NTNU
Det skapende universitet

TKP4580 1 Kjemisk prosessteknologi, fordypningsprosjekt

Kandidat 10023

Oppgaver	Oppgavetype	Vurdering	Status
1 Handing in the specialization report.	Filopplasting	Manuell poengsum	Levert

TKP4580 1 Kjemisk prosessteknologi, fordypningsprosjekt

Emnekode	TKP4580	PDF opprettet	19.12.2016 10:33
Vurderingsform	TKP4580	Opprettet av	Hege Johannessen
Starttidspunkt:	30.11.2016 10:00	Antall sider	46
Sluttidspunkt:	16.12.2016 15:45	Oppgaver inkludert	Ja
Sensurfrist	Ikke satt	Skriv ut automatisk rettede	Ja

Seksjon 1

1 OPPGAVE

Handing in the specialization report.

Here you can upload your project report.
Only .pdf files are accepted.

Mark the file name with name_research group_supervisor!

Click here to upload your file

BESVARELSE

Filopplasting

Filnavn	10163874_cand-12117652_9125253
Filtype	pdf
Filstørrelse	434.131 KB
Opplastingstid	16.12.2016 14:26:27

Neste side
Besvarelse vedlagt

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for naturvitenskap og teknologi
Institutt for kjemisk prosessteknologi

SPECIALIZATION PROJECT 2016

TKP4580/TKP4581

PROJECT TITLE:

Investigation of the influence of the sampling domain on the performance of surrogate models.

By

Petter Bjørnstad Markussen

Supervisor for the project: Sigurd Skogestad
Co-supervisor: Julian Straus

Date:16.12.2016

Summary

A surrogate model was fitted to a simple ammonia synthesis model. A PLS regression was first conducted to reduce the number of variables before it was fitted to a neural network. A study of the parameters in PLS indicated that the first component is the most significant. From the analysis it did not make any difference on the performance which combination of the independent variables that was used, as long as the first component was included. From the performance of the surrogate model it was concluded that a orthogonal method with two subspaces should be used for sampling the design space. A random sampling method for designing the surrogate model had the worst performance of the three sampling methods. This argument was based on the maximum absolute value, standard deviation and the confidence interval for the relative average error when random sampling was used.

Table of Contents

List of Tables

List of Figures

Nomenclature

Variable name		Explanation of variable
X	=	Independent variables
X'	=	Reduced number of independent variables
p_{out}	=	Outlet pressure
Q	=	Scores of Y
T	=	Scores of X
U	=	Loadings of Y
P	=	Loadings of X
F	=	Error in Y
E	=	Error in X
Π	=	The product of sequence of terms
M!	=	Factorial of M
μ_U=	=	Average value of U
σ_U	=	Standard deviation of U
E(err)	=	Average error of outlet pressure
a_i	=	Confidence interval for sampling method i
ξ	=	Exten of reaction
μ_i	=	Reaction coefficient for component i
$\frac{a_i}{2}$	=	Upper bound in a 95% confidence interval for the error with sampling size i.
$1 - \frac{a_i}{2}$	=	Lower bound in a 95% confidence interval for the error with sampling size i.

Chapter 1

Introduction

This specialization project presents a Master of Science degree project that is conducted in Chemical engineering, at NTNU.

There consists a problem in optimizing a ammonia plant. The reason for this is that the process is highly integrated (Straus and Skogestad (2016)). Its therefore an idea to split the process into several sub models and optimize each sub models with its own surrogate models. One of the sub models is the reaction section in the ammonia synthesis loop. The surrogate models for this sub model will be studied in detail in this project. Surrogate models is introduced to make a simplified model of the original model. If the surrogate model is a good representation for the behavior of the flowsheet it could be of interest to use it for optimization.

As Forrester and Keane (2009) mentions, all the different optimization schemes depend on the underlying model that is used. Many methods need to use the Hessian matrix from the current sampling point over the design space that is given. This takes a lot of computational power, and even more so if one must use finite difference approximation for the derivatives in the Hessian. For this reason, it is of interest to reduce the time it takes to do the computation, so that the optimal operation point for the Ammonia loop can be found. A surrogate model will make this computational time negligible compared to the original model.

Surrogate models are still very limited to the number of independent variables. As is mentioned in (Straus and Skogestad (2017)), the number of validations of the flowsheet increases exponentially with the number of independent variables. It is therefore important to use a minimum number of independent variables that can describe the flowsheet with good accuracy. The surrogate model must also have enough of sampling points for each variable, such that the whole design space is covered. The surrogate model must remain simple and also be a good representation of the original model. It is for this reason that this project will investigate how the sampling space influences the performance of the surrogate model on the reaction section. An investigation on how to sample the design

space for a surrogate model fitting is done in this project. The surrogate model will only be fitted to one output; the outlet pressure

This thesis uses a surrogate model that already existed in MATLAB. It can be used for optimization of the reactor section in the simple ammonia loop. It uses a partial least square regression (Wold et al. (1983)) and cascade-forward neural network fitting (Mathworks (2010)). The flowsheet for the original model that the surrogate model is fitted to is given by the figure 1.1.

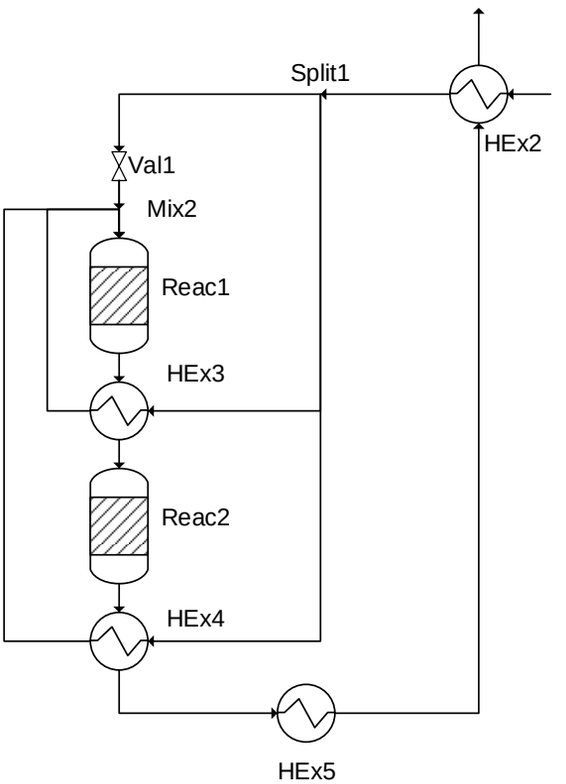

Figure 1.1: Flowsheet for the reaction section that synthesis ammonia

The flowsheet (figure1.1) consist of two reactor beds. To exploit the heat generated in the reactors, a heat exchange network is used. HEx3 removes the heat generated in reactor 1 and preheats the inlet feed before the mixing accurse in Mx2. The reaction is exothermic and removing the heat between the reactors will increase the yield. (Straus and Skogestad (2017)).

Chapter 2

Theory

This chapter will cover the theoretical background of this project. It starts with the explanation of the model structure. This will be followed by an explanation of a partial least square method for regression and the different sampling methods. It will conclude with an introduction of the surrogate models that is used in this project; neural networks.

2.1 Topology of the system

Surrogate modelling is introduced to obtain a simplified model of any given model. The surrogate model that is used in this paper contains two parts; a partial least square regression (PLS) and a neural network fitting (ANN).

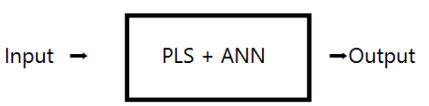

Figure 2.1: Topology for the surrogate model used for the simple ammonia loop

The original model contains 10 independent variables: $n_u = 10$. Seven of these come from the inlet feed (N_i, P_{in}, T_{in}), two split ratios (one variables for the split in the valve (Val1) and one for the split in the heat exchanger 4 (HEx4). The last variable is the outlet temperature of heat exchanger 4 ($T_{Ref,4}$).

Reduction of variables must be done because ten independent variables are too many for fitting to a surrogate model. PLS regression is therefore used to reduce the number of independent variables before it is fitted to the neural network.

It can be possible to get mass build-up in the reaction section in this surrogate model. The reason is that mass creation or destruction can occur when the surrogate model is fitted for each of the outlet molar flows. To circumvent this problem, linear mass balances is introduced by using the extent of reaction. This ensures that the total mass in the system is consistent. The linear mass relationship is given by:

$$F_{i,out}^{:} = F_{i,in}^{:} + \nu_i \xi \tag{2.1}$$

2.2 Partial least square regression

For a number of observations X and a number of predictor variables Y, a PLS regression gives a linear relationship between these data. The output Y can then be predicted by only knowing X. The procedure for making PLS regression is as follows: A decomposition of the matrices X and Y is made such that the scalar product of the two vectors: scores (Q an T) and loadings (P and U) represent X and Y.

$$Y = QU + F \tag{2.2}$$

$$X = TP + E \tag{2.3}$$

T and P are the corresponding scores for the respective observations X (2.3) and Y (2.2). The scores and loadings of both X and Y are generated so that the i-th Yscore (u_i) has *maximum covariance* with the corresponding Xscore, t_i.

$$u_i = r_i T_i \tag{2.4}$$

$$U = [u_1 u_2, ...u_i, ...u_n] \tag{2.5}$$

Here, n is the number of components in the PLS regression and r_i (2.4) is the correlation factor between the scores. The prediction of the outputs Y can be calculated from the scores of Y (2.2). Since PLS finds a maximum correlation between the scores of X and Y (2.4), the scores of Y can be represented with only knowing the scores of X. Using a PLS regression, the prediction of new output(s) Y can be found from only knowing the scores and the respective loadings of X. The PLS will also reduce the dimensionality of the problem, so that the outputs can be described by less components than the original number of independent variables from the reactor section.

2.3 Sampling schemes

Four different sampling techniques are introduced : Latin hyper cube, random, Orthogonal and regular grid sampling.

2.3.1 Random sampling

The random sampling (RND) method samples randomly over the whole space. It does not take into account the previous sampling points when it generates new points. Therefore, one can run into the risk of only sample in one region. Homogeneous sampling over the design space is not certain, but with a sufficient amount of points, it will converge to a homogeneous distribution of points.

A possible sampling from this method is given by 2.2, where each axis represent its own variable and X is the chosen sampling point for the two variables.

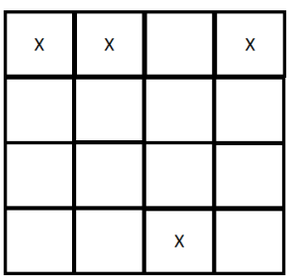

Figure 2.2: Distribution of points for a random sampling. Each variable can have four different values.

2.3.2 Latin hypercube sampling

A Latin hypercube sampling (LHS) ensures that a sample does not appear more than once in the design space. To make a Latin hypercube space, the number of input variables M is chosen and Np is the number of sampling points. Each variable M is divided into Np equal probable intervals. The sampling space is designed in the way that only one of every sample is the only one in its axis-aligned hyperplane containing it. (Iman et al. (1980)). The total number of combinations for sampling is then given by:

$$\prod_{Np=1}^{M-1} M - Np = M!^{Np-1} \tag{2.6}$$

The advantage of this method is that it ensures a more orderly random sampling space. Another advantage is that the sampling space does not increase when additional input variables are chosen, since each variable is chosen from a subset Np with equal likelihood to be found.

The figure below shows a possible Latin hypercube sampling for two variables. X is the chosen sampling point.

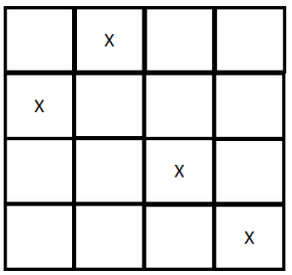

Figure 2.3: Distribution of points for two variables with four different possible values using a LHS method

2.3.3 Orthogonal sampling

Orthogonal sampling (OS) is an extension of LHS. It divides the sampling space into equal dense subspaces. The space must also fulfill the requirement of being a Latin hypercube. If the number of subspaces in OS is equal to one, the OS method becomes ordinary LHS. All the subspaces must be generated simultaneously since they must have the same density and fulfill the requirement of being a Latin hypercube.

The figure below shows a possible orthogonal sampling, where the number of subspaces is set to 4, X is the chosen sampling point for the two variables.

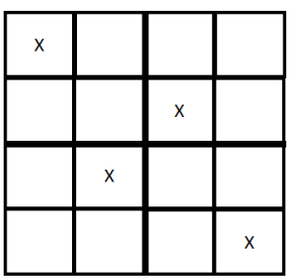

Figure 2.4: Distribution of points for two variables with four different possible values when using four subspaces in a orthogonal sampling method.

The orthogonal sampling requires to sample all of the subspaces simultaneously without overlapping the points. The total number of samples is always:

$$OS = n * k^N \tag{2.7}$$

Where N is the number of independent variables, n is the number of samples in each subspace and k is the total number of subspaces. The computational power in this method is also severely restricted by the number of components and the number of subspaces.

2.3.4 Regular grid

A regular grid (RG) is gven by equally spaced point between the design space. If two points per variable is used, and the number of variables is 3 it will result in 2^3 points in total.

Table 2.1: Total number of sampling points with a regular grid for three variables (a,b and c). Two values (1 and 2) for each variable is used.

a1	b1	c1
a1	b1	c2
a1	b2	c1
a1	b2	c2
a2	b1	c1
a2	b1	c2
a2	b2	c1
a2	b2	c2

The total number of sampling points when using k points for each variable n is then:

$$S = k^n \tag{2.8}$$

2.4 Artificial neural networks as surrogate models

A neural network (ANN) fitting is used in this project to predict the variables Y, from a set of inputs X'. The only output that the neural network if fitted to in this project is the outlet pressure p_{out}. An artificial neural network consist of a network that tries to map a certain input to a certain output. The network achieves this by a self-organizing process (Rojas (2013)). For the case study of the reaction section, we are interested in linking the reduced number of independent variables to the outlet pressure. The networks behaves as "mapping machines" from input X' to output Y. Some nodes produce values that are sent as arguments to other nodes in the network. As long as the number of layers in the cascade-network is greater or equal to two; the network can map any input-output relationship arbitrary well if it is given enough hidden nodes(Mathworks (2010)).

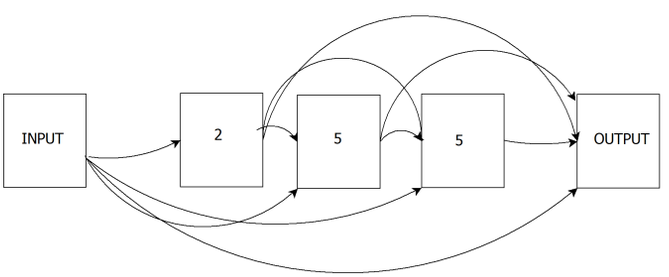

Figure 2.5: Structure for the cascade forward neural network. The numbers in the figure represent the number of nodes in that layer. [2,5,5] nodes are used in each layer respectively.

Figure 2.5 show the structure of the cascade forward neural network that is used in this project. Every point in the structure has a direct link to any of the other layers.

A distinguishing between X and X' must be made. X is the sampled space from the simple ammonia loop. X' is the input that the neural network (NN) is using. If the NN is using the resulting loadings from the PLS, then X and X' is different.

2.5 Standardisation of a data set

Its useful to scale the design space properly before fitting the PLS to it, as mentioned in (Straus and Skogestad (2017)). Otherwise the first component in PLS will point toward the space. The equation for standardizing a space is given by:

$$U_s = \frac{(U - \mu_{U)}}{\sigma_U} \tag{2.9}$$

U_s is the scaled design space, U is the design space, μ is the mean value and σ is the standard deviation in the design space U with respect to component u.

Chapter 3

How scaling and sampling size influences components in PLS

3.1 Experimental procedure

PLS regression for each sampling method was first done in MATLAB without standardisation of the output (p_{out}). The output was scaled with nominal values. The procedure for the regression was as follows:

A design space was chosen for each sampling method and used as input for the PLS regression. The output for the PLS was scaled with its initial values. The function that was used for PLS regression is given by: *plsregress*. The input for *plsregress* was X and Y. X and Y are the respective input and output of the original model. The regression was done for increasing number of design points for every sampling method. The tables below show the different options the PLS was executed on:

Table 3.1: Number of points for both LHS and RND sampling. $[S_1 - S4]$ is the notation used throughout the report

S1	S2	S3	S4
2000	4000	8000	16000

Table 3.2: The number of points for orthogonal sampling. N is the number of points in each subspace, k is the number of subspaces and n is the dimensionality.

	S1	S2	S3	S4
k	2	2	2	3
n	10	20	40	5
P	10240	20480	40960	295245

Table 3.3: The number of points for sampling with regular grid. Np is the number of values for every variable. X is the number of independent variables.

	S1	S2
Np	2	3
X	10	10
P	1024	59049

3.2 Results from scaling without Standardization

The values for the loadings in X from the *plsregress* when using LHS sampling is given in table 3.4 and 3.5. Comp1-Comp4 represents the components obtained by PLS regression.

Table 3.4: xloads in PLS when LHS is used with 4000 points and output was scaled with initial values

Comp1	Comp2	Comp3	Comp4	Comp5	Comp6	Comp7	Comp8	Comp9	Comp10
1,00	0,26	0,23	0,01	0,13	0,00	-0,11	-0,39	0,22	-0,15
0,00	-0,37	-0,65	-0,33	1,00	-0,01	0,21	-0,39	0,94	0,81
-0,32	0,61	0,48	0,35	0,78	-0,14	-0,19	-0,60	0,77	-0,84
-0,09	0,49	0,37	0,90	-0,70	-0,24	-0,45	-0,37	0,38	1,00
-0,05	-0,60	0,33	-0,34	-0,65	-0,70	0,67	-1,00	-0,03	-0,12
-0,09	-0,50	-0,35	-0,59	-0,33	0,43	-1,00	-0,84	-0,16	-0,18
-0,09	0,46	1,00	-1,00	0,46	0,44	0,12	-0,24	-0,56	0,57
0,02	-0,29	0,53	-0,95	-0,36	-0,46	-0,41	0,81	1,00	-0,04
0,05	-1,00	0,55	0,61	0,90	-0,50	-0,43	0,14	-0,64	0,14
-0,01	0,84	-0,60	-0,58	0,28	-1,00	-0,31	-0,04	-0,65	0,07

Table 3.5: xloads in PLS when LHS is used with 8000 points and the output was scaled with initial values

Comp1	Comp2	Comp3	Comp4	Comp5	Comp6	Comp7	Comp8	Comp9	Comp10
1,00	0,02	-0,17	0,07	0,12	-0,33	0,32	0,01	-0,26	0,05
0,00	0,89	-0,74	0,20	-0,49	-0,70	-0,92	-0,43	-0,26	0,22
-0,32	-0,58	-0,57	0,48	-0,06	-0,78	1,00	-0,13	-0,57	-0,09
-0,09	0,96	-1,00	-1,00	0,21	0,26	0,43	0,31	-0,04	0,20
-0,05	0,92	0,48	0,41	0,07	0,66	0,36	-0,34	-0,46	1,00
-0,09	1,00	0,64	0,13	0,05	-1,00	0,03	1,00	0,03	0,15
-0,09	-0,74	-0,04	-0,21	1,00	-0,54	-0,57	-0,16	-0,27	0,45
0,02	-0,54	0,35	-0,63	-0,39	-0,73	0,37	-0,45	0,68	0,69
0,04	-0,94	-0,73	0,54	-0,17	0,36	-0,18	0,74	0,39	0,72
-0,01	0,87	-0,36	0,65	0,52	-0,16	0,26	-0,38	1,00	-0,15

Table 3.5 shows the loadings of x from PLS when 8000 LHS points are sampled. The coefficients of the loadings in the first components is low compared to the other components.

Table 3.6: The differences in the loadings of X when 8000 and 4000 points is used for a Latin hypercube sampling

Comp1	Comp2	Comp3	Comp4	Comp5	Comp6	Comp7	Comp8	Comp9	Comp10
0,00	0,24	0,40	-0,05	0,01	0,34	-0,44	-0,40	0,47	-0,20
0,00	-1,26	0,09	-0,53	1,49	0,69	1,13	0,04	1,19	0,59
0,00	1,19	1,06	-0,14	0,83	0,65	-1,19	-0,47	1,34	-0,75
0,00	-0,47	1,37	1,90	-0,90	-0,50	-0,88	-0,68	0,41	0,80
0,00	-1,52	-0,15	-0,76	-0,72	-1,36	0,31	-0,66	0,43	-1,12
0,00	-1,50	-0,99	-0,72	-0,38	1,43	-1,03	-1,84	-0,19	-0,33
0,00	1,20	1,04	-0,79	-0,54	0,98	0,69	-0,08	-0,29	0,12
0,00	0,25	0,18	-0,32	0,03	0,27	-0,78	1,26	0,32	-0,73
0,00	-0,06	1,29	0,07	1,07	-0,86	-0,25	-0,60	-1,03	-0,58
0,00	-0,03	-0,24	-1,23	-0,24	-0,84	-0,58	0,35	-1,65	0,22

From table 3.6, the loadings of X do not change for the first component when the sampling space increases. All the other components change significantly. The first components also represent most of the explained variance. It was there for necessary to investigate if the plsregress-function in MATLAB generated the first component in the direction of the sampling space. If this was the case, then the pls regression will not represent the true component. The input for the PLS-function was X, Y and Np. Where X is the design space, Y is the output from the design space in the original model and Np is the number of components in the PLS regression model.

3.3 Results from scaling with standardization

The simulation was done again with standardization of the output Y. Both the input and output values for the simple ammonia loop were now standardized. The PLS regression was done again for the different combination of the sampling methods. The loadings of X and the explained variance for each sub case was saved. The new loadings of X are given in the table below. Only the loadings of X for a Latin hypercube sampling is shown.

Table 3.7: xload for 10 different components with 8000 points sampled using LHS

Comp1	Comp2	Comp3	Comp4	Comp5	Comp6	Comp7	Comp8	Comp9	Comp10
1,00	-0,01	0,29	-0,06	0,14	0,42	-0,08	0,08	0,07	0,09
0,00	0,54	-0,36	0,35	0,11	1,00	-0,20	-0,23	-0,68	-0,71
-0,32	-0,09	1,00	0,21	0,15	0,86	-0,45	-0,10	0,03	0,39
-0,09	0,95	-0,05	-0,28	-0,28	0,31	-0,17	1,00	0,06	0,17
-0,05	-0,56	0,59	-0,23	-0,04	-0,08	0,51	0,46	-0,17	-1,00
-0,09	-0,45	-0,26	-1,00	0,71	0,32	-0,49	-0,02	-0,26	0,07
-0,09	0,44	-0,09	-0,04	0,79	0,47	0,55	-0,13	1,00	-0,28
0,02	1,00	0,47	-0,62	-0,24	-0,38	0,36	-0,60	-0,28	-0,03
0,05	0,43	0,23	0,22	0,56	-0,98	-1,00	0,05	0,15	-0,48
0,00	0,18	0,10	0,32	1,00	-0,36	0,65	0,27	-0,68	0,46

The table 6.5 shows the loadings of X in a PLS regression that tries to explain the outlet pressure p_{ut}.

Table 3.8: xload for 10 different components with 4000 points sampled using LHS

Comp1	Comp2	Comp3	Comp4	Comp5	Comp6	Comp7	Comp8	Comp9	Comp10
1,00	0,27	0,37	0,00	-0,02	-0,14	0,15	-0,10	-0,28	0,14
0,00	0,26	-0,19	0,46	-0,48	1,00	0,47	-0,90	-0,33	-0,55
-0,32	0,98	0,58	-0,66	-0,26	-0,46	0,69	-0,26	-0,65	0,26
-0,09	-0,73	1,00	0,22	-0,75	-0,10	-1,00	-0,16	-0,44	-0,11
-0,05	0,59	0,50	1,00	0,05	-0,38	0,34	0,88	0,05	-0,86
-0,09	0,99	-0,36	0,95	0,57	-0,16	-0,74	-0,54	-0,34	0,49
-0,09	-1,00	0,88	0,42	1,00	0,06	0,61	-0,55	-0,03	0,22
0,02	0,69	0,62	-0,11	-0,20	-0,23	-0,17	-1,00	1,00	-0,16
0,05	-0,34	-0,51	-0,50	0,51	-0,64	-0,26	-0,70	-0,33	-1,00
0,00	0,69	0,64	-0,73	0,79	0,79	-0,58	0,44	-0,08	-0,32

Table 3.8 shows the loadings of X for all ten components in PLS that tries to explain the outlet pressure, p_{out}. The coefficients for the first components is much lower than for the other components.

Table 3.9: Difference in the loadings of X for 8000 and 4000 points sampled using LHS

Comp1	Comp2	Comp3	Comp4	Comp5	Comp6	Comp7	Comp8	Comp9	Comp10
0,00	-0,28	-0,07	-0,06	0,16	0,57	-0,23	0,17	0,35	-0,05
0,00	0,29	-0,17	-0,10	0,59	0,00	-0,66	0,67	-0,35	-0,16
0,00	-1,07	0,42	0,87	0,41	1,32	-1,14	0,15	0,68	0,13
0,00	1,68	-1,05	-0,50	0,47	0,41	0,83	1,16	0,50	0,28
0,00	-1,15	0,09	-1,23	-0,09	0,30	0,18	-0,42	-0,22	-0,14
0,00	-1,44	0,10	-1,95	0,14	0,49	0,24	0,52	0,07	-0,43
0,00	1,44	-0,97	-0,46	-0,21	0,41	-0,07	0,42	1,03	-0,51
0,00	0,31	-0,14	-0,51	-0,04	-0,15	0,53	0,40	-1,28	0,14
0,00	0,77	0,74	0,72	0,05	-0,35	-0,74	0,75	0,49	0,52
0,00	-0,51	-0,54	1,05	0,21	-1,15	1,22	-0,17	-0,60	0,78

As seen in table 3.9, the first component did not change even though standardized scaling of input and output for the PLS is used. The first components do not change for either case, even though the design space increases. To be certain that the components are not generated in the direction of the space, only standardization of the output for PLS regression was used in the rest of the project.

Chapter 4

Performance of the complete surrogate model

4.1 Experimental procedure

The MATLAB script for the simple ammonia synthesis loop was then modified to include a cascade-forward neural network fitting. The cascade-forward neural network contained three layers of respective size of [2 , 5 ,5] nodes in each layer. A variable transformation was performed for the artificial neural network. This was done by multiplying the loadings for the three-different linear independent variables with the corresponding input matrix. Four different combinations of the independent variables was used to compare the performance of the overall surrogate model.

An investigation of what factors affected the performance of the complete surrogate model was done. The MATLAB script 6.1 was used to do this analysis. The design space (input, X) was first sampled using three different methods, LHS, RG and OS. A validation of the design space (output, Y) was performed. Both X and Y were then standardized and a PLS regression was conducted of X on Y (6.1). The number of independent variables the neural network was fitted to was first reduced through the application of PLS. This was done by variable transformation. The loadings of X from the PLS regression was multiplied by the standardized sampling space. The figure below shows the three different combinations that were tested:

Table 4.1: The different combination of the linear independent variables

	Comb 1	Comb 2	Comb 3	Comb 4
Variable	1	1	1	1
Variable	2	3	4	5
Variable	10	9	8	7

The neural network was then fitted for all the sub cases. 75% of the data was used for the training set, 15% for the validation and 15% for the testing. The number of variables was reduced to three since this maintained a good accuracy according to Straus and Skogestad (2017). Another space was then sampled for validating the performance of the surrogate model. The validation space contained 500 000 random points. The space was standardised according to equation 2.9.

All of the networks that were trained was validated. The validation step was included to see the performance of the surrogate model compared to the original model.

4.2 Results from surrogate model performance

Only the results for Latin hypercube, Orthogonal and random sampling was used for the rest of the project. The regular grid method was not conducted further in this project. The reason for this was that the number of points increased exponentially fast when additional points for each variable was included(2.8).

4.2.1 Relative average error

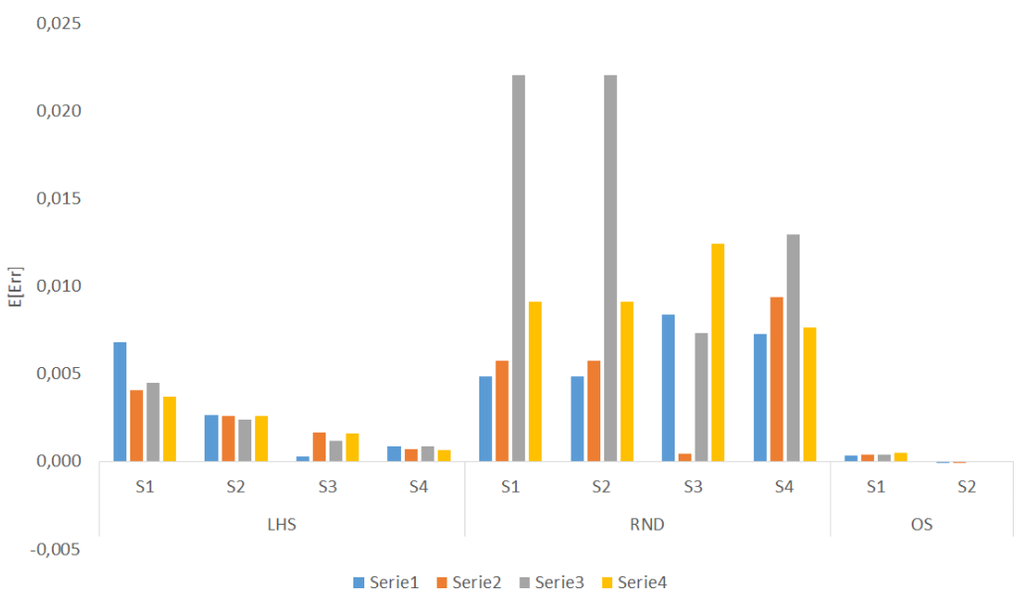

Figure 4.1: Comparison between the average values of the absolute relative errors for different sampling methods, points and combination of the linear independent variables

The figure 4.1 shows that the average error for LHS decreases when the number of points in the design space increases. The random sampling method shows no sign of a decreasing average error when the design space increases. This leads to the conclusion

that a random sampling method will not ensure relative low average error for the surrogate model, even though the design space increases. Comb 3 in the random sampling has the highest average error for S1, S2 and S4. The orthogonal sampling method gives low average error for all different combination of the independent variables. The relative average error of the surrogate model increases slightly with increasing number of points for this method.

4.2.2 Standard deviation of the error

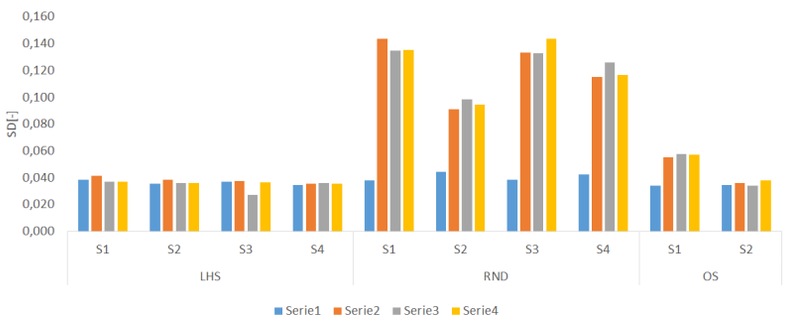

Figure 4.2: Comparison between the standard deviation(SD) of the errors for different sampling methods, points, and combinations of the independent variables.

Diagram 4.2 shows evidence that the worst method of sampling the data for to surrogate model is by random sampling. This argument only holds for the Comb 2, Comb 3 and Comb 4. The standard deviation is significantly higher than the other methods and random sampling should therefore not be used. The standard deviation for OS decreases slightly when sampling option S2 is used. The standard deviation in the LHS does not change when the design space increases.

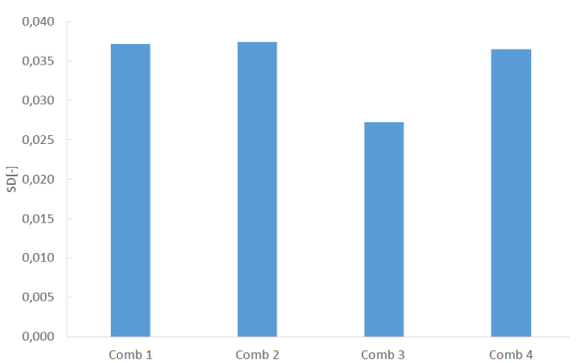

Figure 4.3: Comparison between the standard deviation(SD) of the errors for a LHS sampling using different combination of linear independent variables with 8000 points.

The diagram in 4.3 shows the standard deviation for four different combination of three independent variables. 8000 sampling point are used in this analysis. Combination 3 (Comb 3) has the lowest standard deviation in this case.

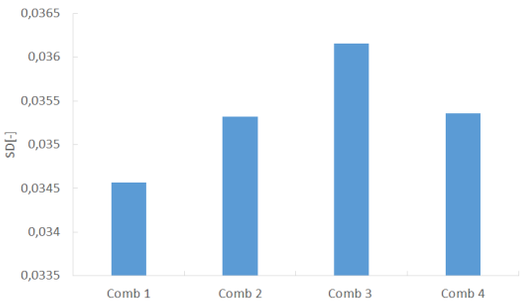

Figure 4.4: Comparison between the standard deviation(SD) for a LHS sampling using different combination of linear independent variabes with 16000 points.

The diagram 4.4 shows that combination 1 has the lowest standard deviation and combination three has the highest. Comb 3 when 8000 LHS points was sampled has the lowest standard deviation in the average error but it had the highest value for 16000 points. One of the explanation of these differences is that there is no consistency between the best combination of the independent variables when the sampling space increases. This is because the initialization of a neural network is random. The initialization of the neural network will change every time.

4.2.3 Maximum absolute error

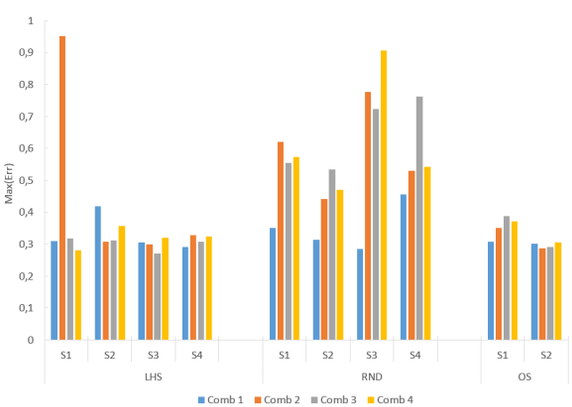

Figure 4.5: The maximum error on the performance for different sampling methods, sizes and different combinations of the independent variables.

The diagram 4.5 shows the different maximum error in percent for the different neural network fittings. Random sampling method showed significantly higher maximum error than OS and LHS. The maximum absolute error for LHS and RND did not show any sign of decreasing when the sampling space was increased. The maximum absolute error for the orthogonal sampling method decreased slightly from S1 to S2.

4.2.4 Confidence interval

The columns in 4.2 represent the upper and lower values of the average relative errors for different sampling sizes in a 95% confidence interval. The rows are the different combinations of the independent variables. a_i corresponds to sampling size i.

Table 4.2: Confidence interval of the error for LHS. All of the confidence intervals are scaled with 10^3.

	$\frac{a_1}{2}$	$(1-\frac{a_1}{2})$	$\frac{a_2}{2}$	$(1-\frac{a_2}{2})$	$\frac{a_3}{2}$	$(1-\frac{a_3}{2})$	$\frac{a_4}{2}$)	$(1-\frac{a_4}{2})$
Comb 1	8,05	5,63	3,75	1,55	1,09	-0,54	2,55	-0,84
Comb 2	5,40	2,79	3,81	1,43	2,47	0,83	2,47	-0,99
Comb 3	5,67	3,36	3,52	1,28	1,81	0,62	2,62	-0,92
Comb 4	4,86	2,54	3,74	1,51	2,40	0,80	2,42	-1,04

Table 4.2 shows that all possible combination of the linear independent variables has zero average error in the 95% confidence interval. This was only the case for 16000 points.

The columns in table 4.3 represent the upper and lower value of the average relative error for different sampling size in a 95% confidence interval. The rows are the different combinations of the independent variables. a_i corresponds to the upper value in the confidence interval. $1-a_i$ corresponds to the lowest value in the confidence interval. The

Table 4.3: Confidence interval of the error for random sampling. All of the confidence intervals are scaled with 10^3.

	$\frac{a_1}{2}$	$(1-\frac{a_1}{2})$	$\frac{a_2}{2}$	$(1-\frac{a_2}{2})$	$\frac{a_3}{2}$	$(1-\frac{a_3}{2})$	$\frac{(a_4}{2})$	$(1-\frac{a_4}{2})$
Comb 1	6,53	3,18	14,59	11,83	9,25	7,57	7,96	6,65
Comb 2	12,03	-0,54	21,03	15,39	3,39	-2,45	11,16	7,59
Comb 3	27,97	16,16	18,73	12,64	10,28	4,47	14,91	11,02
Comb 4	15,03	3,19	13,14	7,29	15,57	9,28	9,44	5,83

Table 4.3 shows that only two neural network fittings (Comb 2 for S1 and Comb 2 for S3) has zero average relative error included in the confidence interval of 95%.

The columns in the table 4.4 represents the upper and lower value of the average relative error for different sampling size in a 95% confidence interval. The rows are the different combinations of the independent variables. $\frac{a_i}{2}$ corresponds to the upper value in the confidence interval. $1-\frac{a_i}{2}$ corresponds to the lowest value in the confidence interval.

Table 4.4: Confidence interval of the error for OS sampling. All the confidence intervals are scaled with 10^3.

	$\frac{a_1}{2}$	$(1-\frac{a_1}{2})$	$\frac{a_2}{2}$	$(1-\frac{a_2}{2})$
Comb 1	0,80	-0,14	0,42	-0,52
Comb 2	1,15	-0,38	0,44	-0,55
Comb 3	1,19	-0,40	0,51	-0,42
Comb 4	1,29	-0,29	0,52	-0,52

The table shows that all of the different combinations in every sampling space contains an average relative error of zero, in the 95% confidence interval. This is ensuring since it gives a high probability that a surrogate model that is sampled with OS contains zero average relative error in the 95% confidence interval.

Chapter 5

Discussion

From the comparison between the different scaling methods used, it is clear that the first component in the PLS regression is not in the direction of the space. The reason for this was that the loadings of X did not change when standardized scaling was used. The results from the standard deviation (4.2) and maximum absolute error (4.5) for the different sampling schemes indicate that random sampling is the least favorable mapping to use. This result agrees with the argument that the random sampling technique does not cover the whole design space. The worst combination of variables for the random sampling is with Comb 3.

Most of the variance in the PLS regression is explained in the first component. Its therefore, not surprising that different combination[Comb1-Comb4] had the same performance for the output. It concludes that the first components in the PLS regression is the most important independent variable. The first component should therefore always be included in the combination of the reduced number of independent variables a neural network is fitted to. Only the fit in the sampling space increases when additional components are used. The performance for the output does not change when this is done.

Orthogonal sampling is the best option to use for generating the design space as seen in table 4.4. It includes zero average error in its 95% confidence interval for all different combination of the independent variables and sampling sizes. It also had the lowest relative average error of the three methods that was tested. If more than two subspaces are used for OS, the number of points in OS space becomes too high. The reason is that the number of points increases exponentially fast, as seen from equation 2.7.

Latin hypercube sampling is a better alternative than random sampling for the performance of the surrogate model. This can be seen in the differences of maximum absolute error and the standard deviation (See figures: 4.2 and 4.5). The Latin hypercube method for significant number of points (16000 LHS points) have similar average error as OS. LHS also includes zero average error in its 95% confidence interval for all combinations

of the independent variables when the sampling size was 16000.

Orthogonal sampling with two subspaces should be used for generating the surrogate model. It always includes zero average relative error in its confidence interval, for the combinations [S1-S2] in this project. The main reason for this is that the orthogonal method of sampling the data gives a very good representation of the real variability of the original variables. It ensures that the whole sampling space is covered and therefore has the best performance of the three different methods that were compared.

Chapter 6

Conclusion

From the analysis of the loadings of X and the explained variance from the PLS regression; the first component in PLS had the greatest influence. The first component should always be included in the reduced independent variables the neural network it fitted from. None of the different combination of the independent variables showed better performance.

Orthogonal sampling was the best sampling method on the performance for the surrogate model of the reaction section. The random sampling has relatively high standard deviations and maximum absolute error. Most of the surrogate models that were fitted using a random design space did not include zero average error in its 95% confidence interval.

Not more than two subspaces should be used in the Orthogonal sampling space. If more subspaces are used, the sampling size will become too big for validation and the computational time for generating the surrogate model will be time consuming.

Future work is to study how the performance of the surrogate model change when the sampling space in the most important variables for the outlet pressure increases. Different combinations of the number of neurons in each hidden layer for the neural network can given different performance on the surrogate model and should be investigated.

22

Bibliography

Forrester, A. I., Keane, A. J., 2009. Recent advances in surrogate-based optimization. Progress in Aerospace Sciences 45 (13), 50 – 79.
URL http://www.sciencedirect.com/science/article/pii/S0376042108000766

Iman, R. L., Davenport, J. M., Zeigler, D. K., 1980. Latin hypercube sampling (program user's guide).[lhc, in fortran]. Tech. rep., Sandia Labs., Albuquerque, NM (USA).

Mathworks, 2010. cascadeforwardnet. Cascade-forwardnet was introduced in 2010b.
URL https://se.mathworks.com/help/nnet/ref/cascadeforwardnet.html

Rojas, R., 2013. Neural networks: a systematic introduction. Springer Science & Business Media.

Straus, J., Skogestad, S., 2016. Minimizing the complexity of surrogate models for optimization. In: Kravanja, Z., Bogataj, M. (Eds.), 26th European Symposium on Computer Aided Process Engineering. Vol. 38 of Computer Aided Chemical Engineering. Elsevier, pp. 289 – 294.
URL http://www.sciencedirect.com/science/article/pii/B9780444634283500539

Straus, J., Skogestad, S., 1 2017. Variable reduction for surrogate modelling, submitted to Foundations of Computer Aided Process Operations 2017.

Wold, S., Martens, H., Wold, H., 1983. The multivariate calibration problem in chemistry solved by the PLS method. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 286–293.
URL http://dx.doi.org/10.1007/BFb0062108

Appendix

6.1 Main script for generating the surrogate model

```
%% Simple Ammonia Loop.
% This program can be used to evaluate the fitting of neural networks to
% the defined components for the reaction section of the simple ammonia
% loop. It can either work with loaded data, or perform the data anlysis
% beforehand and then fits neural networks to the problem. The neural
% netowrk fitting is done using the GPU to increase the speed of the
% fitting.
%
%  The components in this system are defined as
%    x(1) - Hydrogen
%    x(2) - Nitrogen
%    x(3) - Ammonia
%    x(4) - Argon
%    x(5) - Methane

% Clearing of the workspace and the memory, closing of all open
% windows/figures
clc, clear variables, close all

% Start of the timer for the overall function file
time.in.overall = tic;

% Definition of the numbe rof reactors in the CSTR cascade
Reac1_n  = 10;
Reac2_n  = 10;
SplinVer = 3;

% Loading of the needed parameters
ParSamp

% Definition of the inlet values for the makeup section
def.M.FMake = [1823.90; 27.32e5; 7.495;...                  Feed Stream Q [mol/s], p [Pa], and T [C]
               [73.45; 24.03; 1.6; 0.41; 0.51]/100];...     Feed Stream mole fractions
def.M.HEx1  = [9274.61; 10e5; -10;...                       Water Stream HEx1 Q [mol/s], p [Pa], and T [C]
                ones(5,1)];...                              Water Stream HEx1 mole fractions
def.M.FMix  = [6756.71; 126.40e5; 25.61;...                 Mixer Stream Q [mol/s], p [Pa], and T [C]
                0.6959; 0.1701; 0.017; 0.0554; 0.0616];...  Mixer Stream Q mole fractions

defRO.FMake = def.M.FMake./scl.safe.Stream;
defRO.HEx1  = def.M.HEx1./scl.safe.Stream;
defRO.FMix  = def.M.FMix./scl.safe.Stream;

% Definition of the inlet values for the reaction section
def.R.FReact = [8571.84; 138.97e5; 50.78;...
                 0.7005; 0.1755; 0.0167; 0.0512; 0.0561];
def.R.Val1   = 832.4;...                                    Split Valve Q [mol/s]

defRO.FReact = def.R.FReact./scl.safe.Stream;
defRO.Val1   = def.R.Val1./scl.safe.Stream(1);

% Definition of the inlet values for the separation section
def.S.FSep = [7718.20; 129.65e5; 55.59;...                  Entering Stream Q [mol/s], p [Pa], and T [C]
              0.6186; 0.1510; 0.1259; 0.0495; 0.0550];...   Entering Stream Mole fractions -
def.S.HEx6 = [11184.22; 3.828e5; 10;...                     Water Stream HEx6 Q [mol/s], p [Pa], and T [C]
               ones(5,1)];...                               Water Stream HEx6 mole fractions

defRO.FSep  = def.S.FSep./scl.safe.Stream;
defRO.HEx6  = def.S.HEx6./scl.safe.Stream;

% Definition of version histories
Ver.Save   = 'n';       % loading saved data or not
Ver.Sol    = ['y' 'n']; % Question whether solution should be used or not for the initial conditions
Ver.Lid    = 'n';
Ver.Approx = 'n';       % Approximation on or not

%run this code with different CG, LHS, OS, RND.
% Definition of the initial conditions and the anonymous functions
```

```
[initMakeUp,    ConAnMakeUp,...
 initSep,       ConAnSep,...
 initReact,     ConAnReact]...
= InitSamp(par,scl,con,ther,def,defRO,itg,dirmain,Ver);

% Definition of the options used in each solver 'iter-detailed'
options = optimset('Algorithm','trust-region-dogleg','Display','off',...
                   'Jacobian','on','MaxIter',1e3,'MaxFunEvals',1e5,...
                   'TolCon',1e-8,'TolFun',1e-8);

% Addition of the  Path direction of the unit operations
UO = [dirmain,'\UO2 Full Order\New Scaling'];
addpath(UO)

% Addition of an additional extra line break
fprintf('\n\n')

% Definition of the varied variables
indin.R = 1:10;

% Definition of the anonymous function
SampAnReact  = @(initReact,A)  SampReact_MolFlow (initReact,A,options,par.R,scl.R,con,ther,def.R,itg);

% Definition of the calculated values and indices used in as outlets
ncal = [];
indout.React  = 9:16;                              % Reactor Section with n CSTR in the cascade
olol = 1;
switch Ver.Save
case 'y'
    % Loading of the values and definition of the sample length
    load Reac20160707.mat
    nsamp.R = length(A.R);

end

% Calculation of the number of varied variables
n_var = numel(indin.R);
CG = 2^n_var;


% Presolving of the system
A.init = [def.R.FReact(2:3)' def.R.FReact(4:8)'*def.R.FReact(1) par.R{3}.ratioFO(1:2)' par.R{17}.Tref];
[x.init,~,~,exitflag.init,time.init] = fsamp(SampAnReact,A.init,initReact,ncal);



% Printing of the results
display('The initialization has been performed.')

% Definition of auxiliarry variables
Onlj = ones(1,itg.j);

% Definition of the component molar flowrates
B.samp.init = A.init(:,3:7);
x.samp.init = x.init(:,indout.React(4:8)).*(x.init(:,indout.React(1))*scl.Q*Onlj);


Diff.init = x.samp.init-B.samp.init;
ex_o_R.init = Diff.init.*(1./par.R{13}.stoi');
x.PLS.init =  [1 1 ex_o_R.init(1)];

%% Preequesite: Definition of the grid for the Reaction Sections

% Definition of the range for plotting
lbmub = zeros(3,numel(A.init));
lbmub(1,:)  = A.init.*[1 0.8 0.875 0.85 0.5 0.6 0.6 1 1 1] - [5e5 zeros(1,6) 3 10 10]; % [3e5 zeros(1,6) 2 10 5];
lbmub(2,:)  = A.init;
lbmub(3,:)  = A.init.*[1 1.2 1.125 1.15 2.0 1.5 1.5 1 1 1] + [5e5 zeros(1,6) 3 10 10];

%% This part Petter has edited, to run the regression with different options




M=4 ; % Number of methods; RND,CG,LHS and OS.
N=7; % Number of iterations for each variable, with different number of point in each time.

Tmat=cell(M,N); % En cell med  [comp x number of iterations ]
% Tmat(:)={0};
Tmat{1,1}=500;
Tmat{2,1}=500;
% Tmat = {100, 500, 1000, 5000, 1e4, 5e4};
    for j = 2:N
    Tmat{1,j}=Tmat{1,j-1}*2;
    Tmat{2,j}=Tmat{2,j-1}*2;
    end
    Tmat{3,1}={2,10};
```

```
159       Tmat{3,2}={2,20};          %OS
160       Tmat{3,3}={2,40};          %OS
161       Tmat{3,4}={3,5};
162
163       Tmat{4,1}=2;               %CG
164       Tmat{4,2}=3;
165   %     Tmat{4,3}=4;
166
167   Nmax = size(Tmat,2);
168   %The explained variance in 5 different ways, store it in variable explsum
169   explsumLHS = cell(Nmax,3);
170   explsumOS  = cell(Nmax,3);
171   explsumCG  = cell(Nmax,3);
172   explsumRND = cell(Nmax,3);
173
174   xloadLHS   = cell(Nmax,3);
175   xloadCG    = cell(Nmax,3);
176   xloadOS    = cell(Nmax,3);
177   xloadRND   = cell(Nmax,3);
178
179   % LINE NUMBER 1 and 2 I HAD TO CHANGE
180   StoreErrIt = cell(size(Tmat,1),Nmax);
181   StoreNetIt = cell(size(Tmat,1),Nmax);
182
183   % Loading of the validation space
184   load('ValSpace.mat')
185
186   % Storing the variables
187   A.val = Aval(1:end-CG,:);
188   x.val=xval(1:end-CG,1:16);
189
190   %%First run LHS,CG,OS and then RND.
191   for i = 1:4
192       if i==1          % LHS = number of points
193           optS.type = 'LHS';
194           optS.val = 'n';
195           optS.Np = Tmat{i,1};
196       elseif i==2      % CG = Np^10
197           optS.type = 'RND';
198           optS.Np = Tmat{i,1};
199       elseif i==3      % OS = (Ns^10)*Np
200           optS.type = 'OS';
201           optS.Np = Tmat{i,1}{2};
202           optS.Ns = Tmat{i,1}{1};
203       else             % RND = number of points
204         optS.type='CG';
205         optS.Np = Tmat{i,1};
206       end
207
208   for j = 3:6   %: size(Tmat,2)
209       if j > 1
210           if isempty(Tmat{i,j})
211               break
212           end
213           switch i
214               case 1
215                   optS.Np=Tmat{1,j};
216               case 2
217                   opt.Np=Tmat{2,j};
218
219               case 3
220                   optS.Ns=Tmat{3,j}{1};
221                   optS.Np=Tmat{3,j}{2};
222
223               case 4
224                   optS.Np=Tmat{4,j};
225           end
226       end
227       fprintf('Iteration i = %1i and j = %1i \n',i,j)
228   %
229   %     if j == 4
230   %         switch i
231   %             case 2
232   %                 break
233   %         end
234   %     end
235
236   %%
237   switch Ver.Save
238   case 'n'
239
240   % Call of the grid definition function
241   Afull = GridDef(lbmub,indin.R,optS);
242
243   % Readjustment of the output values
244   A.R = Afull.out;
245
246   % Definition of the approximation matrix
247   A.approx = A.R(:,indin.R);
248
```

```
% Definition of the number of sampling points
nsamp.R = size(A.R,1);

%% Part a) Calculating and fitting of the Reaction Section

% Sampling of the data
time.in.React = tic;
[x.R,cal.R,b.R,exitflag.R,time.R] = fsamp(SampAnReact,A.R,x.init',ncal);
time.React = toc(time.in.React);

% Printing of the results
display('The training space has been sampled.')
end

% Definition of a local safe directory
dirsave = 'LocalSave\';
fprintf('\n\n')


% Definition of the scaling values
 scl.A = [scl.R{1}.Var(17) scl.R{1}.Var(18)*10 scl.R{1}.Var(19)*.1 ...
                  1 1 .01 10 100 scl.R{1}.Var(3)*.1];
% scl.x = [1 10 .1 1 1 .1 .1 .1];
% scl.PLS = [1 1 .1*scl.R{1}.Var(17)];
scl.A = A.init(indin.R);
scl.x = x.init(indout.React);
scl.PLS = x.PLS.init;

% Definition of the number of sampling points
nsamp.R = size(A.R,1);
% Definiton of the independent variables which are varied
A.samp.R = A.R(:,indin.R);

% Definition of the component molar flowrates
B.samp.R = A.R(:,3:7);
x.samp.R = x.R(:,indout.React(4:8)).*(x.R(:,indout.React(1))*scl.Q*On1j);

% Scaling of the independent and dependent variables and definition of
% auxiliary variables for the sampling space
OnN1.R = ones(nsamp.R,1);
A.scaled.R = A.samp.R./(OnN1.R*scl.A);
x.scaled.R = x.R(:,indout.React)./(OnN1.R*scl.x);

% Calculation of the differences in molar flow rate and the extent of
% reaction for the sampling space
Diff.R = x.samp.R-B.samp.R;
ex_o_R.R = Diff.R.*(OnN1.R*(1./par.R{13}.stoi'));
x.PLS.R =  [x.scaled.R(:,2:3) ex_o_R.R(:,1)]./(OnN1.R*scl.PLS);


%% Definiton of the independent variables which are varied
   A.samp.val = A.val(:,indin.R);
   nsamp.val = size(A.val,1);
%
%% Definition of the component molar flowrates
  B.samp.val = A.val(:,3:7);
  x.samp.val = x.val(:,indout.React(4:8)).*(x.val(:,indout.React(1))*scl.Q*On1j);
%
%% Scaling of the independent and dependent variables and definition of
%% auxiliary variables for the validation space
  OnN1.val = ones(nsamp.val,1);
  A.scaled.val = A.samp.val./(OnN1.val*scl.A);
  x.scaled.val = x.val(:,indout.React);
%% Calculation of the differences in molar flow rate and the extent of
%% reaction for the validation space
  Diff.val = x.samp.val-B.samp.val;
  ex_o_R.val = Diff.val.*(OnN1.val*(1./par.R{13}.stoi'));
  x.PLS.val =  [x.scaled.val(:,2:3) ex_o_R.val(:,1)];%./(OnN1.val*scl.PLS);

  [A.scaled.R,Var.muR,Var.std] = zscore(A.samp.R);
%  A.scaled.R(:,6:7) = A.scaled.R(:,6:7)*.1;

  A.scaled.val = (A.samp.val-OnN1.val*Var.muR)./(OnN1.val*Var.std);


  [x.PLS.R,Var.mux,Var.stdx]  = zscore(x.PLS.R);



% Definition of the problem in mole fraction
Q = sum(A.samp.R(:,3:7),2);
A.Molfrac.R = [Q A.samp.R(:,1:2) A.samp.R(:,3:6)./(Q*ones(1,4)) ...
                  A.samp.R(:,8:end)];

 Q = sum(A.samp.val(:,3:7),2);
 A.Molfrac.val = [Q A.samp.val(:,1:2) A.samp.val(:,3:6)./(Q*ones(1,4)) ...
                  A.samp.val(:,8:end)];
clear Q              % clear Q

trainR = 70/100;
```

```
valR    = 15/100;
testR   = 15/100;
[trainInd,valInd,testInd] = dividerand(nsamp.R-CG,trainR,valR,testR);



%% Part e) PLS on the values of the reaction section
% This part of this script calculates the extent of reaction for the
% different inlet conditions and then fits with PLS.


% Define the value, PLS should be fitted to (as row vector)
%   1 = Outlet p
%   2 = Outlet T
%   3 = Extent of reaction
var_fit = 1;
var_comp = 1;
k=var_comp;


label.RM = {'Pressure [hbar]'; 'Temperature [hC]'; 'Extent of reaction [kmol]'};

for l = var_fit


    % Definition of the components
    n_comp = 10;
    n_fit = 1;

    % Question, which Definition should be used
     switch n_fit
         case {1, 2}
            [A.scaled.R,Var.muR,Var.std] = zscore(A.samp.R);
             A.scaled.val = (A.samp.val-OnN1.val*Var.muR)./(OnN1.val*Var.std);

         case {3}
             [A.scaled.R,Var.muR,Var.std] = zscore(A.Molfrac.R);
             A.scaled.val = (A.Molfrac.val-OnN1.val*Var.muR)./(OnN1.val*Var.std);
     end

    % Calculation of the Partial least square regression
    [xload.R,yload.R,xscores.R,yscores.R,...
                par.PLS.R,pctvar.R,~,stats.R] = plsregress(A.scaled.R,x.PLS.R(:,n_fit),n_comp);

    % Plotting of the cumulated sums
    expvar.R = cumsum(pctvar.R,2);
    xload.R = xload.R./(ones(n_var,1)*max(abs(xload.R)));


    CombOpts = 4;
    StoreErr=cell(CombOpts,4);
    StoreNet=cell(CombOpts,4);
    for M = 1:CombOpts


            mult = xload.R(:,[1,M+1,11-M]);
%         case 10
%             mult = 1;
%     end
    C.R = A.scaled.R*mult;
    nsamp.val = size(A.val,1);
    C.val = A.scaled.val*mult;

    % Definition of network parameter
    hiddenLayerSize = [2 ones(1,2)*5];
    net = cascadeforwardnet(hiddenLayerSize,'trainlm');
    net.trainParam.max_fail = 200;
    net.trainParam.epochs = 1e5;
    net.trainParam.min_grad = 1e-9;


    % Set up Division of Data for Training, Validation, Testing
    net.divideFcn = 'divideind';
    net.divideParam.trainInd = [1:CG trainInd+CG];
    net.divideParam.valInd = valInd+CG;
    net.divideParam.testInd = testInd+CG;
    % net.trainFcn = 'trainbr'; 'trainlm'; 'trainscg'

     %Train the Network
     tic;
     [net,tr] = train(net,C.R',x.PLS.R(:,n_fit)','UseParallel','yes'); % );%  ,'UseGPU','yes');%
     toc
     StoreNet{M,1} = net;

      x.PLS.ANN(:,n_fit) = net(C.val')'.*(OnN1.val*Var.stdx(n_fit))+OnN1.val*Var.mux(n_fit);

    % Definition of the number of sampling points
    % Calculation of the results using PLS
%     x.PLS.ANN(:,n_fit) = combnet{1,k}(C.val')';
```

```
429
430       switch n_fit
431           case {1, 2}
432               x.PLS.ANN(:,n_fit) = x.PLS.ANN(:,n_fit)*scl.x(n_fit+1);
433               EoRfit(:,n_fit) = (([OnN1.val  A.scaled.val]*par.PLS.R).*(OnN1.val*Var.stdx(n_fit))+OnN1.val*Var.mux(n_fit))*scl.x(n_fit+1);
434           case {3}
435               x.PLS.ANN(:,n_fit) = x.PLS.ANN(:,n_fit)*scl.PLS(n_fit);
436               x.PLS.val(:,n_fit) = (x.PLS.val(:,n_fit).*(OnN1.val*Var.stdx(n_fit))+OnN1.val*Var.mux(n_fit))*scl.PLS(n_fit);
437               EoRfit(:,n_fit) = (([OnN1.val  A.scaled.val]*par.PLS.R).*(OnN1.val*Var.stdx(n_fit))+OnN1.val*Var.mux(n_fit))*scl.PLS(n_fit);
438       end
439
440
441
442
443
444
445       % Definition of the number of sampling points
446       % Calculation of the results using PLS
447   %
448
449       rnge.PLS = 1:nsamp.val;
450
451       % Calculation of the fitted values
452       EoRfit(:,n_fit) = [OnN1.val  A.scaled.val]*par.PLS.R;
453
454       % Calculation of the relative error
455       err.lin{l,k} = abs((x.PLS.val(:,n_fit)-EoRfit(:,n_fit))./x.PLS.val(:,n_fit))*100;
456       err.val{l,k} = (x.PLS.val(:,n_fit)-x.PLS.ANN(:,n_fit))./x.PLS.val(:,n_fit)*100;
457
458
459
460
461
462       % Calculation of the maximum, mean, and median error
463       err.comb.val = [max(abs(err.val{l,k}));
464                       mean(abs(err.val{l,k}));
465                       median(abs(err.val{l,k}))];
466       err.comb.lin = [max(abs(err.lin{l,k}));
467                       mean(abs(err.lin{l,k}));
468                       median(abs(err.lin{l,k}))];
469
470       itg.fig = l*10+k;
471   %
472       expvarc{l} = expvar.R;
473       errval{l,k} = [max(abs(err.val{l,k}));
474                      mean(abs(err.val{l,k}));
475                      median(abs(err.val{l,k}))];
476       errlin{l,k} = [max(abs(err.lin{l,k}));
477                      mean(abs(err.lin{l,k}));
478                      median(abs(err.lin{l,k}))];
479
480
481
482       StdDevVal=std(err.val{l,k});
483       meanVal=mean(err.val{l,k});
484       StoreErr{M1}=StdDevVal;
485       StoreErr{M2}=meanVal;
486       StoreErr{M3}=max(abs(err.val{l,k}));
487       StoreErr{M4}=min(abs(err.val{l,k}));
488       end
489       StoreErrIt{i,j} = StoreErr;
490       StoreNetIt{i,j} = StoreNet;
491
492       %Here I store the values of the explained variance for the four
493       %different sampling methods, when I increase the number of sampling points.
494       if i==1
495           explsumLHS{j,1} = expvar.R;
496           xloadLHS{j,1} = xload.R;
497        elseif i==2
498           explsumRND{j,1}=expvar.R;
499           xloadRND{j,1} = xload.R;
500        elseif i==3
501           explsumOS{j,1}=expvar.R;
502           xloadOS{j,1} = xload.R;
503       else
504           explsumCG{j,1}=expvar.R;
505           xloadCG{j,1} = xload.R;
506
507       end
508
509   end
510   end
511   end
512
513
514
515   % Calculation of the overall time save('AppendedData.txt', 'excel', '-ASCII', '-append');
516   time.overall = toc(time.in.overall);
517
518   save('Save_function_was_again_not_specified.mat','Tmat','StoreErrIt','StoreNetIt','explsumLHS','explsumOS','explsumRND','xloadLHS','xloadOS','xl
```

6.2 Results from Standarisation

6.2.1 Explained variance

Table 6.1: Explained variance of the components using a random sampling method

N/C	Comp1	Comp2	Comp3	Comp4	Comp5	Comp6	Comp7	Comp8	Comp9	Comp10
500,000	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
1000,000	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
2000,000	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
4000,000	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
8000,000	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
16000,000	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
32000,000	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99

Table 6.2: Explained variance of the components using a LHS method

Np/Cp	Comp1	Comp2	Comp3	Comp4	Comp5	Comp6	Comp7	Comp8	Comp9	Comp10
500,000	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
1000,000	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
2000,000	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
4000,000	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
8000,000	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
16000,000	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
32000,000	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00

Table 6.3: Explained variance of the components using orthogonal method

Np/Cp	Comp1	Comp2	Comp3	Comp4	Comp5	Comp6	Comp7	Comp8	Comp9	Comp10
10240,000	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
20480,000	0,99	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
40960,000	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00
295245,000	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00	1,00

Table 6.4: Explained variance in the components using a regular grid

Np/Cp	Comp1	Comp2	Comp3	Comp4	Comp5	Comp6	Comp7	Comp8	Comp9	Comp10
1024,000	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99
59049,000	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99	0,99

6.2.2 Xloads for a Latin hypercube sampling

Table 6.5: Xload for 10 different components from a sampling size of 8000 LHS points

Comp1	Comp2	Comp3	Comp4	Comp5	Comp6	Comp7	Comp8	Comp9	Comp10
1,00	-0,01	0,29	-0,06	0,14	0,42	-0,08	0,08	0,07	0,09
0,00	0,54	-0,36	0,35	0,11	1,00	-0,20	-0,23	-0,68	-0,71
-0,32	-0,09	1,00	0,21	0,15	0,86	-0,45	-0,10	0,03	0,39
-0,09	0,95	-0,05	-0,28	-0,28	0,31	-0,17	1,00	0,06	0,17
-0,05	-0,56	0,59	-0,23	-0,04	-0,08	0,51	0,46	-0,17	-1,00
-0,09	-0,45	-0,26	-1,00	0,71	0,32	-0,49	-0,02	-0,26	0,07
-0,09	0,44	-0,09	-0,04	0,79	0,47	0,55	-0,13	1,00	-0,28
0,02	1,00	0,47	-0,62	-0,24	-0,38	0,36	-0,60	-0,28	-0,03
0,05	0,43	0,23	0,22	0,56	-0,98	-1,00	0,05	0,15	-0,48
0,00	0,18	0,10	0,32	1,00	-0,36	0,65	0,27	-0,68	0,46

Table 6.6: Xload for 10 different components of 4000 LHS points

Comp1	Comp2	Comp3	Comp4	Comp5	Comp6	Comp7	Comp8	Comp9	Comp10
1,00	0,27	0,37	0,00	-0,02	-0,14	0,15	-0,10	-0,28	0,14
0,00	0,26	-0,19	0,46	-0,48	1,00	0,47	-0,90	-0,33	-0,55
-0,32	0,98	0,58	-0,66	-0,26	-0,46	0,69	-0,26	-0,65	0,26
-0,09	-0,73	1,00	0,22	-0,75	-0,10	-1,00	-0,16	-0,44	-0,11
-0,05	0,59	0,50	1,00	0,05	-0,38	0,34	0,88	0,05	-0,86
-0,09	0,99	-0,36	0,95	0,57	-0,16	-0,74	-0,54	-0,34	0,49
-0,09	-1,00	0,88	0,42	1,00	0,06	0,61	-0,55	-0,03	0,22
0,02	0,69	0,62	-0,11	-0,20	-0,23	-0,17	-1,00	1,00	-0,16
0,05	-0,34	-0,51	-0,50	0,51	-0,64	-0,26	-0,70	-0,33	-1,00
0,00	0,69	0,64	-0,73	0,79	0,79	-0,58	0,44	-0,08	-0,32

Table 6.7: Difference in the xloads for 8000 and 4000 Latin hypercub points

Comp1	Comp2	Comp3	Comp4	Comp5	Comp6	Comp7	Comp8	Comp9	Comp10
0,00	-0,28	-0,07	-0,06	0,16	0,57	-0,23	0,17	0,35	-0,05
0,00	0,29	-0,17	-0,10	0,59	0,00	-0,66	0,67	-0,35	-0,16
0,00	-1,07	0,42	0,87	0,41	1,32	-1,14	0,15	0,68	0,13
0,00	1,68	-1,05	-0,50	0,47	0,41	0,83	1,16	0,50	0,28
0,00	-1,15	0,09	-1,23	-0,09	0,30	0,18	-0,42	-0,22	-0,14
0,00	-1,44	0,10	-1,95	0,14	0,49	0,24	0,52	0,07	-0,43
0,00	1,44	-0,97	-0,46	-0,21	0,41	-0,07	0,42	1,03	-0,51
0,00	0,31	-0,14	-0,51	-0,04	-0,15	0,53	0,40	-1,28	0,14
0,00	0,77	0,74	0,72	0,05	-0,35	-0,74	0,75	0,49	0,52
0,00	-0,51	-0,54	1,05	0,21	-1,15	1,22	-0,17	-0,60	0,78

6.3 Analysis with LHS

Table 6.8: Statistical parameters for a Latin hypercube sampling of 2000 points(S1)

	StdDev	Mean	Max	Min x10^8
Comb 1	0,0386	0,00684	0,310	1,49
Comb 2	0,0416	0,00409	0,9518116	162,33
Comb 3	0,0368	0,00451	0,318	1,04
Comb 4	0,0370	0,00370	0,281	13,3

Table 6.9: Statistical parameters for a Latin hypercube sampling of 4000 points(S2)

	StdDev	Mean	Max	Min x 10^8
Comb 1	0,0355	0,00265	0,419	2,78
Comb 2	0,0383	0,00262	0,308	7,21
Comb 3	0,0362	0,00240	0,311	8,60
Comb 4	0,0360	0,00262	0,356	5,37

Table 6.10: Statistical parameters for a Latin hypercube sampling of 8000 points.

	StdDev	Mean	Max	Min 10^8
Comb 1	0,037	0,000272	0,306	5,08
Comb 2	0,037	0,00165	0,300	4,22
Comb 3	0,027	0,00121	0,270	6,00
Comb 4	0,037	0,00160	0,320	9,64

Table 6.11: Statistical parameters for a Latin Hypercube sampling of 16000 points

	StdDev	Mean	Max	Min x 10^8
Comb 1	0,0346	0,000854	0,290	9,57
Comb 2	0,0353	0,000740	0,329	42,9
Comb 3	0,0360	0,000853	0,308	1,92
Comb 4	0,0355	0,000689	0,324	2,64

6.3.1 Analysis with RND

Table 6.12: Statistical parameters from a random sampling, when sampled 2000 points(S1)

	StdDev	Mean	Max	Min x 10^8
Comb 1	0,0382	0,00486	0,350	10,1
Comb 2	0,143	0,00575	0,621	13,3
Comb 3	0,135	0,0220	0,554	89,3
Comb 4	0,135	0,00910	0,572	14,9

Table 6.13: Statistical parameters for a random sampling of 4000 points (S2)

	StdDev	Mean	Max	Min x10^8
Comb 1	0,0446	0,0132	0,314	4,60
Comb 2	0,0910	0,0182	0,442	7,71
Comb 3	0,0981	0,0157	0,533	31,3
Comb 4	0,0943	0,0102	0,471	16,8

Table 6.14: Statistical parameters of a random sampling of 8000 points(S3)

	StdDev	Mean	Max	Min x 10^8
Comb 1	0,0385	0,00841	0,285	2,24
Comb 2	0,133	0,000470	0,776	2,76
Comb 3	0,132	0,00737	0,722	29,8
Comb 4	0,143	0,0124	0,907	9,09

Table 6.15: Statistical parameters of a random sampling for 16000 points(S4)

	StdDev	Mean	Max	Min x 10^8
Comb 1	0,0423	0,00730	0,456	20,1
Comb 2	0,115	0,00937	0,530	13,0
Comb 3	0,125	0,0127	0,763	0,613
Comb 4	0,116	0,00764	0,541	32,1

6.3.2 Analysis with OS

Table 6.16: Statistical parameters of a orthogonal sampling for using sampling option S1

	StdDev	Mean	Max	Min x 10^8
Comb 1	0,0342	0,000330	0,308	0,513
Comb 2	0,0552	0,000389	0,351	4,29
Comb 3	0,0574	0,000394	0,388	45,0
Comb 4	0,0572	0,000450	0,371	9,28

Table 6.17: Statistical parameters of a orthogonal sampling for using sampling option S2

	StdDev	Mean x 10^5	Max	Min x 10^8
Comb 1	0,0345	-4,90	0,30219431	12,7
Comb 2	0,0361	-5,82	0,28667256	16,7
Comb 3	0,0340	4,76	0,291	12,5
Comb 4	0,0380	0,247	0,306	1,50