,1);
ModelInfo.Option='Pred';
for i=1:n_max
```

```
    in=S(i,:);
    Y(i,1)=regpredictor(in);
end
%find quad poly coefficients, a=Y/X;
%a=((D'*D))\(D'*Y);
a=pinv(D)*Y;
Yappr=D*a;
max_err=max(abs(Y-Yappr))
%LSF definition
syms x1 x2 x3 x4 x5 x6 x7;
L=[x1 x2 x3 x4 x5 x6 x7];


% 1 - definition of the limit state function


g=a(1)+a(2)*x1+a(3)*x1^2+a(4)*x2+a(5)*x2^2+a(6)*x3+a(7)*x3^2+a(8)*x4+a
(9)*x4^2+a(10)*x5+a(11)*x5^2+a(12)*x6+a(13)*x6^2+a(14)*x7+a(15)*x7^2;
%FORM iterations
 if iter==1
g_v=[double(subs(diff(g,L(1)),L(1),mi(1)))
double(subs(diff(g,L(2)),L(2),mi(2)))
double(subs(diff(g,L(3)),L(3),mi(3)))
double(subs(diff(g,L(4)),L(4),mi(4)))
double(subs(diff(g,L(5)),L(5),mi(5)))
double(subs(diff(g,L(6)),L(6),mi(6)))
double(subs(diff(g,L(7)),L(7),mi(7))) double(subs(g, L, mi))];
squaroot=sqrt((g_v(1)*sigma(1))^2+(g_v(2)*sigma(2))^2+(g_v(3)*sigma(3)
)^2+(g_v(4)*sigma(4))^2+(g_v(5)*sigma(5))^2+(g_v(6)*sigma(6))^2+(g_v(7
)*sigma(7))^2);
beta=g_v(8)/squaroot
eta=1;
    else
g_v=[double(subs(diff(g,L(1)),L(1),X(1)))
double(subs(diff(g,L(2)),L(2),X(2)))
double(subs(diff(g,L(3)),L(3),X(3)))
double(subs(diff(g,L(4)),L(4),X(4)))
double(subs(diff(g,L(5)),L(5),X(5)))
double(subs(diff(g,L(6)),L(6),X(6)))
double(subs(diff(g,L(7)),L(7),X(7))) double(subs(g, L, X))];
beta_p=beta;
squaroot=sqrt((g_v(1)*sigma(1))^2+(g_v(2)*sigma(2))^2+(g_v(3)*sigma(3)
)^2+(g_v(4)*sigma(4))^2+(g_v(5)*sigma(5))^2+(g_v(6)*sigma(6))^2+(g_v(7
)*sigma(7))^2);
beta=(g_v(8)-
(g_v(1)*sigma(1)*u(1)+g_v(2)*sigma(2)*u(2)+g_v(3)*sigma(3)*u(3)+g_v(4)
*sigma(4)*u(4)+g_v(5)*sigma(5)*u(5)+g_v(6)*sigma(6)*u(6)+g_v(7)*sigma(
7)*u(7)))/squaroot
```

```
eta=abs(beta_p-beta)/abs(beta_p);
     end
 alfa=[-sigma(1)*g_v(1)/squaroot -sigma(2)*g_v(2)/squaroot -
sigma(3)*g_v(3)/squaroot -sigma(4)*g_v(4)/squaroot -
sigma(5)*g_v(5)/squaroot -sigma(6)*g_v(6)/squaroot -
sigma(7)*g_v(7)/squaroot];
 X=[mi(1)+beta*sigma(1)*alfa(1) mi(2)+beta*sigma(2)*alfa(2)
mi(3)+beta*sigma(3)*alfa(3) mi(4)+beta*sigma(4)*alfa(4)
mi(5)+beta*sigma(5)*alfa(5) mi(6)+beta*sigma(6)*alfa(6)
mi(7)+beta*sigma(7)*alfa(7)];
 u=[(X(1)-mi(1))/sigma(1) (X(2)-mi(2))/sigma(2) (X(3)-mi(3))/sigma(3)
(X(4)-mi(4))/sigma(4) (X(5)-mi(5))/sigma(5) (X(6)-mi(6))/sigma(6)
(X(7)-mi(7))/sigma(7)];
 iter=iter+1;
end
Beta(counter,1)=beta;
Pf(counter,1)=1-normcdf(beta);


toc
end
```

B.3 Surrogate modelling

B.3.1 krichk.m

```
% giving a number of samples checks the error done by kriging using
this number of samples
clear all
clc
close all
k=7; %# of variables
mi=[mean values];
sigma=[std deviation values];
i=input('# intervals: '); %measure of how much refined is the k-
dimensional function we are building
ns=input('# of sample points: '); %measure of how much we sample the
actual function, affects accuracy in the prediction.
max=zeros(1,k);
min=zeros(1,k);
incr=zeros(1,k);
tic
for j=1:k
    max(j)=mi(j)+3*sigma(j);
```

```
    min(j)=mi(j)-3*sigma(j);
    incr(j)=(max(j)-min(j))/i;
end
n=(i+1)^k;
M=zeros(i+1,k);
M(1,:)=min;
for j=2:(i+1)
    M(j,:)=M(j-1,:)+incr;
end
X=zeros(n,k);
% upper design matrix
for col=1:k
    rlast=0;
    for j=1:(i+1)
        val=M(j,col);
        for r=(rlast+1):((i+1)^(k-col)+rlast)
            X(r,col)=val;
        end
        rlast=r;
    end
end
```

```
%complete design matrix
for col=2:k
    v=[];
    for j=1:(i+1)^(k-col+1)
        v(j,1)=X(j,col);
    end
    [rx,cx]=size(X);
    [rv,cv]=size(v);
    m=rx/rv;
    v=repmat(v,m,1);
    X(:,col)=v;
end
% real responses generation
for j=1:n
    in=X(j,:);
    Y(j,1)=scenario(in);
end
%sampling
%samples creation, LHS
S=bestlh(ns,k,100,40);
%samples scaling
Ss=zeros(size(S));
y=zeros(ns,1);
% samples positions, cardinality for scaling
binsize=1/ns;
pos=ones(size(S));
for i=1:k
    for j=1:ns
        for l=1:ns
    lowlim=binsize*(l-1);
    uplim=binsize*l;
        if lowlim<S(j,i)&& S(j,i)<=uplim
        pos(j,i)=l;
        end
        end
    end
end
```

```
% variables scaling
    % norminv on scaled probability matrix
    NrmInv_Pm=zeros(size(S));
for i=1:k
    for j=1:ns
        NrmInv_Pm(j,i)=norminv(S(j,i)/ns+(pos(j,i)-1)/ns);
        if isinf(NrmInv_Pm(j,i))==1 && S(j,i)==0
            NrmInv_Pm(j,i)=-5;
        end
        if isinf(NrmInv_Pm(j,i))==1 && S(j,i)==1
            NrmInv_Pm(j,i)=5;
        end
    end
end
    % scaled samples matrix creation
for i=1:k
    for j=1:ns
Ss(j,i)=mi(i)+sigma(i)*NrmInv_Pm(j,i);
```

```
    end
end
Ysamp=zeros(ns,1);
for i=1:ns
    inputs=Ss(i,:);
    Ysamp(i,1)=scenario(inputs);
end



%kriging correlation
global ModelInfo
ModelInfo.X=Ss;
ModelInfo.y=Ysamp;
% Start iterating infill points
for I=1:4
% Tune regressing Kriging model
UpLim=ones(1,k)*2;
LowLim=ones(1,k)*-25;
UpLim(end+1)=0;
LowLim(end+1)=-12;
[Params,MaxLikelihood]=ga(@reglikelihood,k+1,[],[],[],[],LowLim,UpLim)
;
% Extract model parameters
ModelInfo.Theta(1:k)=Params(1:k);
ModelInfo.Lambda=Params(end);
% Put Choleski factorization of Psi
% into ModelInfo
[NegLnLike,ModelInfo.Psi,ModelInfo.U,ModelInfo.mu]=reglikelihood([Mode
lInfo.Theta ModelInfo.Lambda]);
end


%kriging prediction at points of X, n predictions
Ypred=zeros(n,1);
ModelInfo.Option='Pred';
for i=1:n
    in=X(i,:);
    Ypred(i,1)=regpredictor(in);
end
```

```
%error calculation, check kriging accuracy over the design space
e=abs(Y-Ypred)./Y;
avgerr=mean(e)
maxerr=max(e)
toc
```

Appendix C Matrix algebra

C.1 Partitioned inverse derivation

$$\tilde{\boldsymbol{\Psi}}^{-1}\tilde{\boldsymbol{\Psi}} = \tilde{\boldsymbol{\Psi}}\tilde{\boldsymbol{\Psi}}^{-1} = \begin{pmatrix} A & B \\ C & D \end{pmatrix}\begin{pmatrix} \boldsymbol{\Psi} & \boldsymbol{\psi} \\ \boldsymbol{\psi}^{T} & 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{I} & 0 \\ 0 & \boldsymbol{I} \end{pmatrix}$$

$$\begin{cases} A\boldsymbol{\Psi} + B\boldsymbol{\psi}^{T} = \boldsymbol{I} \\ A\boldsymbol{\psi} + B = 0 \\ C\boldsymbol{\Psi} + D\boldsymbol{\psi}^{T} = 0 \\ C\boldsymbol{\psi} + D = \boldsymbol{I} \end{cases}$$

$$\begin{cases} A = \boldsymbol{\Psi}^{-1} + \boldsymbol{\Psi}^{-1}\boldsymbol{\psi}(1 - \boldsymbol{\psi}^{T}\boldsymbol{\Psi}^{-1}\boldsymbol{\psi})^{-1}\boldsymbol{\psi}^{T}\boldsymbol{\Psi}^{-1} \\ B = -\boldsymbol{\Psi}^{-1}\boldsymbol{\psi}(1 - \boldsymbol{\psi}^{T}\boldsymbol{\Psi}^{-1}\boldsymbol{\psi})^{-1} \\ C = B^{T} = -(1 - \boldsymbol{\psi}^{T}\boldsymbol{\Psi}^{-1}\boldsymbol{\psi})^{-1}\boldsymbol{\psi}^{T}\boldsymbol{\Psi}^{-1} \\ D = (1 - \boldsymbol{\psi}^{T}\boldsymbol{\Psi}^{-1}\boldsymbol{\psi})^{-1} \end{cases}$$

C.2 Cholesky factorization

$$\tilde{\boldsymbol{\Psi}} = LL^{\dagger}$$

where $L^{\dagger}$ is the transpose complex conjugate. Since in our particular application there is no complex domain analysis it coincides with the transpose.

Appendix D Results

Colours are mapped according to the legend:

year

el#	1	2	2.1	2.2	2.3	2.4	2.5	2.6	2.7
1	-	-	-	-	-	-	-	-	-
2	-	-	-	-	-	-	-	-	-
3	-	-	-	-	-	-	-	-	-
4	-	-	-	-	-	-	-	-	-
5	-	-	-	-	-	-	-	-	-
6	-	-	-	-	-	-	-	-	-
7	-	-	-	-	-	-	-	-	-
8	-	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-	-
10	-	-	-	-	-	-	-	-	-
11	-	-	-	-	-	-	-	-	-
12	-	-	-	-	-	-	-	-	-
13	-	-	-	-	-	-	-	-	-
14	-	-	-	-	-	-	-	-	-
15	-	-	-	-	-	-	-	-	-
16	-	-	-	-	-	-	-	-	-
17	-	-	-	-	-	-	-	-	-
18	-	-	-	-	-	-	-	-	-
19	-	-	-	-	-	-	-	-	-
20	-	-	-	-	-	-	-	-	-
21	-	-	-	-	-	-	-	-	-
22	-	-	-	-	-	-	-	-	-
23	-	-	-	-	-	-	-	-	-
24	-	-	-	-	-	-	-	-	-
25	-	-	-	-	-	-	-	-	-

26	-	-	-	-	-	-	-	-	-
27	-	-	-	-	-	-	8.076571	-	-
28	-	-	-	-	-	-	-	-	8.209536
29	-	-	-	-	-	-	-	-	7.757533
30	-	-	-	-	-	-	7.118466	7.869778	7.23673
31	-	-	-	-	-	-	7.972552	7.360526	6.674624
32	-	-	-	-	-	8.209536	7.641094	6.992328	6.261359
33	-	-	-	-	-	8.0414	7.452972	6.782261	6.022901
34	-	-	-	-	-	7.877848	7.259404	6.56486	5.775273
35	-	-	-	8.209536	8.209536	7.698505	7.060531	6.341147	5.518202
36	-	-	-	-	8.125891	7.516181	6.856092	6.109375	5.250203
37	-	-	-	-	7.941444	7.32814	6.644955	5.869223	4.970104
38	-	-	-	-	7.764209	7.135345	6.428668	5.620703	4.676606
39	-	-	-	8.209536	7.586514	6.937758	6.204799	5.362172	4.368176
40	-	-	-	8.014016	7.404921	6.734205	5.97336	5.092457	4.041666
41	-	-	-	7.835685	7.219034	6.525465	5.733963	4.81058	3.695535
42	-	-	-	7.748126	7.118887	6.41115	5.601373	4.652021	3.496435
43	-	-	-	7.728523	7.097454	6.386129	5.57082	4.613463	3.444849
44	-	-	8.209536	7.711503	7.075611	6.360131	5.539198	4.573659	3.391818
45	-	-	8.209536	7.692463	7.054077	6.333823	5.507157	4.533236	3.337915
46	-	-	8.209536	7.672441	7.031454	6.306906	5.47434	4.491773	3.282091
47	-	-	8.209536	7.655089	7.00918	6.280062	5.441272	4.449585	3.224929
48	-	-	8.209536	7.633364	6.985381	6.251703	5.406635	4.40566	3.165556
49	-	7.232089	6.025717	5.355374	4.104562	2.448713	-0.04884	-2.07642	-2.98489
50	-	-	-	7.580521	6.75608	5.789865	4.621976	3.129823	0.989601
51	-	-	-	-	-	7.972552	7.190842	6.280169	5.199463
52	-	-	-	-	-	-	-	8.209536	7.606326
53	-	-	-	-	-	-	-	-	-
54	-	-	-	-	-	-	-	-	-

NOMENCLATURE

Symbols

a	accuracy
$\boldsymbol{c}$	centre of radial basis function
cov	covariance
CoV	Coefficient of Variation
D	diameter
$\boldsymbol{e}$	approximation error vector
f	friction factor
f	‘horizon’ factor
F	design factor
g	gravitational acceleration
g	LSF
$\boldsymbol{\mathcal{G}}$	kriging predictor term independent from prediction point
h	height
H	head
k	component pressure loss coefficient
k	number of variables
L	length
L	limit value
$\dot{M}$	mass flow rate
MoS	Margin of safety
n	number of samples
n_{CO_2}	carbon dioxide mol%
N_f	number of failures
N	number of runs

p_j	'smoothness' coefficient
p	pressure
P	probability
P_f	probability of failure
Re	Reynolds number
s	trajectory
$SYMS$	Yield strength
t	thickness
T	temperature
v	velocity
v_c	corrosion velocity
$\dot{V}$	volumetric flow rate
V	actual value
x	inputs
$\boldsymbol{X}$	design matrix, samples matrix
y	outputs
$\hat{y}$	approximated output
y^*	predicted output, predicted response
$\hat{\boldsymbol{Y}}$	approximated responses vector
$\boldsymbol{Y}$	responses vector
$Y(x)$	variables realisation
z	elevation
$\boldsymbol{Z}$	HL-transformed design variable
α	HL director cosines, sensitivity factors
$\boldsymbol{\alpha}$	SRSM vector coefficient
β	reliability index

γ	density
ε	roughness
ε	approximation error
ϑ_j	'activity' parameter
λ	noise-removal constant
μ	mean
π	pi constant
σ^2	variance
σ	standard deviation
φ	correlation function
Φ, F	CDF
φ, f	PDF
$\boldsymbol{\Psi}$	correlation matrix
$\boldsymbol{\varphi}$	correlation vector

Abbreviations

CU	Cranfield University
CV, CoV	Coefficient of Variation
FMEA	Failure Modes and Effects Analysis
FMECA	Failure Modes, Effects and Criticalities Analysis
FORM	First Order Reliability Method
FORM-HL	First Order Reliability Method – Hasofer and Lind formulation
FOSM	First Order - Second Moment
FP	Failure Point
FTA	Fault Tree Analysis
HL	Hasofer and Lind
ISO	International Standards Organisation
LHS	Latin Hypercube Sampling
LSM	Least Square Method
MCS	Monte Carlo Simulation

MLE	Maximum Likelihood Estimate
MoS	Margin of Safety
MPP	Most Probable failure Point
MVFOSM	Mean Value First Order Second Moment
NPSHa	Net Pressure Suction Head available
NPSHr	Net Pressure Suction Head required
Oreda	Offshore Reliability Data
PDF	Probability Distribution Function
PFE2010	Pipe Flow Expert 2010
RA	Reliability Analysis
RCM	Reliability-Centered Maintenance
SOSM	Second Order – Second Moment
SRSM	Stochastic Response Surface Method
WOAD	Worldwide Offshore Accidents Databank

REFERENCES

[1] Pipe characteristics handbook / Williams National Gas Company Engineering Group, (1996), Tulsa, OK : Pennwell, 1996.

[2] Alfon, P., Soedarsono, J.W., Priadi, D. and Sulistijono, (2012), Pipeline material reliability analysis regarding to probability of failure using corrosion degradation model.

[3] Alfon, P., Soedarsono, J.W., Priadi, D. and Sulistijono, (2012), Pipeline material reliability analysis regarding to probability of failure using corrosion degradation model.

[4] Bai, Y., Moe, E. T. and Mork, K. (1994), "Probabilistic assessment of dented and corroded pipeline", Proceedings of the International Offshore and Polar Engineering Conference, Vol. 2, pp. 93.

[5] Bai, Y. and Referex (2001), Pipelines and risers, Elsevier, Amsterdam.

[6] Belov, D.A., Coupe, P., Fukuhara, M. and Myalo, E.V., (2011), Material-loss estimation for pipes.

[7] Bhattacharya, B., Basu, R. and Kai-tung Ma "Developing target reliability for novel structures: the case of the Mobile Offshore Base", Marine Structures, vol. 14, pp. 37-58.

[8] Carpenter, P., Nicholas, E. and Henrie, M. (2006), "Bayesian Belief Networks For Pipeline Leak Detection", 11-13 October 2006, Pipeline Simulation Interest Group, .

[9] Choi, S., Grandhi, R.V., Canfield, R.A. and SpringerLink, (2007), Reliability-based structural design, Springer, London.

[10] Cimbala, J. M. and Cengel, Y. A. (2008), Essentials of fluid mechanics: fundamentals and applications, McGraw-Hill, Boston, MA.

[11] Coulson, J.M., Richardson, J.F., Coulson, J.M. and Knovel, (2009; 1999), Coulson & Richardson's chemical engineering, Butterworth-Heinemann, Amsterdam ; Boston.

[12] Davis, C. V. and Sorensen, K. E. (1969), Handbook of applied hydraulics, 3d ed, McGraw-Hill, New York.

[13] Edwards, , John, Mork, K., Norske, D. and Sydberger, T. (1996), "Reliability Based Design of CO2 Corrosion Control", 24-29 March 1996, NACE International, .

[14] Forrester, A. I. J., Sóbester, A. and Keane, A. J. (2006), "Optimization with missing data", Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 462, no. 2067, pp. 935-945.

[15] Forrester, A.I.J., Sóbester, A. and Keane, A.J., (2008), Engineering design via surrogate modelling, Wiley, Chichester, UK.

[16] Fulford, J. M., et al. (2006), Accuracy of radar water level measurements .

[17] Gavin, H. P. and Yau, S. C. (2008), "High-order limit state functions in the response surface method for structural reliability analysis", Structural Safety, vol. 30, no. 2, pp. 162-179.

[18] Hasofer, A. M. and Lind, N. C. (1973), An exact and invariant First-Order Reliability format, Solid Mechanic Division, University of Waterloo, Waterloo, Ontario, Canada.

[19] Hasofer, A. M. and Lind, N. C. (1974), "EXACT AND INVARIANT SECOND-MOMENT CODE FORMAT.", ASCE J Eng Mech Div, vol. 100, no. EM1, pp. 111-121.

[20] Isukapalli, S. (1999), Uncertainty analysis of transport-transformation models (PhD thesis), The State University of New Jersey, New Brunswick, New Jersey.

[21] Jeong, S., Yamamoto, K. and Obayashi, S. (2004), "Kriged-Based Probabilistic Method for Constrained Multi-Objective Optimization Problem", AIAA Journal, , no. 6437.

[22] Jones, D. R. (2001), "A Taxonomy of Global Optimization Methods Based on Response Surfaces", Journal of Global Optimization, vol. 21, no. 4, pp. 345-383.

[23] Kolios, A. I. and Brennan, F. (2010), A multi-configuration approach to reliability based structural integrity assessment for ultimate strength. / Athanasios Ioannis Kolios, .

[24] Kolios, A. and Brennan, F. "Reliability deterioration of offshore/marine steel structures due to effect of corrosion", .

[25] Lansey, K. E., Mays, L. W., Tung, Y. K. and Duan, N. (1989), "Water distribution system design under uncertainties.", J.WATER RESOUR.PLANN.& MANAGE.(ASCE), vol. 115, no. 5 , Sep. 1989, p.630-645.

[26] Ma, B., Shuai, J., Wang, J. and Han, K. (2011), "Analysis on the Latest Assessment Criteria of ASME B31G-2009 for the Remaining Strength of Corroded Pipelines", Journal of Failure Analysis & Prevention, vol. 11, no. 6, pp. 666.

[27] Mahajan, U., Asprolis, N., Kolios, A. I. and Cranfield University. School of Engineering (2011), Sensitivity analysis in jet impingment heat transfer using stochastic surface response method. .

[28] Melchers, R. E. (1999), Structural reliability analysis and prediction, 2nd ed, John Wiley, Chichester.

[29] Melchers, R. E. "Corrosion uncertainty modelling for steel structures", Journal of Constructional Steel Research, vol. 52, pp. 3-19.

[30] Nyborg, R. (2002), "Overview of CO2 corrosion models for wells and pipelines", NACE - International Corrosion Conference Series, , no. 02233.

[31] Peet, S., Race, J., Dawson, J. and Krishnamurthy, R. M. (2001), "Pipeline Corrosion Management", 11-16 March 2001, NACE International, .

[32] Proia, S. and Kolios, A. (2011), Reliability assessment of complex structures in composite materials under stochastic loading / Stefano Proia, .

[33] Quinio, A. and Kolios, A. (2010), A stochastic finite element based approach to the design optimization of a truss structure. / Adrien Quinio, .

[34] Sandalci, M., Mançuhan, E., Alpman, E. and Küçükada, K. (2010), "Effect of the flow conditions and valve size on butterfly valve performance", Isi Bilimi Ve Teknigi Dergisi/ Journal of Thermal Science and Technology, vol. 30, no. 2, pp. 103-112.

[35] Shibu, A. and Janga Reddy, M. (2011), "Uncertainty analysis in water distribution networks by Fuzzy-Cross entropy approach", World Academy of Science, Engineering and Technology, vol. 59.

[36] Simpson, T. W., Mauery, T. M., Korte, J. J. and Mistree, F. (1998), "Comparison of response surface and kriging models for multidisciplinary design optimisation", AIAA Journal, , no. 4755.

[37] Sobey, A., Underwood, J., Blake, J. and Shenoi, A. "Reliability analysis of damaged steel structures using finite element analysis".

[38] Srdjan Nešić, Jiyong Cai and Kun-Lin John Lee (2005), "A multi-phase flow and internal corrosion prediction model for mild steel pipelines", NACE - International Corrosion Conference Series, , no. 05556.

[39] Swamee, P. K. and Sharma, A. K. (2008), Desing of water supply pipe networks, John Wiley and Sons, Hoboken, New Jersey.

[40] Toal, D. J. J., Bressloff, N. W. and Keane, A. J. (2008), "Kriging hyperparameter tuning strategies", AIAA Journal, vol. 46, no. 5, pp. 1240-1252.

[41] Twort, A. C., Ratnayaka, D. D. and Brandt, M. J. (2000), Water supply / Alan C. Twort, Don D. Ratnayaka, Malcolm J. Brandt, London : Arnold, 2000; 5th ed.

[42] U.S. Army Corps of Engineers (1999), EM 1110-1-4008 Liquid Process Piping, .

[43] Wyant, J. C. and Creath, K. (1990), "Absolute measurement of surface roughness", Proceedings of SPIE - The International Society for Optical Engineering, Vol. 1319, pp. 568.

[44] DeFlow ltd. , available at: http://www.deflow.com/.

[45] Elster Metering , available at: http://www.elster.com/en/index.

[46] Engineering Toolbox , available at: http://www.engineeringtoolbox.com/.

[47] ESA AATSR EnviSat , available at: https://earth.esa.int/web/guest/missions/esa-operational-eo-missions/envisat/instruments/aatsr/.

[48] Kolios, A. (2011), Risk and Reliability module lecture notes (unpublished Process Systems MSc thesis), Cranfield Uni., Cranfield, UK.

[49] Norwegian Society for Oil and Gas Measurement and Tekna (2005), Handbook of Multiphase Flow Metering, .

[50] Oxford Dictionaries , available at: http://oxforddictionaries.com/.

[51] SPT Group, (2010), OLGA Manual.

[52] Fleming, K. N. (2004), "Markov models for evaluating risk-informed in-service inspection strategies for nuclear power plant piping systems", Reliability Engineering & System Safety, vol. 83, no. 1, pp. 27-45.

[53] Hohenbichler, M., and Rackwitz, R., "Non-normal Dependent Vectors in Structural Safety," Journal of the Engineering Mechanics Division, ASCE, Vol. 107, No. EM6, Dec. 1981, pp.1227-1238.

[54] Sacks, J., Welch, W. J., Mitchell, T. J. and Wynn, H. (1989) Design and analysis of computer experiments. Statistical Science, 4(4), 409–423.

[55] Theil, H. (1971) Principles of Econometrics, John Wiley & Sons, Inc., New York.

[56] Press, W. H., Flannery, B. P., Teukolsky, S. A. and Vetterling, W. T. (1986) Numerical Recipes – The Art of Scientific Computing, Cambridge University Press.

"POST NUBES LUX"