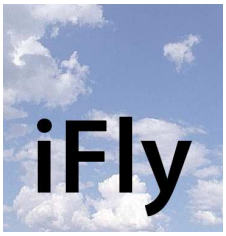


Project no. TREN/07/FP6AE/S07.71574/037180 IFLY

iFly

Safety, Complexity and Responsibility based design and validation of highly automated Air Traffic Management

Specific Targeted Research Projects (STREP)

Thematic Priority 1.3.1.4.g Aeronautics and Space

# iFly Deliverable D7.2f
# Sensitivity analysis in Monte Carlo simulation based rare event estimation

Version: 1.3

**M.B. Klompstra, G.J. Bakker and H.A.P. Blom**

NLR

Due date of deliverable: 22 January 2010
Actual submission date: 28 May 2010

Start date of project: 22 May 2007 Duration: 51 months

| **Project co-funded by the European Commission within the Sixth Framework Programme (2002-2006)** | | |
|---|---|---|
| **Dissemination Level** | | |
| **PU** | Public | ✓ |
| **PP** | Restricted to other programme participants (including the Commission Services) | |
| **RE** | Restricted to a group specified by the consortium (including the Commission Services) | |
| **CO** | Confidential, only for members of the consortium (including the Commission Services) | |

# DOCUMENT CONTROL SHEET

| | |
|---|---|
| ***Title of document:*** | *Sensitivity analysis in Monte Carlo simulation based rare event estimation* |
| ***Authors of document:*** | *M.B. Klompstra, G.J. Bakker and H.A.P. Blom* |
| ***Deliverable number:*** | *D7.2f* |
| ***Project acronym:*** | *iFly* |
| ***Project title:*** | *Safety, Complexity and Responsibility based design and validation of highly automated Air Traffic Management* |
| ***Project no.:*** | *TREN/07/FP6AE/S07.71574/037180 IFLY* |
| ***Instrument:*** | *Specific Targeted Research Projects (STREP)* |
| ***Thematic Priority:*** | *1.3.1.4.g Aeronautics and Space* |

## DOCUMENT CHANGE LOG

| Version # | Issue Date | Sections affected | Relevant information |
|---|---|---|---|
| 0.1 | 14 Mar 2008 | All | Initial draft |
| 0.2 | 19 May 2008 | All | Second draft |
| 0.3 | 30 Sep 2008 | All | Examples 1 and 2 added |
| 0.4 | 7 Nov 2008 | All | Examples 3, 4 and 5 added |
| 0.5 | 7 Apr 2009 | All | Examples 6, 7 and 8 added |
| 0.6 | 8 Feb 2010 | All | Internal discussion based |
| 0.7 | 26 Feb 2010 | All | Reorganized draft |
| 0.8 | 22 Mar 2010 | All | Updated version |
| 0.90 | 27 April 2010 | All | Updated figures, discussion, etc. |
| 0.91 | 25 May 2010 | All | Novel Section 2 & updates all |
| 1.0 | 28 May 2010 | All | Editorial update |
| 1.1 | 19 Aug 2010 | All | Editorial update |
| 1.2 | 10 Dec 2010 | All | Internal review comments incorporated |
| 1.3 | 22 Mar 2011 | All, incl Section 8 | External review comments incorporated and additional simulations included |

| | | Organisation | Signature/Date |
|---|---|---|---|
| **Authors** | M.B. Klompstra | NLR | |
| | G.J. Bakker | NLR | |
| | H.A.P. Blom | NLR | |
| **Internal reviewers** | M.H.C. Everdij | NLR | |
| | S.H. Stroeve | NLR | |
| **External reviewers** | Uwe Voelckers | Independent | |

## Executive summary

Within HYBRIDGE novel Monte Carlo simulation speed up techniques have successfully been developed and applied for rare event estimation. In iFly WP7 potential candidates are identified that are expected to provide significant room for the development of complementary speed-up techniques. Within iFly WP7 various options for improvement are identified and these are subsequently elaborated and tested within parallel studies. One of these studies is to combine *sensitivity analysis with Monte Carlo simulation based rare event estimation*, which is addressed in the current report, i.e. iFly deliverable D7.2f.

## Table of Contents

# Abbreviations

| | |
|---|---|
| $A^3$ | Autonomous Aircraft Advanced |
| ATM | Air Traffic Management |
| CLS | Classical Least Squares estimation |
| ESARR | Eurocontrol Safety Regulatory Requirement |
| HYBRIDGE | Distributed Control and Stochastic Analysis of Hybrid Systems Supporting Safety Critical Real-Time Systems Design (EC 5th Framework Programme) |
| ICAO | International Civil Aviation Organisation |
| IPS | Interacting Particle System |
| LHS | Latin Hypercube Sampling |
| LS-MP | Least Squares estimation with Moore-Penrose |
| MC | Monte Carlo |
| MLR | Multi-dimensional Linear Regression |
| NIPALS | Nonlinear Iterative Partial Least Squares |
| OAT | One-At-a-Time |
| PLS | Partial Least Squares estimation |
| PLS-N | NIPALS based PLS |
| PLS-S | SIMPLS based PLS |
| RTD | Research, Technology and Development |
| SA | Situation Awareness |
| SESAR | Single European Sky ATM Research |
| SIMPLS | Straightforward IMPlementation of a statistically inspired modification of the PLS method |
| SRS | Standard Random Sampling |
| SVD | Singular Value Decomposition |
| TOPAZ | Traffic Organization and Perturbation AnalyZer |
| WP | Work Packages |

# 1 Introduction

This section introduces this report by giving its background as output of work package 7 of the iFly project, by describing its objective, and by outlining its contents.

## *1.1 iFly project*

Air transport throughout the world, and particularly in Europe, is characterised by major capacity, efficiency and environmental challenges. With the predicted growth in air traffic, these challenges must be overcome to improve the performance of the Air Traffic Management (ATM) system. The iFly project addresses these critical issues by developing a paradigm step change in advanced ATM concept development through a systematic exploitation of state-of-the-art mathematical techniques including stochastic modelling, analysis, optimisation and Monte Carlo simulation.

The iFly project will develop a highly automated ATM design for en-route traffic, which takes advantage of autonomous aircraft operation capabilities and which is aimed to manage a three to six times increase over 2005 en-route traffic demand.

iFly will perform two operational concept design cycles and an assessment cycle comprising human factors, safety, efficiency, capacity and economic analyses. The general work structure is illustrated in Figure 1. During the first design cycle, state of the art Research, Technology and Development (RTD) aeronautics results will be used to define a "baseline" operational concept. For the assessment cycle and second design cycle, innovative methods for the design of safety critical systems will be used to develop an operational concept capable of managing a three to six times increase in current air traffic levels. These innovative methods find their roots in robotics, financial mathematics and telecommunications.

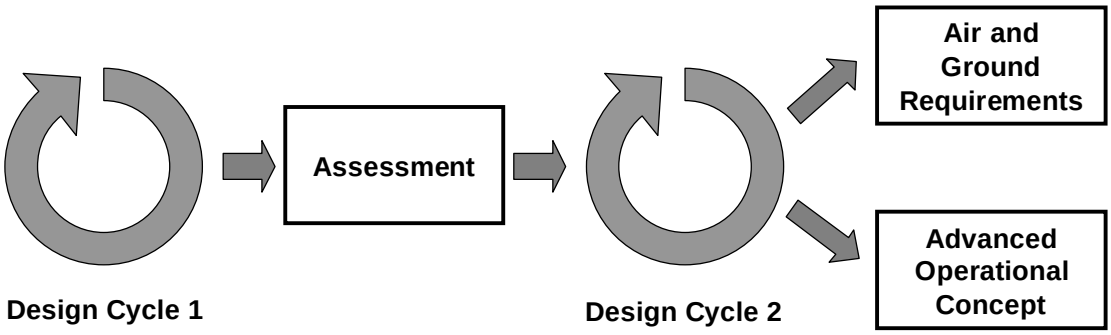


**Figure 1. iFly Work Structure.**

As depicted in Figure 2, iFly work is organised through nine technical Work Packages (WPs), each of which belongs to one of the four types of developments mentioned above:

Design cycle 1

The aim is to develop an Autonomous Aircraft Advanced ($A^3$) en-route operational concept which is initially based on the current "state-of-the-art" in aeronautics research. The $A^3$ ConOps is developed within WP1. An important starting and reference point for this $A^3$ ConOps development is formed by the human responsibility analysis in WP2.

Innovative methods

Develop innovative architecture free methods towards key issues that have to be addressed by an advanced operational concept:

- Develop a method to model and predict complexity of air traffic (WP3).
- Model and evaluate the problem of maintaining multi-agent Situation Awareness (SA) and avoiding cognitive dissonance (WP4).
- Develop conflict resolution algorithms for which it is formally possible to guarantee their performance (WP5).

Assessment cycle

Assess the state-of-the-art in Autonomous Aircraft Advanced ($A^3$) en-route operations concept design development with respect to human factors, safety and economy, and identify which limitations have to be mitigated in order to accommodate a three to six times increase in air traffic demand:

- Assess the $A^3$ operation on economy, with emphasis on the impact on organisational and institutional issues (WP6).
- Assess the $A^3$ operation on safety as a function of traffic density increase over current and mean density level (WP7).

Design cycle 2

The aim is to refine the $A^3$ ConOps of design cycle 1 and to develop a vision how $A^3$ equipped aircraft can be integrated within SESAR concept thinking (WP8). WP9 develops preliminary safety and performance requirements on the applicable functional elements of the $A^3$ ConOps, focused on identifying the required technology.

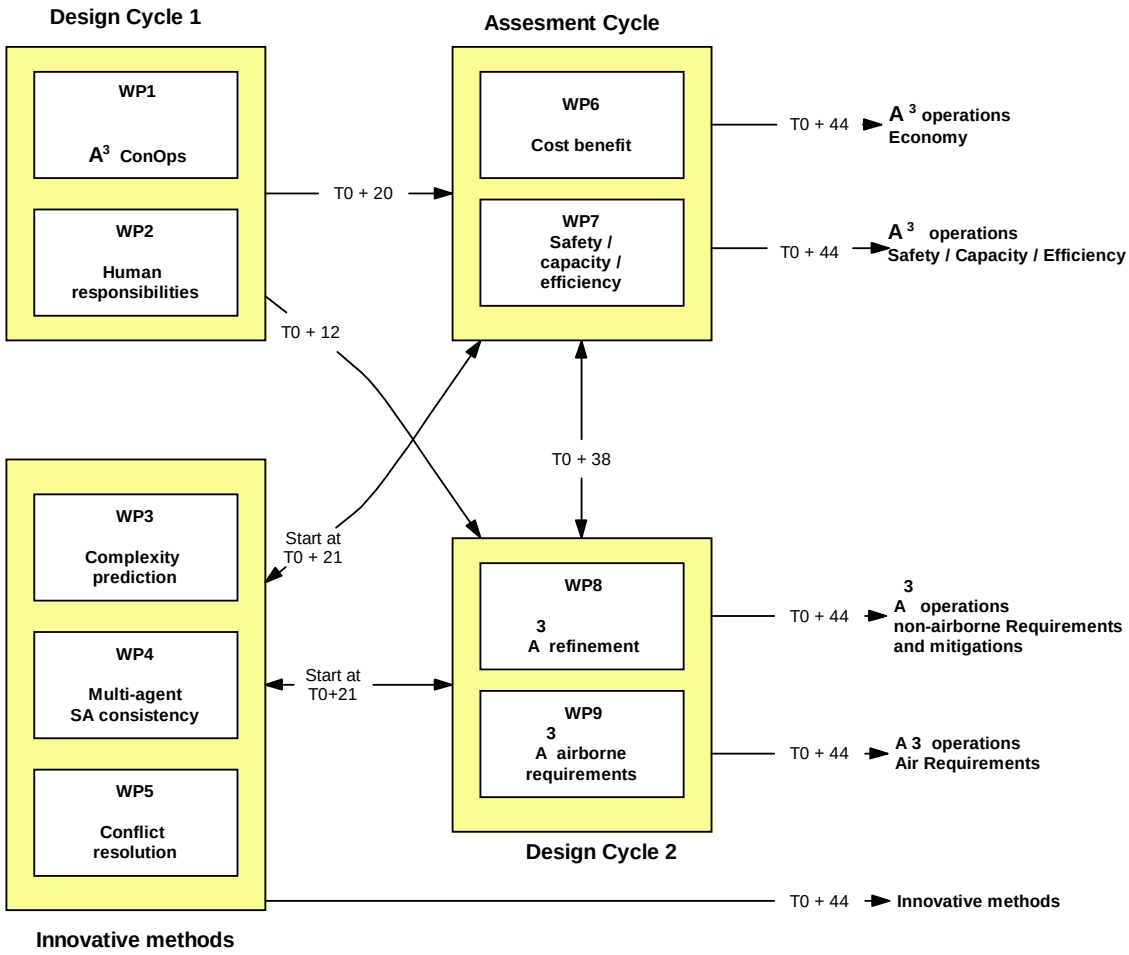


**Figure 2. Organisation of iFly research.**

## *1.2 Objective of iFly work package 7*

The aim of iFly WP7 is to assess the Autonomous Aircraft Advanced ($A^3$) operations developed by WP1 ($A^3$ Concept) and WP2 (Human responsibilities in autonomous aircraft operations), through hazard identification and Monte Carlo simulation on accident risk as a function of traffic demand, to assess what traffic demand can safely be accommodated by this advanced operational concept, and to assess the efficiency of the flights. The accident risk levels assessed should be in the form of an expected value, a 95% uncertainty area, and a decomposition of the risk level over the main risk contributing sources. In order to accomplish this assessment through Monte Carlo simulation, the complementary aim of this WP is to further develop the innovative HYBRIDGE speed up approaches in rare event Monte Carlo simulation. The work is organised in four sub-WPs:

- WP7.1: Monte Carlo simulation model of $A^3$ operation
- WP7.2: Monte Carlo speed up methods

- WP7.3 Perform Monte Carlo simulations
- WP7.4 Final report

This report addresses a specific task within WP7.2, as is explained below.

### *1.3 WP7.2: Monte Carlo speed up methods*

Within HYBRIDGE novel Monte Carlo simulation speed up techniques have successfully been developed and applied. As such, we start with a review of the Monte Carlo simulation based accident risk assessment situation, this is reported in [iFly D7.2a], entitled 'Review of risk assessment status for air traffic'.

Subsequently, potential candidates are identified that are expected to provide significant room for the development of complementary speed-up and bias and uncertainty assessment techniques. In order to spread the risk as much as is possible, within this task various options for improvement are identified and these are subsequently elaborated and tested within parallel tasks. In order to explore the various options, several studies will be conducted, i.e.:

- Develop an effective combination of Interacting Particle System based rare event simulation with Markov Chain Monte Carlo speed up technique. This is reported in iFly Deliverable [iFly D7.2b], entitled 'Trans-dimensional simulation for rare-events estimation on stochastic hybrid systems'.
- Develop a method to assess the sensitivity of multiple aircraft encounter geometries to collision risk, and develop importance sampling approaches which take advantage of these sensitivities. This will be reported in iFly deliverable D7.2c [iFly D7.2c], entitled 'Interim Report on Importance sampling of multi aircraft encounter geometries'.
- Develop novel ways how Interacting Particle System speed up techniques that apply to a pair of aircraft can effectively be extended to situations of multiple aircraft. This is reported in iFly deliverable [iFly D7.2d], entitled 'Periodic Boundary Condition in Large Scale Random Air Traffic Scenarios'.
- Develop an efficient extension of Interacting Particle System based rare event simulation for application to hybrid systems. This is reported in iFly deliverable [iFly D7.2e], entitled 'Rare event estimation for a large scale stochastic hybrid system with air traffic application - Interacting particle system (IPS) extension to large hybrid systems'.
- Combine Monte Carlo simulation based bias and uncertainty assessment with operation design parameter optimization. The key towards accomplishing this is to integrate sensitivity analysis with MC simulation based rare event estimation. This study is addressed in the current report, i.e. iFly deliverable D7.2f.

Finally, the results obtained from these studies will be as much as possible combined and integrated with the innovative speed up approaches developed within HYBRIDGE. This way we prepare an improved speed up approach for application to the $A^3$ ConOps Monte Carlo simulation model of WP7.1. This will be reported in iFly deliverable D7.2g, entitled 'Monte Carlo speed up studies'.

### 1.4 Objective and organisation of the study in this report

The objective of this report is to study Monte Carlo based assessment of the sensitivity of collision risk versus changes in parameter values. The motivation for studying assessment of these sensitivities stems from two complementary purposes. One purpose is that for a parameter value having uncertainty, one wants to know how this parameter uncertainty influences risk uncertainty. The other purpose is that in the design of a novel concept some parameters are under control of the design team. In such case parameter sensitivity knowledge shows the design team which requirements should be posed on these parameter values in order to reduce the risk in a predictable way.

This report is organised as follows. Section 2 starts with the rationale and approach of this study. The multi-dimensional regression problem and methods studied in this report are presented in Section 3. Next, in Section 4 the set-up of the Monte Carlo simulations are presented and parameter choices in the example are given in Section 5. The results of the Monte Carlo simulations are discussed in Sections 6, 7 and 8. Section 6 considers four different estimation types and two sampling types for a low number of regression coefficients. Section 7 considers the effect of the number of sampling types and standard deviation of the noise for a low number of regression coefficients. Section 8 considers the effect of a higher number of regression coefficients. Concluding remarks are given in Section 9.

# 2 Rationale and approach of this study

## 2.1 Dual role of sensitivity analysis

In a series of studies [Blom et al., 2007, 2009] the Interacting Particle System (IPS) approach has been developed for the speeding up Monte Carlo simulation based estimation of collision risk $G$ of a stochastic hybrid system model of an advanced ATM operation. Because such a model has an $n$-dimensional parameter $q$, the collision risk $G$ is a function of $q$, i.e. $G(q)$.

Potentially, each of the $n$ elements of $q$ may be one of the following two types:

- a parameter under control of the design,
- a variable having aleatory or epimistic uncertainty[1].

In both cases it is important to assess how $G$ changes as a result of changes in $q$. If $G(q)$ is linear in $q$, then this is expressed by the partial derivative $\frac{\partial G(q)}{\partial q}$, which is an $n$-dimensional vector. Hence, a change $\Delta_q$ in the value of parameter $q$ yields a change of size $\Delta_G$ in risk, i.e.

$$\Delta_G = \left[\frac{\partial G(q)}{\partial q}\right]^T \Delta_q$$

The $\Delta_q$ may either be due to a controlled change in the design parameter components, or due to errors in the assumed values for the other variables. If the uncertainty in $q$ has a covariance $\Sigma_q$, then $G(q)$ is estimated with a standard deviation $\sigma_G$, satisfying:

$$\sigma_G^2 = \left[\frac{\partial G(q)}{\partial q}\right]^T \Sigma_q \left[\frac{\partial G(q)}{\partial q}\right]$$

[1] Uncertainty can be formally classified as aleatory uncertainty and epistemic uncertainty [Swiler & Giunta, 2007]. *Aleatory uncertainty* (or stochastic uncertainty) characterizes the inherent randomness in the behavior of the system under study. *Epistemic uncertainty* (or state of knowledge uncertainty, or subjective uncertainty) characterizes the lack of knowledge about the appropriate value to use for a quantity that is assumed to have a fixed value in the context of a specific application. Thus stochastic uncertainty is a property of the system under study, while subjective uncertainty is a property of the analysts performing the study [Helton, 1993].

A large $\frac{\partial G(q)}{\partial q_i}$ is helpful when $q_i$ is a design parameter. However the opposite is true when $q_i$ is a variable having uncertainty. Because of the dual role that may be played by the derivative $\frac{\partial G(q)}{\partial q_i}$ this report studies how to estimate this derivative by conducting multiple IPS runs.

In handling this problem there are several issues that have to be taken into account:

- In general, $G(q)$ is not linear in *q*;
- The number *n* of scalar parameters is large, e.g. $n > 100$.
- One IPS run using parameter value *q* does not yield $G(q)$ but yields it with a random error $\varepsilon$ [Blom et al., 2007, 2009], i.e.

$$\tilde{G} = (1+\varepsilon)G(q)$$

- The standard deviation of random error $\varepsilon G(q)$ decreases only with the square root of the number of Monte Carlo simulation runs that are used for one IPS run [Blom et al., 2007].

## 2.2 Identification of suitable sensitivity analysis approach

The issue of sensitivity analysis for large computer simulation models has been a rich area of research for several decades. Nice overviews of the resulting developments are provided by [Morgan & Henrion, 1990], [Cacuci, 2003], [Kurowicka & Cooke, 2006], [Saltelli et al., 2008] and [DeRocquigny et al., 2008]. The general setting of the problems considered is that $G(q)$ is nonlinear in *q*, the number *n* of scalar parameters is significant, e.g. $n \gg 1$, and a computation of $G(q)$ for one value of $q$ is demanding.

Our application of the IPS approach towards the advanced ATM application [Blom et al., 2007, 2009] clearly is at the demanding side of the spectrum of sensitivity analysis problems. In particular since $n > 100$ and the random error $\varepsilon G(q)$ has a significant variance.

The classical approach towards sensitivity analysis is to write $G(q)$ as a Taylor series expansion, and then linearize around a specific (local) working point $q^*$. For this classical linearization approach, well working sensitivity analysis approaches have been developed [Cacuci, 2003]. The key shortcoming of a local approach is that it works well in a linear neighbourhood of $q^*$ only. However, if relevant $q$ values fall outside of this linear neighbourhood, then the approach falls short.

In order to capture the full spectrum of relevant $q$ values, sensitivity analysis has to be done in some global way. There are three main global sensitivity approaches [Saltelli et al, 2008]:

- One-At-a-Time
- Sobol Total Effects
- Meta-modelling

For each of these three main approaches, a short summary and its usability in combination with IPS are shortly discussed below.

One of the simplest types of global methods is known as One-At-a-Time (OAT). The OAT approach comes down to calculating deltas in *G* as a result of deltas in one component of *q*. [Morris, 1991] developed an efficient randomized experimental plan for performing OAT which requires computation of *G(q)* for $(n+1)K$ values of *q*. Here *K* is the number of grid points needed to cover the nonlinearity of *G(q)*. A disadvantage of an OAT approach is its non-robustness towards random errors if a computation of $G(q)$ for each value of $q$ is done through running Monte Carlo simulations. This makes OAT unsuitable to be used in combination with an IPS approach.

Sobol Total Effects approach towards sensitivity analysis represents $G(q)$ as a finite series of commonly named "Sobol terms" with increasing order of interaction between the components of *q* [Sobol, 1993]. Although the number of "Sobol terms" is finite, there are combinatorially many of them. However, for sensitivity analysis there is no need to assess each of these terms individually. What is needed only is the "total effect" for each of the components of *q*. Following this principle, [Saltelli, 2002] developed a systematic method to compute these Sobol "Total Effects" through computation of $G(q)$ for $K(n+2)$ values of $q$. Here $K$ is a base number of samples, which may vary from a few hundred to a few thousand. Hence, Sobol "Total Effects" based global variance estimation is computationally very demanding, and therefore not a suitable candidate to be combined with IPS.

In order to escape from the limitations of the above explained general methods, for specific domain applications often a kind of analytical meta-model $\hat{G}(q)$ is being developed for $G(q)$. Once $\hat{G}(q)$ has been estimated, sensitivity computations can be done for $\hat{G}(q)$ rather than for $G(q)$. The estimation of $\hat{G}(q)$ is done by finding the optimal fit from a family of analytically defined functions to input-output samples $\{q_k, \tilde{G}_k;\ k=1,..,K\}$. Through a regression analysis the estimation of $\hat{G}(q)$ is robust against random errors in the output samples. A meta-model approach needs the base number of *K* samples only once. Obviously, the effectiveness of a meta-model

approach largely depends on a proper choice of the family of analytically defined functions.

During earlier collision risk assessment for ATM in [Everdij & Blom, 2002], for sensitivity analysis the following exponential family of functions has shown to work well for fitting a meta-model $\hat{G}(q)$ of the form:

$$\ln \hat{G}(q) = \hat{f}_0 + \hat{F}^T \ln q \qquad (1)$$

with estimated intercept term $\hat{f}_0$ and estimated parameter $\hat{F}$. In order to apply (1) it is required that each of the *n* components of *q* assumes strictly positive values only [Everdij & Blom, 2005; Everdij et al., 2006]. In previous TOPAZ studies this meta-model has been used in combination with an OAT approach. In the current study, meta-modelling approach (1) has been identified as the logical candidate to be combined with multi-dimensional regression analysis.

### 2.3 Logarithmic meta-model of sensitivity estimation approach

Assume the *k*-th IPS run uses $q_k \sim p_{q_k}(.)$ as input sample and yields an IPS computed output value

$$\tilde{G}_k = (1+\varepsilon_k) G(q_k)$$

Now taking logarithm yields:

$$\ln \tilde{G}_k = \ln(1+\varepsilon_k) + \ln G(q_k) \qquad (2)$$

Through a multi dimensional regression analysis of *K* data pairs $\{\ln q_k, \ln \tilde{G}_k\},\ k = 1,..,K,$ we get the following $\hat{G}(q_k)$ estimate

$$\ln \hat{G}(q_k) = \hat{f}_0 + \hat{F}^T \ln q_k$$

with $\hat{f}_0$ and $\hat{F}^T$ such that Root Mean Square of $\left[\ln \tilde{G}_k - \ln \hat{G}(q_k)\right]$ is minimal.

The implication is that a change of a factor $\chi_q$ in $\ln q_k$ yields a factor $\chi_G$ change in $\ln \hat{G}(q_k)$ with

$$\chi_G = \hat{F}^T \chi_q$$

Similarly if the uncertainty in $\ln q_k$ has covariance $Cov\{\ln q_k\}$ then

$$Var\{\ln \hat{G}(q_k)\} = \hat{F}^T Cov\{\ln q_k\}\hat{F}$$

### 2.4 Multi-dimensional linear regression problem

For the derivation of a multi-dimensional regression analysis approach we consider the log-linear situation

$$\ln G(q_k) = f_0 + F^T \ln q_k$$

Substituting this in Equation (2) yields

$$\ln \tilde{G}_k = \ln(1+\varepsilon_k) + f_0 + F^T \ln q_k \qquad (3)$$

Next, we define:

$$y_k = \ln \tilde{G}_k$$
$$w_k = \ln(1+\varepsilon_k)$$
$$x_k = \ln q_k$$

Hence, $x_k$ is an *n*-dimensional vector.
With this, Equation (3) becomes:

$$y_k = f_0 + F^T x_k + w_k \qquad (4)$$

From Equation (4) we get

$$y_k = \begin{bmatrix} f_0 & F^T \end{bmatrix} \begin{bmatrix} 1 \\ x_k \end{bmatrix} + w_k = \tilde{F}^T \tilde{x}_k + w_k$$

with

$$\tilde{x}_k^T = \begin{bmatrix} 1 & x_k \end{bmatrix}$$
$$\tilde{F}^T = \begin{bmatrix} f_0 & F^T \end{bmatrix}$$

and noise process $\{w_k\}$, a sequence of independent and identically distributed (i.i.d.) random variables, with $E\{w_k\} = 0$ and $Var\{w_k\} = \sigma_w^2$.

# 3 Multi-dimensional linear regression methods

In this report the linear version of the sensitivity estimation problem is considered. Assume for every $k = 1, 2, \ldots, K$, $y_k$ is a function of an $n_p$ dimensional vector $x_k$, i.e. consisting of $n_p$ components, say $x_{k,1}, x_{k,2}, \ldots, x_{k,n_p}$ and noise process $\{w_k\}$

$$y_k = \begin{bmatrix} f_0 & F^T \end{bmatrix} \begin{bmatrix} 1 \\ x_k \end{bmatrix} + w_k = \tilde{F}^T \tilde{x}_k + w_k \tag{5}$$

with

$$\tilde{x}_k^T = \begin{bmatrix} 1 & x_k \end{bmatrix}$$
$$\tilde{F}^T = \begin{bmatrix} f_0 & F^T \end{bmatrix}$$

with $F^T = \begin{pmatrix} f_1 & f_2 & \cdots & f_{n_p} \end{pmatrix}$ and intercept term $f_0$, and where $\{w_k\}$ is a sequence of independent and identically distributed (i.i.d.) random variables with $\{x_k\}$ and $\{w_k\}$ independent and $E\{w_k\} = 0$.

The values of $f_0$ and $f_1, f_2, \ldots, f_{n_p}$ are unknown; these have to be estimated from $\{y_k, x_k; k = 1, 2, \cdots, K\}$

Written in full, Equation (5) reads as follows:

$$y_k = \begin{pmatrix} f_0 & f_1 & \cdots & f_{n_p} \end{pmatrix} \begin{pmatrix} 1 \\ x_{k,1} \\ \vdots \\ x_{k,n_p} \end{pmatrix} + w_k \quad \text{for} \quad k = 1, 2, \ldots, K. \tag{6}$$

The multi-dimensional linear regression problem is to estimate the $n_p$-dimensional parameter $F$ and intercept term $f_0$ from the data sequences $\{y_1, y_2, \cdots, y_K\}$ and $\{x_1, x_2, \cdots, x_K\}$.

The problem addressed in this section is how does a linear regression based estimation of $F$ depend on the variables $x_k$ and $w_k$.

For this multi-dimensional regression problem, three different types of estimation approaches are explained in the following subsections, i.e. Classical Least Squares (CLS) estimation in Subsections 3.1-3.3, Least Squares estimation with Moore

Penrose (LS-MP) in Subsection 3.4, and Partial Least Squares (PLS) estimation in Subsection 3.5.

### 3.1 Classical Least Squares (CLS) estimation without intercept term

Consider Equation (5) with zero intercept term $f_0 = 0$ , i.e. consider

$$y_k = F^T x_k + w_k \quad \text{for} \quad k = 1, 2, \ldots, K \,. \tag{7}$$

with $F^T = \begin{pmatrix} f_1 & f_2 & \cdots & f_{n_p} \end{pmatrix}$. The values of $f_1$ , $f_2$ ,…, $f_{n_p}$ are unknown, and have to be estimated from $\{y_k, x_k; k = 1, 2, \cdots, K\}$. Written in full, Equation (7) reads as follows:

$$y_k = \begin{pmatrix} f_1 & f_2 & \cdots & f_{n_p} \end{pmatrix} \begin{pmatrix} x_{k,1} \\ x_{k,2} \\ \vdots \\ x_{k,n_p} \end{pmatrix} + w_k \quad \text{for} \quad k = 1, 2, \ldots, K \,. \tag{8}$$

Assume $\hat{F}$ is the Classical Least Squares (CLS) estimator of the $n_p$-dimensional parameter $F$ of Equation (7). Estimator $\hat{F}$ is the value of $F$ which minimizes the sum of squares of the deviations[2]

$$\hat{F} = \min_F \sum_{k=1}^{S} \left\| y_k - F^T x_k \right\|^2 \tag{9}$$

with known samples $x_k$ and known realisations $y_k$. That is

$$\hat{F} = \min_F \left( Y - X F \right)^T \left( Y - X F \right)$$

where the following definitions are used for $Y$ of size $K \times 1$ and $X$ of size $K \times n_p$

$$\begin{aligned} Y &\triangleq \mathrm{Col}\left( y_1, y_2, \ldots, y_K \right) \\ X &\triangleq \mathrm{Col}\left( x_1^T, x_2^T, \ldots, x_K^T \right) \end{aligned} \tag{10}$$

and thus matrix $X$ (this matrix is called the design matrix, see [Campbell and Meyer, 1979]) is given by

[2] The notation || . || represents the Euclidean norm (or 2-norm).

$$X = \begin{pmatrix} x_{1,1} & \cdots & x_{1,n_p} \\ \vdots & & \vdots \\ x_{K,1} & \cdots & x_{K,n_p} \end{pmatrix} \tag{11}$$

Similarly with these definitions and with $W \triangleq \mathrm{Col}(w_1, w_2, \ldots, w_K)$, Equations (7) and (8) for $k = 1, 2, \ldots, K$ can be written as

$$Y = X\,F + W \tag{12}$$

or written in full:

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_K \end{pmatrix} = \begin{pmatrix} x_{1,1} & \cdots & x_{1,n_p} \\ \vdots & & \vdots \\ x_{K,1} & \cdots & x_{K,n_p} \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n_p} \end{pmatrix} + \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_K \end{pmatrix}$$

Multiplying left and right hand terms in (12) by $X^T$ and subsequently taking *X,Y*-conditional expectation yields a characterisation for $\hat{F} \triangleq E\{F \mid X, Y\}$. Because $E\{X^T W \mid X, Y\} = 0$, the least squares solution of estimator $\hat{F}$ for the case that there is no intercept term satisfies the following equation

$$\left(X^T X\right)\hat{F} = X^T Y \tag{13}$$

Solving Equation (13) yields the least squares estimator $\hat{F}$ for the case that there is no intercept term, which consists of $n_p$ scalar values, say $F \triangleq \mathrm{Col}(f_1, f_2, \ldots, f_{n_p})$, hence $F$ is of size $n_p \times 1$.

***If full rank condition is satisfied***

Because matrix $X$ has size $K \times n_p$, matrix $X^T X$ is a square matrix of size $n_p \times n_p$ and $\mathrm{rank}\left(X^T X\right) \le \min\left(K, n_p\right)$. The latter follows from $\mathrm{rank}\left(X^T X\right) = \mathrm{rank}\left(X X^T\right) = \mathrm{rank}\left(X\right) \le \min\left(K, n_p\right)$. So this means that the rank of the square matrix $X^T X$ is always smaller than or equal to the number of parameters $n_p$.

If the inverse of the $n_p \times n_p$ matrix $X^T X$ exists, then the Classical Least Squares (CLS) estimator $\hat{F}$ follows directly from

$$\hat{F} = \left( X^T X \right)^{-1} X^T Y ,$$

which can for example be determined with Gaussian elimination.

The inverse of the $n_p \times n_p$ matrix $X^T X$ exists if and only if matrix $X^T X$ has full rank, i.e. if $\text{rank}\left( X^T X \right) = n_p$. Since the size of matrix $X$ is $K \times n_p$, this case can only happen for $K \geq n_p$, not for $K < n_p$.

***Ill-conditioning***

The *condition number* of a square matrix *A* is defined as $\kappa(A) = \|A\| \cdot \|A^{-1}\|$ for a given matrix norm (e.g. *p*-norm, Frobenius norm) with the convention that $\kappa(A) = \infty$ for singular *A*, and the condition number is always greater than or equal to 1. The condition number depends on the underlying norm. It is a measure of stability or sensitivity of a matrix to numerical operations. Consider for example a linear system $Ax = b$, the condition number of *A* is a measure of the sensitivity of the solution to perturbations of *A* or *b*. If the condition number is close to 1, then small relative perturbations in *b* will lead to similarly small relative perturbations in the solution *x*, in which case *A* is said to be *well-conditioned*. If the condition number of *A* is large, then small relative perturbations in *b* will lead to large relative perturbations in the solution *x*, in which case *A* is said to be *ill-conditioned* [Golub and van Loan, 1996].

This means that if matrix $X^T X$ is *ill-conditioned*, then the matrix inversion of $X^T X$ can cause numerical problems.

***Rank deficient case***

If the inverse of the $n_p \times n_p$ matrix $X^T X$ does not exist, then the least squares problem for the no intercept case as considered in this subsection has an infinite number of solutions. Alternative ways to determine a least squares solution for rank deficient cases are considered in Subsections 3.4 and 3.5.

### 3.2 Estimation error of unknown parameter vector F

**Lemma 1.** If the data satisfies $y_k = F^T x_k + w_k$ with $\{x_k\}$ and $\{w_k\}$ independent and $E\{w_k\} = 0$ and standard deviation $\sigma_w > 0$ for $k = 1, 2, \ldots, K$, and matrix $X^T X$ has full rank with $X \triangleq \mathrm{Col}\left(x_1^T, x_2^T, \ldots, x_K^T\right)$, then $\hat{F}$ is an unbiased estimator, i.e. $E\left\{\hat{F}\right\} = F$. It also follows that $Var\left(\hat{F}\right) = \sigma_w^2 \left(X^T X\right)^{-1}$.

*Proof:* According to (12):

$$Y = X F + W$$

where $W \triangleq \mathrm{Col}\left(w_1, w_2, \ldots, w_K\right)$. Because $X^T X$ has full rank it is invertible, and (13) yields

$$\begin{aligned}\hat{F} &= \left(X^T X\right)^{-1} X^T Y \\ &= \left(X^T X\right)^{-1} X^T \left(X F + W\right) \\ &= F + \left(X^T X\right)^{-1} X^T W\end{aligned}$$

Hence

$$E\left\{\hat{F}\right\} = F + \left(X^T X\right)^{-1} X^T E\{W\} = F$$

and

$$\begin{aligned}Var\left(\hat{F}\right) &= E\left[\left(\hat{F} - F\right)\left(\hat{F} - F\right)^T\right] \\ &= E\left[\left(X^T X\right)^{-1} X^T W \left(\left(X^T X\right)^{-1} X^T W\right)^T\right] \\ &= E\left[\left(X^T X\right)^{-1} X^T W W^T X \left(X^T X\right)^{-1}\right] \\ &= \left(X^T X\right)^{-1} X^T E\left\{W W^T\right\} X \left(X^T X\right)^{-1}\end{aligned}$$

The standard deviation $\sigma_w > 0$ implies that $E\left\{W W^T\right\} = \sigma_w^2 I$. Substitution yields

$$\begin{aligned}Var\left(\hat{F}\right) &= \left(X^T X\right)^{-1} X^T \sigma_w^2 I X \left(X^T X\right)^{-1} \\ &= \sigma_w^2 \left(X^T X\right)^{-1}\end{aligned}$$

■

In practice $\sigma_w^2$ is often not known. Because CLS algorithm does not use $\sigma_w^2$, then $\sigma_w^2$ can be estimated from $Y, X$ as follows.

$$Y = X\hat{F} + \hat{W} \quad \text{and thus} \quad \hat{W} = Y - X\hat{F}$$

Hence

$$E\left\{\sigma_w^2 \mid Y, X\right\} = \frac{\hat{W}^T\hat{W}}{K-1} = \frac{1}{K-1}\left[Y - X\hat{F}\right]^T\left[Y - X\hat{F}\right].$$

### 3.3 Classical Least Squares (CLS) estimation with intercept term

Consider Equation (5) with intercept term $f_0$, i.e.

$$y_k = \begin{bmatrix} f_0 & F^T \end{bmatrix} \begin{bmatrix} 1 \\ x_k \end{bmatrix} + w_k = \tilde{F}^T \tilde{x}_k + w_k \quad \text{for} \quad k = 1, 2, \ldots, K \,. \tag{14}$$

with

$$\tilde{x}_k^T = \begin{bmatrix} 1 & x_k \end{bmatrix} \qquad \tilde{F}^T = \begin{bmatrix} f_0 & F^T \end{bmatrix}$$

that is

$$\tilde{x}_k^T = \begin{pmatrix} 1 & x_{k,1} & \cdots & x_{k,n_p} \end{pmatrix} \qquad \tilde{F}^T = \begin{pmatrix} f_0 & f_1 & \cdots & f_{n_p} \end{pmatrix}$$

The values of $f_0$ and $f_1$, $f_2$, …, $f_{n_p}$ are unknown; these have to be estimated from $\{y_k, x_k; k = 1, 2, \cdots, K\}$.

Written in full, Equation (14) reads as follows:

$$y_k = \begin{pmatrix} f_0 & f_1 & \cdots & f_{n_p} \end{pmatrix} \begin{pmatrix} 1 \\ x_{k,1} \\ \vdots \\ x_{k,n_p} \end{pmatrix} + w_k \quad \text{for} \quad k = 1, 2, \ldots, K \,. \tag{15}$$

Assume $\hat{\tilde{F}}$ is the Classical Least Squares (CLS) estimator of the $(n_p + 1)$-dimensional parameter $\tilde{F}$ of Equation (14). Estimator $\hat{\tilde{F}}$ is the value which minimizes the sum of squares of the deviations

$$\hat{\tilde{F}} = \min_{\tilde{F}} \sum_{k=1}^{K} \left\| y_k - \hat{\tilde{x}}_k^T \tilde{F} \right\|^2$$

with known samples $\tilde{x}_k$ and known realisations $y_k$. That is

$$\hat{\tilde{F}} = \min_{\tilde{F}} \left( Y - \tilde{X}\,\tilde{F} \right)^T \left( Y - \tilde{X}\,\tilde{F} \right)$$

where the following definitions are used for $Y$ of size $K \times 1$ and $\tilde{X}$ of size $K \times (n_p + 1)$

$$\begin{aligned} Y &\triangleq \mathrm{Col}\left(y_1, y_2, \ldots, y_K\right) \\ \tilde{X} &\triangleq \mathrm{Col}\left(\tilde{x}_1^T, \tilde{x}_2^T, \ldots, \tilde{x}_K^T\right) \end{aligned} \tag{16}$$

and thus matrix $\tilde{X}$ (also called design matrix) is

$$\tilde{X} = \begin{pmatrix} 1 & x_{1,1} & \cdots & x_{1,n_p} \\ \vdots & \vdots & & \vdots \\ 1 & x_{K,1} & \cdots & x_{K,n_p} \end{pmatrix} \tag{17}$$

Similarly with these definitions and with $W \triangleq \mathrm{Col}\left(w_1, w_2, \ldots, w_K\right)$, Equations (14) and (15) for $k = 1, 2, \ldots, K$ can be written as

$$Y = \tilde{X}\,\tilde{F} + W \tag{18}$$

or written in full:

$$\begin{pmatrix} y_1 \\ \vdots \\ y_K \end{pmatrix} = \begin{pmatrix} 1 & x_{1,1} & \cdots & x_{1,n_p} \\ \vdots & \vdots & & \vdots \\ 1 & x_{K,1} & \cdots & x_{K,n_p} \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_{n_p} \end{pmatrix} + \begin{pmatrix} w_1 \\ \vdots \\ w_K \end{pmatrix}$$

The least squares solution of estimator $\hat{\tilde{F}}$ for the case that there is an intercept term satisfies the following equation

$$\left(\tilde{X}^T \tilde{X}\right)\hat{\tilde{F}} = \tilde{X}^T Y \tag{19}$$

Solving Equation (19) yields the least squares estimator $\hat{\tilde{F}}$ for the case that there is an intercept term, which consists of $n_p + 1$ scalar values, say $\tilde{F} \triangleq \mathrm{Col}\left(f_0, f_1, f_2, \ldots, f_{n_p}\right)$ of size $\left(n_p + 1\right) \times 1$.

If the inverse of the $\left(n_p + 1\right) \times \left(n_p + 1\right)$ matrix $\tilde{X}^T \tilde{X}$ exists, Equation (19) can in principle be solved by determined with Gaussian elimination. However using matrix $\tilde{X}$, more information would be used than the data provided, because of the first column $\mathbf{j}_{K\times 1}$, a $K \times 1$-dimensional vector of ones. Therefore consider application of Theorem 2.4.1 on page 36 in [Campbell and Meyer, 1979], from which the following Corollary can be derived.

**Corollary 1.** Let matrix $\tilde{X} = [\mathbf{j}_{K\times 1} \quad X]$ be a matrix of size $K \times (n_p + 1)$, where $\mathbf{j}_{K\times 1} = \begin{pmatrix} 1 \\ \vdots \\ 1 \end{pmatrix}_{K\times 1}$ is a $K \times 1$ vector of ones.

The vector $\hat{\tilde{F}}^T = \begin{bmatrix} \hat{f}_0 & \hat{F}^T \end{bmatrix} = \begin{pmatrix} \hat{f}_0 & \hat{f}_1 & \cdots & \hat{f}_{n_p} \end{pmatrix}$ with $\hat{F}^T = \begin{pmatrix} \hat{f}_1 & \hat{f}_2 & \cdots & \hat{f}_{n_p} \end{pmatrix}$, is a least squares solution of $\tilde{X}\,\tilde{F} = Y$ if and only if

$$\hat{f}_0 = \frac{1}{K}\,\mathbf{j}_{1\times K}\left(Y - X\,\hat{F}\right) \tag{20}$$

and $\hat{F}$ is a least squares solution of

$$X_0\,F = Y_0 \tag{21}$$

where

$$\begin{aligned} X_0 &= \left(I_{K\times K} - \frac{1}{K}J_{K\times K}\right)X \\ Y_0 &= \left(I_{K\times K} - \frac{1}{K}J_{K\times K}\right)Y \end{aligned} \tag{22}$$

and $J_{K\times K} = \mathbf{j}_{K\times 1}\,\mathbf{j}_{1\times K}$ is a matrix of ones, and $I_{K\times K}$ is an identity matrix of size $K \times K$.

■

Equation (20) written in full reads as follows

$$\begin{aligned} \hat{f}_0 &= \frac{1}{K}\,\mathbf{j}_{1\times K}\left(Y - X\,\hat{F}\right) \\ &= \frac{1}{K}\begin{pmatrix} 1 & \cdots & 1 \end{pmatrix}\left[\begin{pmatrix} y_1 \\ \vdots \\ y_K \end{pmatrix} - \begin{pmatrix} x_{1,1} & \cdots & x_{1,n_p} \\ \vdots & & \vdots \\ x_{K,1} & \cdots & x_{K,n_p} \end{pmatrix}\begin{pmatrix} \hat{f}_1 \\ \vdots \\ \hat{f}_{n_p} \end{pmatrix}\right] \\ &= \bar{y} - \begin{pmatrix} \bar{x}_1 & \cdots & \bar{x}_{n_p} \end{pmatrix}\begin{pmatrix} \hat{f}_1 \\ \vdots \\ \hat{f}_{n_p} \end{pmatrix} \end{aligned} \tag{23}$$

where the mean $\bar{y}$ and the mean $\bar{x}_j$ of the $j$-th column are defined as

$$\overline{y} = \frac{1}{K}\sum_{k=1}^{K} y_k \,, \qquad \overline{x}_j = \frac{1}{K}\sum_{k=1}^{K} x_{k,j} \,.$$

Matrix $X_0$ and vector $Y_0$ in Equation (22) written in full read as follows:

$$X_0 = \begin{pmatrix} x_{1,1} - \overline{x}_1 & \cdots & x_{1,n_p} - \overline{x}_{n_p} \\ \vdots & & \vdots \\ x_{K,1} - \overline{x}_1 & \cdots & x_{K,n_p} - \overline{x}_{n_p} \end{pmatrix}, \qquad Y_0 = \begin{pmatrix} y_1 - \overline{y} \\ \vdots \\ y_K - \overline{y} \end{pmatrix} \tag{24}$$

From Corollary 1 it follows that the least squares solution $\hat{\bar{F}}^T = \begin{bmatrix} \hat{f}_0 & \hat{F}^T \end{bmatrix} = \begin{pmatrix} \hat{f}_0 & \hat{f}_1 & \cdots & \hat{f}_{n_p} \end{pmatrix}$ of Equation (19) is equivalent to solving:

(i) estimator of the intercept term $\hat{f}_0$ which satisfies Equation (20), that is

$$\hat{f}_0 = \frac{1}{K}\,\mathbf{j}_{1\times K}\left(Y - X\,\hat{F}\right), \text{ and}$$

(ii) least squares solution of estimator $\hat{F}$, where $\hat{F}^T = \begin{pmatrix} \hat{f}_1 & \hat{f}_2 & \cdots & \hat{f}_{n_p} \end{pmatrix}$ satisfies the following equation

$$\left(X_0{}^T X_0\right)\hat{F} = X_0{}^T Y_0 \tag{25}$$

For the case with an intercept term $f_0$ as considered in Equation (14), the solvability of Equation (25) depends on whether or not the $n_p \times n_p$ matrix $X_0{}^T X_0$ is invertible, but even if matrix $X_0{}^T X_0$ is invertible, it can be *ill-conditioned*, which implies that matrix inversion of $X_0{}^T X_0$ can cause numerical problems.

***If full rank condition is satisfied***

Because of the construction of matrix $X_0$ in Equation (24) it follows that the maximum number of independent rows equals $K - 1$, since the rows of matrix $X_0$ add up to zero. Therefore it follows that for square matrix $X_0{}^T X_0$ of size $n_p \times n_p$ that $\operatorname{rank}\left(X_0^T X_0\right) = \operatorname{rank}\left(X_0 X_0^T\right) = \operatorname{rank}\left(X_0\right) \le \min\left(K-1, n_p\right)$. So the rank of the square matrix $X_0{}^T X_0$ is always smaller than or equal to the number of parameters $n_p$.

If the inverse of the $n_p \times n_p$ matrix $X_0{}^T X_0$ exists, then the Classical Least Squares (CLS) estimator $\hat{F}$ follows directly from

$$\hat{F} = \left( X_0{}^T X_0 \right)^{-1} X_0{}^T Y_0 ,$$

which can for example be determined with Gaussian elimination.

The inverse of the $n_p \times n_p$ matrix $X_0{}^T X_0$ exists if and only if matrix $X_0{}^T X_0$ has full rank, i.e. if $\text{rank}\left( X_0{}^T X_0 \right) = n_p$. This case can only happen for $K \geq n_p + 1$, not for $K \leq n_p$.

***Rank deficient case***

If the inverse of the $n_p \times n_p$ matrix $X_0{}^T X_0$ does not exist, then the least squares problem for the intercept case considered in this subsection has an infinite number of solutions. An alternative way to determine a least squares solution to Equation (19) is to apply the Moore-Penrose or pseudo-inverse to $X^T X$ or $X_0{}^T X_0$, this is considered in Subsection 3.4. Another alternative way is to apply Partial Least Squares (PLS) estimation, which is described in Subsection 3.5.

### 3.4 Least Squares estimation with Moore Penrose (LS-MP)

It was shown in Subsection 3.3 that for data satisfying Equation (14) including an intercept term, the least squares solution of estimator $\hat{F}$ satisfies Equation (25), that is

$$\left(X_0{}^T X_0\right)\hat{F} = X_0{}^T Y_0 .$$

If the inverse of the $n_p \times n_p$ matrix $X_0{}^T X_0$ does not exist, then the least squares problem with the intercept term has an infinite number of solutions.

An alternative way to determine a least squares solution to Equation (25) is to apply the Moore-Penrose or pseudo-inverse to square matrix $X_0{}^T X_0$. Then the least squares estimator $\hat{F}$ follows from

$$\hat{F} = \left(X_0{}^T X_0\right)^+ X_0{}^T Y_0 .$$

where $\left(X_0{}^T X_0\right)^+$ represents the Moore-Penrose or pseudoinverse of matrix $X_0{}^T X_0$.

A Moore-Penrose or pseudoinverse $A^+$ of an $m \times n$ matrix $A$ is a generalization of the inverse matrix, it is defined as the unique $n \times m$ matrix satisfying all of the following four Moore-Penrose conditions:

$$AA^+A = A,\ A^+AA^+ = A^+,\ \left(AA^+\right)^* = AA^+,\ \left(A^+A\right)^* = A^+A,$$

where $H^*$ is the Hermitian transpose (also called conjugate transpose) of a matrix $H$. For matrices whose elements are real numbers instead of complex numbers, $H^* = H^{\mathrm{T}}$. A Moore-Penrose or pseudoinverse of a matrix can be determined by using the Singular Value Decomposition[3]. If rank($A$) = $n$, then $A^+ = \left(A^T A\right)^{-1} A^T$, while if $m = n$ = rank($A$), then $A^+ = A^{-1}$. This latter means that if a square matrix is invertible, the Moore-Penrose inverse and inverse coincide by definition (see for example [Golub and van Loan, 1996] or [Strang, 1980]).

If square matrix $X_0{}^T X_0$ is invertible, then the Moore-Penrose inverse and inverse coincide by definition, which means that the solution of estimator $\hat{F}$ obtained by CLS coincides with the solution of estimator $\hat{F}$ obtained by least squares estimation with Moore-Penrose. The Moore-Penrose inverse can be applied to matrix $X_0^T X_0$ (and similarly to matrix $X^T X$ in case there is no intercept term), though for large matrices this is time-consuming [Courrieu, 2005].

[3] A Singular Value Decomposition (SVD) decomposes a matrix into the product of three matrices, such that $A = USV^T$. If $A$ is a real matrix, the $U$ and $V$ are orthogonal matrices. If $A$ is a complex matrix, then $U$ and $V$ are unitary matrices. Matrix $S$ is a diagonal matrix whose diagonal values are in descending order. The diagonal values in $S$ are the nonnegative square roots of the eigenvalues of $A^TA$ and are defined as the singular values of $A$. The columns of $U$ and $V$, which are called left and right singular vectors, are orthonormal eigenvectors of $AA^T$ and $A^TA$, respectively, or, when $A$ is complex, unitary eigenvectors of $AA^*$ and $A^*A$.

### 3.5 Partial Least Squares (PLS) estimation

Another approach to deal with rank deficiency of matrices $X^T X$ and $X_0^T X_0$ in the multi-dimensional linear regression problems is considered in this subsection. This is the Partial Least Squares (PLS) regression method, and is described in this subsection for the case with an intercept term.

Consider the multi-linear regression problem with intercept term described in Section 3.3 with Equation (18)

$$Y = \tilde{X}\,\tilde{F} + W$$

or written in full:

$$\begin{pmatrix} y_1 \\ \vdots \\ y_K \end{pmatrix} = \begin{pmatrix} 1 & x_{1,1} & \cdots & x_{1,n_p} \\ \vdots & \vdots & & \vdots \\ 1 & x_{K,1} & \cdots & x_{K,n_p} \end{pmatrix} \begin{pmatrix} f_0 \\ f_1 \\ \vdots \\ f_{n_p} \end{pmatrix} + \begin{pmatrix} w_1 \\ \vdots \\ w_K \end{pmatrix}$$

This latter equation can also be written as

$$Y = F_0 + X\,F + W$$

where $F_0 \triangleq \mathrm{Col}(f_0, f_0, \ldots, f_0)$ is an $K \times 1$ vector , and thus

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_K \end{pmatrix} = \begin{pmatrix} f_0 \\ f_0 \\ \vdots \\ f_0 \end{pmatrix} + \begin{pmatrix} x_{1,1} & \cdots & x_{1,n_p} \\ \vdots & & \vdots \\ x_{K,1} & \cdots & x_{K,n_p} \end{pmatrix} \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_{n_p} \end{pmatrix} + \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_K \end{pmatrix}$$

Here, we consider the rank deficient case, and show how to apply the Partial Least Squares (PLS) regression method. More details of PLS from literature are described in Appendix A.

There are different PLS algorithms as addressed for example in [De Jong, 1993] and [Rosipal & Krämer, 2006]. The two most commonly used algorithms for PLS are NIPALS and SIMPLS as explained shortly below. These two algorithms have been considered for the Monte Carlo simulations in this report.

***NIPALS based PLS (PLS-N)***

The PLS method in its classical form is based on the Nonlinear Iterative Partial Least Squares (NIPALS); it is an iterative process which involves the construction of "deflated data matrices". The approach is described in more detail in Appendices A and B. Literature shows that different choices can be made in the PLS algorithm based on NIPALS, for example the original *X* and *Y* are assumed to be mean-centered and some algorithms, though not all, also assume some kind of scaling (e.g. by subtracting column means and dividing by standard deviation of each column). There is also some freedom in normalisation of column vectors that are calculated in the algorithm as is shown in Appendix B, since normalisation be done at different points in the algorithm. These kinds of differences between algorithms make it difficult to directly compare the scores and loadings of different PLS implementations [Mevik & Wehrens, 2007]. Appendix B describes a PLS algorithm based on NIPALS.

***SIMPLS based PLS (PLS-S)***

In [De Jong, 1993], an algorithm is proposed which calculates the PLS factors directly as linear combinations of the original (centered) matrices and is referred to as SIMPLS which stand for 'Straightforward IMPlementation of a statistically inspired modification of the PLS method'. The SIMPLS approach avoids the construction of deflated matrices of the original *X* and *Y* matrices as is applied in the NIPALS based PLS algorithms. The SIMPLS approach directly finds weight vectors which are applied to the original matrix *X*, and without explicit computation of matrix inverses. As explained in [De Jong, 1993], this implies that SIMPLS is also faster than the PLS algorithms based on classical NIPALS (see also [Alin, 2009]).

**PLS steps based on SIMPLS**

PLS consists of the four steps as described hereafter, where the calculation of the PLS factors - in step 2 - is based on SIMPLS:

***Step 1: PLS centering***

Given matrix *X* of size $K \times n_p$ and column vector *Y* of size $K \times 1$ as in (21)-(22), i.e.

$$X = \begin{pmatrix} x_{1,1} & \cdots & x_{1,n_p} \\ \vdots & & \vdots \\ x_{K,1} & \cdots & x_{K,n_p} \end{pmatrix}_{K \times n_p} \quad \text{and} \quad Y = \begin{pmatrix} y_1 \\ \vdots \\ y_K \end{pmatrix}_{K \times 1} \tag{26}$$

The PLS algorithm first starts with centering of *X* and *Y*, i.e. by subtracting column means to get centered variables $X_c$ and $Y_c$. In this way matrices

$$X_c = X - M_X$$
$$Y_c = Y - M_Y \qquad (27)$$

where $M_X$ is an $K \times n_p$ matrix of column means and $M_Y$ is an $K \times 1$ vector of column means, i.e.

$$M_X = \begin{pmatrix} \overline{x}_1 & \cdots & \overline{x}_{n_p} \\ \vdots & & \vdots \\ \overline{x}_1 & \cdots & \overline{x}_{n_p} \end{pmatrix}_{K \times n_p}, \qquad M_Y = \begin{pmatrix} \overline{y} \\ \vdots \\ \overline{y} \end{pmatrix}_{K \times 1} \qquad (28)$$

where the mean $\overline{x}_j$ of the elements in the *j*-th column of matrix *X* and the mean $\overline{y}$ of vector *Y* satisfy

$$\overline{x}_j = \frac{1}{K}\sum_{k=1}^{K} x_{k,j}\,, \qquad \overline{y} = \frac{1}{K}\sum_{k=1}^{K} y_k\,.$$

With this centering, the (column) mean of $X_c$ is a matrix of zeros and similarly the mean of $Y_c$ is a vector of zeros. Some PLS algorithms also apply scaling of matrix *X* and vector *Y* in this first step as is described in Appendix A.

Remark:

This first PLS step has a relation with Corollary 1 in Subsection 3.3:

- the centered matrix $X_c$ and the centered vector $Y_c$ in Equation (27) are the same as matrix $X_0$ and vector $Y_0$ in Equations (22) and (24), and
- the matrix of column means $M_X$ and vector of column means $M_Y$ in Equation (28) are the same as $\frac{1}{K} J_{K\times K} X$ and $\frac{1}{K} J_{K\times K} Y$ in Equation (22).

Using these two relations, we get:

$$X_0 = X_c = X - M_X$$
$$Y_0 = Y_c = Y - M_Y \qquad (29a)$$

with

$$M_X = \frac{1}{K} J_{K\times K} X$$
$$M_Y = \frac{1}{K} J_{K\times K} Y \qquad (29b)$$

***Step 2: PLS decomposition***

The PLS technique works by successively extracting factors from both $X_0$ and $Y_0$, such that the covariance between the extracted factors is maximized. As described in [Geladi & Kowalski, 1986] and [Rosipal & Krämer, 2006], PLS decomposes the $K \times n_p$ matrix[4] of zero-mean variables $X_0$ and the $K \times 1$ vector of zero-mean variables $Y_0$ into two outer relations (within each block $X_0$ and $Y_0$ individually) and a linear inner relation is assumed between the column vectors of the score matrix $T$ and corresponding column vector in score matrix $U$. The PLS technique tries to find a linear decomposition of $X_0$ and $Y_0$ , such that

$$\begin{aligned} &\text{First block of variables:} && X_0 = T P^T + E_X \\ &\text{Second block of variables:} && Y_0 = U S^T + E_Y \\ &\text{Inner relation:} && U = T D + H \end{aligned} \qquad (30)$$

where $T$ and $U$ are $K \times a$ matrices of the $a$ extracted score vectors (components, latent[5] vectors), the $n_p \times a$ matrix $P$ and the $1 \times a$ matrix $S$ represent matrices of loadings[6] and the $K \times n_p$ matrix $E_X$ and the $K \times 1$ matrix $E_Y$ are the matrices of residuals. Matrix $D$ is an $a \times a$ diagonal matrix and $H$ denotes the $K \times a$ matrix of residuals.

The decomposition of $X_0$ and $Y_0$ is finalized so as to maximize covariance between extracted score matrices $T$ and $U$. There are multiple algorithms available to solve the PLS problem ([De Jong, 1993], [Rosipal & Krämer, 2006]), all algorithms follow an iterative process (i.e. column by column) to find score matrices $T$ and $U$ and loading matrices $S$ and $P$ and diagonal matrix $D$. As already explained above, the two most commonly used algorithms for PLS are NIPALS and SIMPLS. The SIMPLS based PLS approach of [De Jong, 1993] is described below. A NIPALS based PLS approach is described in more detail in Appendices A and B.

***SIMPLS***

In the SIMPLS algorithm of [De Jong, 1993], successive orthogonal vectors $t_1, t_2, \ldots, t_a$ are extracted from a given matrix $X$, such as to maximize their covariance with corresponding vector $u_1, u_2, \ldots, u_a$ of a given matrix (or vector) $Y$. These vectors

[4] Dimensions of $X$ and $Y$ in Appendices A and B are denoted (generally) as $n \times m$ and $n \times p$ instead of $K \times n_p$ and $K \times 1$ respectively.

[5] **Latent variables** replace the original variables by a smaller number of 'underlying' variables.

[6] **Loading vectors** are the estimated weights which are to be applied to the variables when fitting the bilinear relationship between the Y and X variables. (Ref: http://www.bioss.ac.uk/smart/unix/mplsgxe/slides/glossary.htm)

$t_h$ and $u_h$ depend on weight vectors $r_h$ and $s_h$ for *h=1,2,..., a* , which can be applied directly to the centered matrices $X_0$ and $Y_0$:

$$t_h = X_0 r_h, \qquad h = 1, 2, \ldots, a$$
$$u_h = Y_0 s_h, \qquad h = 1, 2, \ldots, a$$

such that

- the covariance $Cov(u_h, t_h)$ of score vectors $t_h$ and $u_h$ are maximised,
- weight vectors $r_h$ and $s_h$ are normalised, i.e. $r_h^T r_h = 1$ and $s_h^T s_h = 1$, and
- vectors $t_h$ are orthogonal, i.e. $t_j^T t_h = 0$ for $h > j$ and $h, j = 1, 2, \ldots, a$.

This means that the aim is to determine weight vectors $r_h$ and $s_h$ for *h = 1,2,..., a* such that

$$\max_{u_h, t_h} \left[ Cov(u_h, t_h) \right] = \max_{\|s_h\|=1, \|r_h\|=1} \left[ Cov(Y_0 s_h, X_0 r_h) \right] = \max_{\|s_h\|=1, \|r_h\|=1} \left[ s_h^T A_0^T r_h \right] \qquad (31)$$

where $A_0 = X_0^T Y_0$ **,** under the constraint that

$$t_j^T t_h = 0 \quad \text{where } t_h = X_0 r_h \text{ for } h > j \qquad (32)$$

Without this last constraint there is only one straightforward solution: vectors $r_1$ and $s_1$ are the first left and right singular vectors of cross product matrix $A_0 = X_0^T Y_0$. Orthogonality constraint (32) is imposed to generate more than one solution and to generate a set of orthogonal factors of *X*.

The first weight vectors $r_1$ and $s_1$ are the first left and right singular vectors of $A_0 = X_0^T Y_0$**,** can be obtained from the Singular Value decomposition (SVD) of cross-product $A_0 = X_0^T Y_0$. This implies that $r_1$ is the dominant[7] eigenvector of $A_0 A_0^T$ and $s_1$ is the dominant eigenvector of $A_0^T A_0$, which equals the maximum attainable covariance.

As is explained in more detail in [De Jong, 1993], with the constraint in Equation (32), the next weight vectors $r_h$ and $s_h$ for $h = 2, 3, \ldots, a$ are obtained as the dominant eigenvectors of $A_h A_h^T$ and $A_h^T A_h$ respectively, which can be obtained from the SVD of deflated product matrices $A_h$

[7] The *dominant eigenvector* of a matrix is an eigenvector corresponding to the eigenvalue of largest magnitude (for real numbers, largest absolute value) of that matrix.

$$A_h = A_{h-1} - v_h \left( v_h^T \, A_{h-1} \right) \quad \text{for} \quad h = 2,3,\ldots,a$$
$$A_1 = A_0$$

Here $[v_1, v_2, \ldots, v_a]$ represents an orthonormal basis of $[p_1, p_2, \ldots, p_a]$ for *X*-loading vectors $p_h$ expressing the relation between the original *X* variables and the $k^{\text{th}}$ PLS factor with

$$p_h = X_0^T \, t_h / \left( t_h^T \, t_h \right)$$

and may be obtained from a Gram-Schmidt orthonormalization (see Equation (33) in [De Jong, 1993]) of orthonormal basis $[v_1, v_2, \ldots, v_a]$

$$v_h \propto p_h - v_{h-1} \left( v_{h-1}^T \, p_h \right) \quad \text{for} \quad h = 2,3,\ldots,a$$
$$v_1 \propto p_1$$

The symbol $\propto$ not only denotes proportionality, but also a subsequent normalization of the resultant vector.

The main difference with the standard PLS (such as NIPALS) is that the deflation process applies to the cross-product $A_0 = X_0^T \, Y_0$ and not to the larger data matrices $X_0$ and $Y_0$ . Note that instead of centering both *X* and *Y*, one might center only *Y*, since $A_0 = X_0^T \, Y_0 = X^T \, Y_0$.

After extraction of the *a* components, matrices *R*, *T*, *P*, *S*, *U* and *V* are created consisting of the columns created by the vectors extracted during the individual iterations, i.e.

$$\begin{aligned} R &= [r_1, r_2, \ldots, r_a]_{n_p \times a} \\ T &= [t_1, t_2, \ldots, t_a]_{K \times a} \\ P &= [p_1, p_2, \ldots, p_a]_{n_p \times a} \\ S &= [s_1, s_2, \ldots, s_a]_{1 \times a} \\ U &= [u_1, u_2, \ldots, u_a]_{K \times a} \\ V &= [v_1, v_2, \ldots, v_a]_{n_p \times a} \end{aligned} \tag{33}$$

***Step 3: PLS estimation in terms of centered variables***

The estimation of *Y* in terms of the centered variables $X_0$ and $Y_0$ is [De Jong, 1993]:

$$Y_0 = X_0 \hat{F}_{PLS} \tag{34}$$

where estimator $\hat{F}_{PLS}$ is the $n_p \times 1$ regression coefficient matrix[8]

$$\hat{F}_{PLS} = R S^T \tag{35}$$

and where $R$ is a weight $n_p \times a$ matrix (also referred to as alternative weight matrix, as opposed to a weight matrix W in the NIPALS algorithm) and $S$ is a $1 \times a$ vector.

***Step 4: PLS estimation in terms of original variables***

In terms of the original variables $X$ and $Y$ it follows from Equations (29) and (34) that

$$Y = \hat{F}_{PLS_0} + X \hat{F}_{PLS} \tag{36}$$

where $\hat{F}_{PLS}$ is the $n_p \times 1$ regression matrix computed from Equation (35)
and $\hat{F}_{PLS_0}$ is an $K \times 1$ intercept term (i.e. the regression coefficient for the intercept) which follows from

$$\hat{F}_{PLS_0} = \mathrm{M}_Y - \mathrm{M}_X \hat{F}_{PLS} \tag{37}$$

[8] In Appendices A and B for the regression matrix, another notation is used, i.e. $\hat{B}_{PLS}$, and similarly for regression matrix and intercept terms used in Equation (36).

# 4 Set-up of Monte Carlo simulation

This section describes the approach taken in conducting the Monte Carlo simulations for the linear version of the sensitivity estimation problem as was explained in Section 3 and for small number of scalar parameters $n_p$.

## *4.1 Monte Carlo simulation approach*

In preparation of a Monte Carlo simulation of model (4), that is $y_k = f_0 + F^T x_k + w_k$, one has to adopt values for the estimator *F* and intercept parameter $f_0$ and shapes for the probability density functions for random variables $w_k$ and $x_k$, and to generate output values for $y_k$ according to Equation (4).
Given *K* samples for $x_k$ and *K* generated outputs for $y_k$, in this section the (partial) least squares estimator $\hat{F}$ will be determined using the various algorithms of Section 3. Finally, the resulting estimators $\hat{F}$ will be compared to the original adopted values for *F*. In this section we assume to run $N$ Monte Carlo simulation runs for each of the methods in Section 3.

The simulation approach can be summarized as follows:

**Step 1: Setup**
First start with the following choices:
1.a. Choose the number of components $n_p$, and choose values for the $n_p$-dimensional vector $F^T = \begin{pmatrix} f_1 & f_2 & \cdots & f_{n_p} \end{pmatrix}$ and the intercept parameter $f_0$.
1.b. Choose a probability density function for random variable $w_k$.
1.c. Choose a probability density function for random variable $x_k$.
1.d. Choose a value for the number of samples *K*.

Assumptions for the probability density functions of noise $w_k$ and random variable $x_k$ are discussed below in Subsection 4.2.

**Step 2: Type of least squares estimation and sampling method**
2.a. Choose the type of least squares estimation. Options are CLS, LS-MP, PLS-N, PLS-S as discussed in Subsections 3.1-3.5.
2.b. Choose the sampling method to draw samples for $x_k$. This is further discussed in Subsection 4.3.

**Step 3: Evaluate least squares estimator using Monte Carlo simulation**

Monte Carlo simulation using the method chosen in Step 2 works as follows:

First, determine the number $N$ of Monte Carlo simulations runs to evaluate the method chosen in Step 2.

Next, for each of these $N$ Monte Carlo runs the following steps apply.

3.a. For $k = 1, 2, \ldots, K$ draw samples for $x_k$ and compose matrix $X$ from these samples.

3.b. For $k = 1, 2, \ldots, K$ draw samples for $w_k$ and compose vector $W$ from these samples.

3.c. For $k = 1, 2, \ldots, K$ generate output $y_k$ using Equation (4) and compose vector $Y$ from these outputs.

3.d. Given matrix $X$ and vector $Y$, determine the (partial) least squares estimator $\hat{F}$ using the method selected in Step 2, where $\left(\hat{F}\right)^T = \left(\hat{f}_1 \quad \hat{f}_2 \quad \cdots \quad \hat{f}_{n_p}\right)$ and determine the estimate of the intercept term $\hat{f}_0$ .

**Step 4: Determine mean and standard deviation of least squares estimator over all MC simulation runs**

Determine the mean $\mu\left(\hat{f}_0\right)$, the mean $\mu\left(\hat{F}^T\right) = \left(\mu(\hat{f}_1) \quad \mu(\hat{f}_2) \quad \cdots \quad \mu(\hat{f}_{n_p})\right)$, the variance $Var\left(\hat{f}_0\right)$ and the covariance $Cov\left(\hat{F}\right)$ over the output values from the $N$ Monte Carlo simulation runs. Finally compare the estimated means and variances with the true values of $F$ and $f_0$.

## 4.2 Probability densities for $w_k$ and $x_k$

***Probability density functions of $w_k$***

Suppose that noise $w_k$ for $k = 1, 2, \ldots, K$ is an independent and identically distributed (i.i.d.) sequence of random variables. It is assumed that noise $w_k$ is Gaussian with mean $\mu_w = 0$ and standard deviation $\sigma_w > 0$, i.e. $w_t \sim N\left\{ \cdot ;\, 0, \sigma_w{}^2 \right\}$.

***Probability density functions of $x_k$***

For the probability density function of $x_k$ it is assumed that

- the process $x_k$ for $k = 1, 2, \ldots, K$ is a sequence of independent and identically distributed (i.i.d.) random variables;
- for each sample $x_k$ where $k = 1, 2, \ldots, K$, the random variables $x_{k,1}, x_{k,2}, \ldots, x_{k,n_p}$ are sampled uniformly from the intervals

$$\begin{aligned}\left[\tilde{z}_j^{LOW}, \tilde{z}_j^{HIGH}\right] &\triangleq \left[\tilde{z}_j + \ln\left(b_j / \ell_j\right), \tilde{z}_j + \ln\left(b_j\, \ell_j\right)\right] \\ &= \left[\tilde{z}_j + \ln\left(b_j\right) - \ln\left(\ell_j\right), \tilde{z}_j + \ln\left(b_j\right) + \ln\left(\ell_j\right)\right]\end{aligned} \tag{38}$$

for $j = 1, 2, \ldots, n_p$, where

- $\tilde{z}_j$ are chosen values for each $j = 1, 2, \ldots, n_p$ in $\tilde{z} = \mathrm{Col}\left(\tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_{n_p}\right)$
- bias parameters $b_j$ for $j = 1, 2, \ldots, n_p$ in $b \triangleq \mathrm{Col}\left(b_1, b_2, \ldots, b_{n_p}\right)$
- uncertainty parameters $\ell_j$ for $j = 1, 2, \ldots, n_p$ in $\ell \triangleq \mathrm{Col}\left(\ell_1, \ell_2, \ldots, \ell_{n_p}\right)$

## *4.3 Sampling methods*

Two sampling methods are used in Step 3a to draw samples according to the probability density function of $x_k$, i.e. Standard Random Sampling (SRS) and Latin Hypercube Sampling (LHS).

### 4.3.1 Standard Random Sampling (SRS)

In this case, for each sample $x_k = \mathrm{Col}\left(x_{k,1}, x_{k,2}, \ldots, x_{k,n_p}\right)$ where $k = 1, 2, \ldots, K$, it is assumed that the random variables $x_{k,1}, x_{k,2}, \ldots, x_{k,n_p}$ are mutually independent and each random variable $x_{k,j}$ for $j = 1, 2, \ldots, n_p$ is according to a uniform density on the the interval in (38).

### 4.3.2 Latin Hypercube Sampling (LHS)

In this case for each $x_k = \mathrm{Col}\left(x_{k,1}, x_{k,2}, \ldots, x_{k,n_p}\right)$ Latin Hypercube sampling is applied. Latin Hypercube picks $K$ different values from each of the $n_p$ random variables $x_{k,1}, x_{k,2}, \ldots, x_{k,n_p}$ from a uniform density as follows:

1. For each random variable $x_{k,j}$ for $j = 1, 2, \ldots, n_p$ in $x_k = \mathrm{Col}\left(x_{k,1}, x_{k,2}, \ldots, x_{k,n_p}\right)$ the interval $\left[\tilde{z}_j^{LOW}, \tilde{z}_j^{HIGH}\right] \triangleq \left[\tilde{z}_j + \ln\left(b_j / \ell_j\right), \tilde{z}_j + \ln\left(b_j\, \ell_j\right)\right]$ is divided into $K$ pieces such that there is equal probability per piece.
2. From each of the $K$ intervals a single value is sampled at random, according to a uniform density on that interval. This produces a sample of $K$ values for each input distribution that are more uniformly spread out than for standard random sampling.
3. The $K$ values thus obtained for $x_{k,1}$ are paired randomly (equally likely combinations) with the $K$ values of $x_{k,2}$. These $K$ pairs are combined in a

random manner with the $K$ values of $x_{k,3}$ to form $K$ triplets, and so on. Each pairing is done by associating a random permutation of the $K$ integers with each input variable.

This results in an $(K \times n_p)$ matrix[9] of input, where the $k^{\text{th}}$ row contains specific values of each of the $n_p$ input variables, to be used on the $k^{\text{th}}$ sample.

[9] This is matrix $X$ in (22)

# 5 Examples for use in MC simulation

This subsection gives an overview of examples that will be used for the Monte Carlo simulations. All examples have the same values for the number of components $n_p$, intercept term $f_0$, vector $F \triangleq \mathrm{Col}\left( f_1, f_2, \ldots, f_{n_p} \right)$, values $\tilde{x} \triangleq \mathrm{Col}\left( \tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_{n_p} \right)$, bias vector $b \triangleq \mathrm{Col}\left( b_1, b_2, \ldots, b_{n_p} \right)$ and uncertainty parameters $\ell \triangleq \mathrm{Col}\left( \ell_1, \ell_2, \ldots, \ell_{n_p} \right)$, and the same kind of the noise process $\{w_k\}$ is assumed as explained hereafter. The steps below refer to the steps in the Monte Carlo simulation approach as introduced in Subsection 4.1.

**Step 1: Setup**

The values of the parameters in the examples considered in this report, as specified by Step 1 (Setup) of the Monte Carlo simulation approach (as described in Subsection 4.1), are as follows:

1.a. For the number of components $n_p$, intercept term $f_0$ and vector $F^T = \left( f_1 \quad f_2 \quad \cdots \quad f_{n_p} \right)$, in all examples in Sections 6 and 7 the following values are assumed:

$n_p = 8$, $f_0 = 0.5$,
$f_1 = 0.1,\ f_2 = 0.2,\ f_3 = 0.4,\ f_4 = 0.8,\ f_5 = 1.6,\ f_6 = 3.2,\ f_7 = 6.4,\ f_8 = 12.8$

In Section 8 the following values are assumed:

$n_p = 200$, $f_0 = 0.5$, $(f_1 \quad \cdots \quad f_{200}) = (f_{init} \quad \cdots \quad f_{end}) = (0.1 \quad \cdots \quad 12)$, where $f_j = \mathrm{round}\left( f_{init} + \frac{j-1}{n_p - 1}( f_{end} - f_{init} ); 1 \right)$ for $j = 1, 2, \ldots, n_p$ where the function round means that the number in the first argument is rounded up til the nearest 1 (second argument in round function) decimal place.

1.b. For the probability density function for random variable $w_k$, in all examples the following is assumed (as in Subsection 4.2). Noise process $\{w_k\}$ for $k = 1, 2, \ldots, K$ is an independent and identically distributed (i.i.d.) sequence of random variables, with $w_k$ Gaussian with mean $\mu_w = 0$ and standard deviation $\sigma_w > 0$, i.e. $w_i \sim N\{ 0, \sigma_w{}^2 \}$. In the simulations, various values for $\sigma_w > 0$ will be used.

1.c. For the probability density function for random variable $x_k$, sampling is applied on the following interval (see also Subsection 4.2):

$$\left[\tilde{z}_j^{LOW}, \tilde{z}_j^{HIGH}\right] = \left[\tilde{z}_j + \ln\left(b_j\right) - \ln\left(\ell_j\right), \tilde{z}_j + \ln\left(b_j\right) + \ln\left(\ell_j\right)\right],$$

where in each of the examples it is assumed that $\tilde{z}_j = 1$, $b_j = 1$ and $\ell_j = 2$ for each $j = 1,2,...,n_p$. The choice for $\tilde{z}_j = 1$ is made for normalization. With these parameter values the sampling interval is

$$\left[\tilde{z}_j^{LOW}, \tilde{z}_j^{HIGH}\right] = \left[1 - \ln 2, 1 + \ln 2\right] \approx [0.31, 1.69].$$

1.d. In the simulations, the chosen values for *K*, will be varied from 2 through 50000.

In the examples considered in this report we will vary the values for *K* and for the standard deviation $\sigma_w$; all other parameter values are fixed.

**Step 2: Type of least squares estimation and sampling method**

2.a. Four types of least squares estimation are applied, i.e. CLS, LS-MP, PLS-N and PLS-S.

2.b. For each of the estimation types, both SRS and LHS are considered to draw samples for $x_k$.

This results in the following eight algorithms, though not all will be considered as becomes clear in the next section.

| Algorithm | Estimation type | Sampling type |
|---|---|---|
| A.1 | Classical Least Squares (CLS) | SRS |
| A.2 | Classical Least Squares (CLS) | LHS |
| B.1 | Least Squares with Moore-Penrose (LS-MP) | SRS |
| B.2 | Least Squares with Moore-Penrose (LS-MP) | LHS |
| C.1 | NIPALS based Partial Least Squares (PLS-N) | SRS |
| C.2 | NIPALS based Partial Least Squares (PLS-N) | LHS |
| D.1 | SIMPLS based Partial Least Squares (PLS-S) | SRS |
| D.2 | SIMPLS based Partial Least Squares (PLS-S) | LHS |

**Step 3: Evaluate least squares estimator using Monte Carlo simulation**

The chosen values for the number of Monte Carlo simulations runs, for each of the examples is $N = 100$.

Subsequently apply steps 3a-3d of the Monte Carlo simulation approach as introduced in Subsection 4.1 (see Page 38).

**Step 4: Determine mean and standard deviation of least squares estimator over all MC simulation runs**

Determine mean and standard deviations.

# 6 Monte Carlo simulation results

In this section the results of the Monte Carlo simulations are discussed for each of the following algorithms.

| Algorithm | Estimation type | Sampling type |
|---|---|---|
| A.1 | Classical Least Squares (CLS) | SRS |
| A.2 | Classical Least Squares (CLS) | LHS |
| B.1 | Least Squares with Moore-Penrose (LS-MP) | SRS |
| B.2 | Least Squares with Moore-Penrose (LS-MP) | LHS |
| C.1 | NIPALS based Partial Least Squares (PLS-N) | SRS |
| D.1 | SIMPLS based Partial Least Squares (PLS-S) | SRS |

In the simulations, the chosen values for $K$ and the chosen values for standard deviation $\sigma_w$, have been varied from low to high.

In this section the following is discussed:

- Results for Classical Least Squares (CLS) with both SRS and LHS are discussed in Subsection 6.1.
- Results for Least Squares with Moore-Penrose (LS-MP) with both SRS and LHS are discussed in Subsection 6.2.
- Results for NIPALS based Partial Least Squares (PLS-N) compared to results for LS-MP are discussed in Subsection 6.3.
- Results for SIMPLS based Partial Least Squares (PLS-S) compared to results for LS-MP are discussed in Subsection 6.4.
- A summarising discussion is given in Subsection 6.5.

## 6.1 Discussion of results for CLS Algorithms A.1 and A.2

Results for Classical Least Squares (CLS) with both sampling types SRS and LHS are presented in this subsection. As explained in Subsection 3.3, in order to apply CLS, the following should hold true: matrix $X_0^T X_0$ must be invertible, i.e. $\text{rank}\left(X_0^T X_0\right) = n_p$. This can only happen for $K \geq n_p + 1$, not for $K \leq n_p$. This means that the CLS approach is not suitable for $K \leq n_p$. So the results presented in this subsection are for values of $K$ larger than or equal to the number of regression components $n_p$.

The specific results that are presented in this subsection are:

- The mean $\mu(\hat{f}_j)$ and the true values for $f_j$ for each $j = 1, 2, \ldots, n_p$.
- The standard deviation $\sigma(\hat{f}_j)$ divided by $\sigma_w / \sqrt{K}$, i.e. $\sigma(\hat{f}_j)\sqrt{K} / \sigma_w$, which is referred to as the normalized standard deviation of $\sigma(\hat{f}_j)$. The reason for doing so is that the standard deviations $\sigma(\hat{f}_j)$ are expected to be proportional to the standard deviation $\sigma_w$ and inversely proportional to the square root of $K$ for each $j = 1, 2, \ldots, n_p$.

This subsection is organised as follows:

- In Subsection 6.1.1, we vary $K$ for a fixed standard deviation $\sigma_w$.
- In Subsection 6.1.2, we vary standard deviation $\sigma_w$ for a fixed $K$.
- In Subsection 6.1.3, we vary standard deviation $\sigma_w$ for a fixed $K$ close to the number of regression components $n_p$.
- In Subsection 6.1.4, the results obtained in Subsections 6.1.1 through 6.1.3 for CLS algorithms A.1 and A.2 are discussed.

### 6.1.1 Variation of *K* and fixed $\sigma_w$

Consider algorithm A.1, i.e. for CLS with SRS, for a fixed value of the standard deviation $\sigma_w$ and with variation of $K$, as specified in the following table:

| **Algorithm** | **Values for *K*** | **Values for $\sigma_w$** |
|---|---|---|
| A.1 (CLS with SRS) | $K$ = {9, 10, 11, 12, 14, 16, 18, 20, 50, 100} | $\sigma_w = 0.5$ |

The following two figures present:

- The mean $\mu(\hat{f}_j)$ as a function of the parameter index $j = 1, 2, \ldots, n_p$ for each of the values of $K$ as specified in the above table, and the true values for $(f_1 \quad f_2 \quad \cdots \quad f_8) = (0.1 \quad 0.2 \quad 0.4 \quad 0.8 \quad 1.6 \quad 3.2 \quad 6.4 \quad 12.8)$.
- The normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for each of the parameter indices $j = 1, 2, \ldots, n_p$ as a function of $K$. To make the figures readable, both the horizontal and vertical axes are presented in a logarithmic scale.

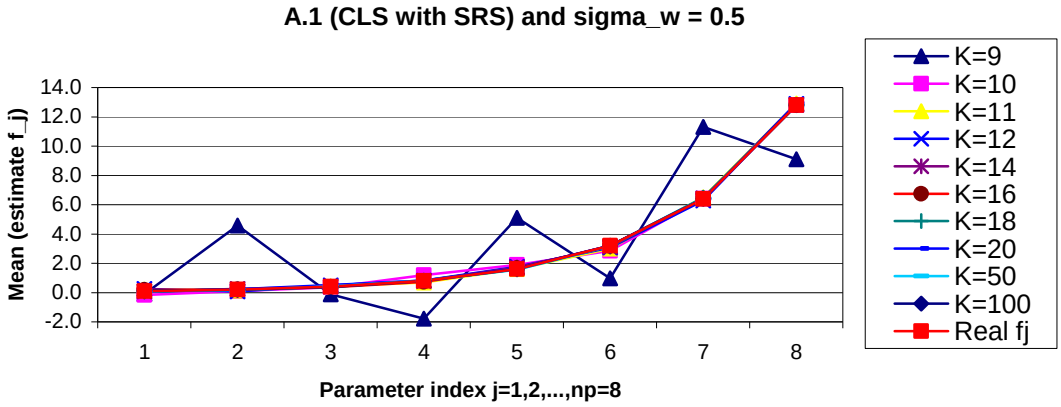


**Figure 3. Mean $\mu(\hat{f}_j)$ values as a function of j = 1,2,...,$n_p$ with algorithm A.1 (CLS with SRS) for $\sigma_w$ =0.5 and $K$ = {9, 10, 11, 12, 14, 16, 18, 20, 50, 100}. The red line indicated as 'Real fj' represents the true values for $f_j$.**

Remark: From this point on in several of the figures, the legend sometimes refer to m1 through m8, this should be read as f1 through f8 (This applies to Figures 4, 7, 24 and 30-48). Moreover real fj should be read as true $f_j$.

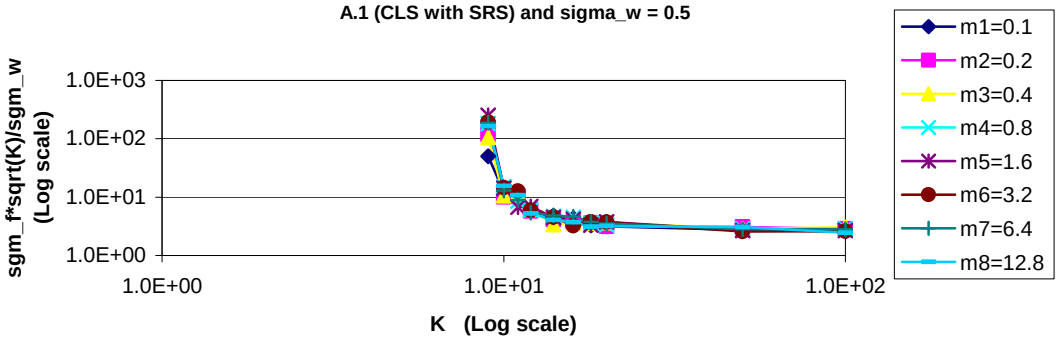


**Figure 4. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,..., $n_p$ with algorithm A.1 (LS-MP with SRS) for $\sigma_w$ =0.5 as a function of $K$ = {9, 10, 11, 12, 14, 16, 18, 20, 50, 100}.**

To show the effect of the type of sampling technique SRS or LHS, consider both algorithms A.1 and A.2, i.e. for CLS with SRS and LHS respectively, for the same fixed value of the standard deviation $\sigma_w$ and variation of $K$, as specified in the table above, that is:

| **Algorithm** | **Values for *K*** | **Values for $\sigma_w$** |
|---|---|---|
| A.1 (CLS with SRS; black line) and A.2 (CLS with LHS; red line) | $K$ = {9, 10, 11, 12, 14, 16, 18, 20, 50, 100} | $\sigma_w$ = 0.5 |

The following figure presents:

- The normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for $j$ = 1, i.e. for parameter $m_1$ = 0.1, as a function of $K$ for algorithm A.1 (i.e. with SRS; black line) and algorithm A.2 (i.e. with LHS; red line), and where both the horizontal and vertical axes are presented in a logarithmic scale.

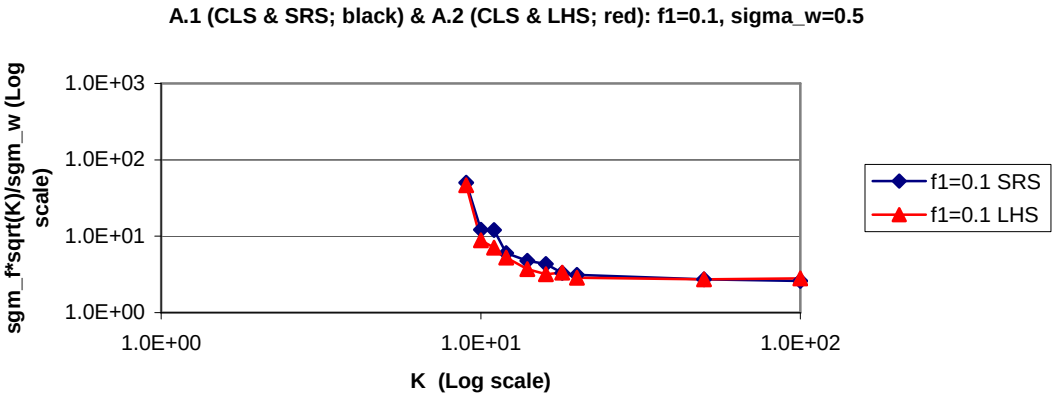


**Figure 5. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1, i.e. for parameter $f_1$ = 0.1, with algorithms A.1 (CLS with SRS) and A.2 (CLS with SRS) for $\sigma_w$ =0.5 as a function of *K* = {9, 10, 11, 12, 14, 16, 18, 20, 50, 100}.**

### 6.1.2 Variation of $\sigma_w$ and fixed *K*

Consider algorithm A.2, i.e. for CLS with LHS, for a fixed value of $K$ and variation of standard deviation $\sigma_w$, as specified in the following table:

| **Algorithm** | **Values for *K*** | **Values for $\sigma_w$** |
|---|---|---|
| A.2 (CLS with LHS) | $K$ = 16 | $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10} |

The following two figures present:

- The mean $\mu(\hat{f}_j)$ as a function of the parameter index $j = 1, 2, \ldots, n_p$ for each of the values of $\sigma_w$ as specified in the above table, and the true values for $(f_1 \quad f_2 \quad \cdots \quad f_8) = (0.1 \quad 0.2 \quad 0.4 \quad 0.8 \quad 1.6 \quad 3.2 \quad 6.4 \quad 12.8)$.
- The normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for each of the parameter indices $j = 1, 2, \ldots, n_p$ as a function of $\sigma_w$ where both the horizontal and vertical axes are presented in a logarithmic scale.

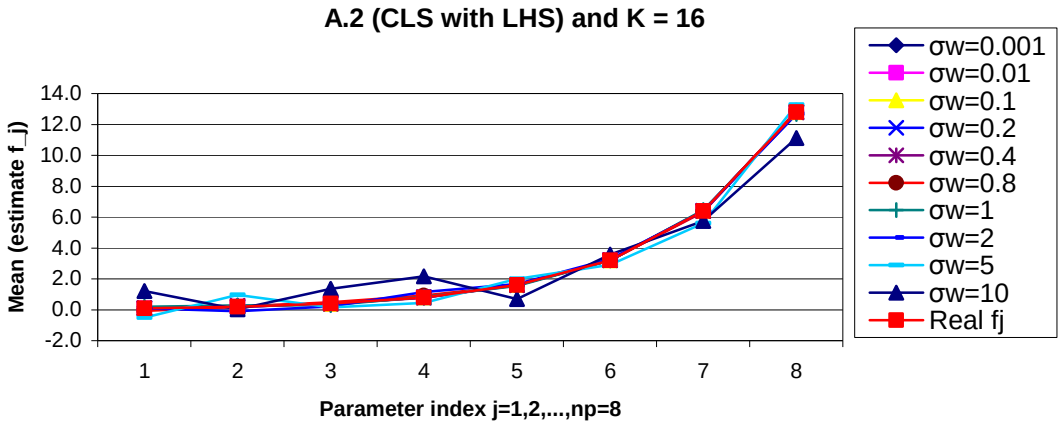


**Figure 6. Mean $\mu(\hat{f}_j)$ values as a function of j = 1,2,..., $n_p$ with algorithm A.2 (CLS with LHS) for $K$ = 16 and $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10}. The red line indicated as 'Real fj' represents the true values for $f_j$.**

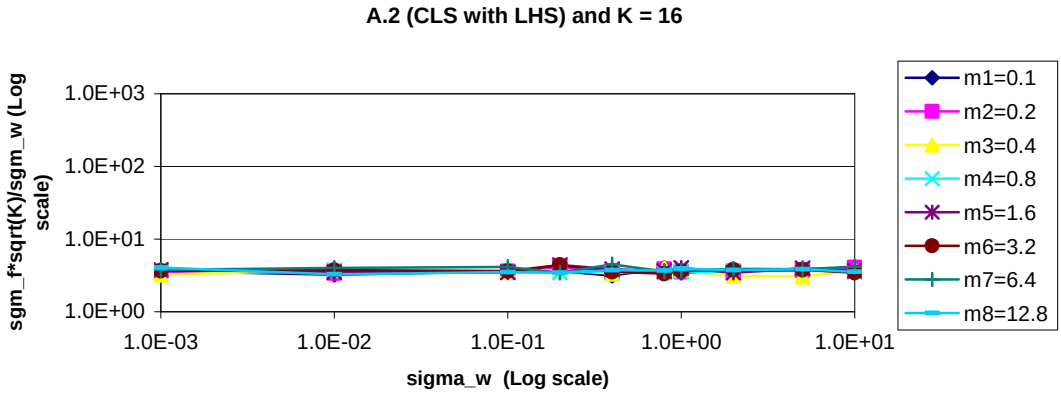


**Figure 7. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$n_p$ with algorithm A.2 (CLS with LHS) for $K$ = 16 as a function of $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10}.**

To show the effect of the type of sampling technique SRS or LHS, consider both algorithms A.1 and A.2, i.e. for CLS with SRS and LHS respectively, for the same fixed value of $S$ and variation of standard deviation $\sigma_w$, as specified in the table above, that is:

| **Algorithm** | **Values for *K*** | **Values for $\sigma_w$** |
|---|---|---|
| A.1 (CLS with SRS; black line) and A.2 (CLS with LHS; red line) | $K = 16$ | $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10} |

The following figure presents:

- The normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for $j = 5$, i.e. for parameter $f_5 = 1.6$, as a function of $\sigma_w$ for algorithm A.1 (i.e. with SRS; black line) and algorithm A.2 (i.e. with LHS; red line), and where both the horizontal and vertical axes are presented in a logarithmic scale

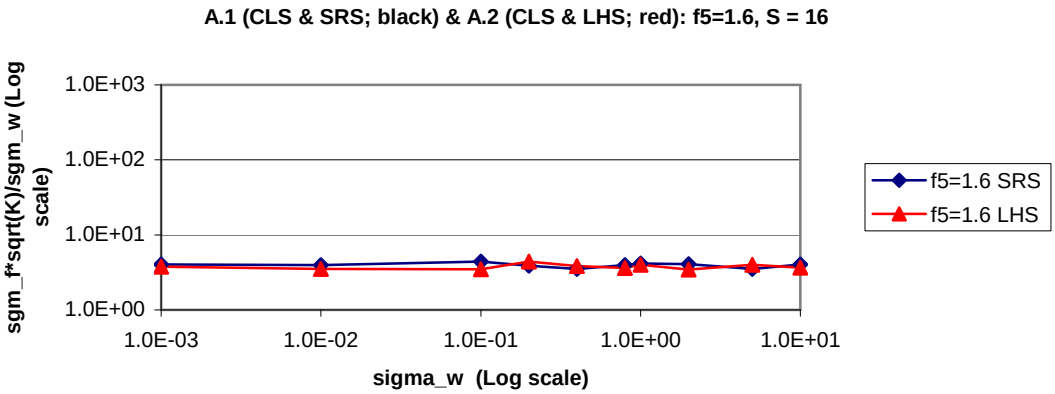


**Figure 8. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 5, i.e. for parameter $f_5$ = 1.6, with algorithms A.1 (CLS with SRS) and A.2 (CLS with SRS) for *K* = 16 as a function of $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10}.**

### 6.1.3 Variation of $\sigma_w$ and fixed $K$ close to $n_p$

Consider algorithm A.2, i.e. for CLS with LHS, for a fixed value of $K$ and variation of standard deviation $\sigma_w$, as specified in the following table:

| Algorithm | Values for $K$ | Values for $\sigma_w$ |
|---|---|---|
| A.1 (CLS with SRS) | $K = 9$ | $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10} |

The following figures present:

- The mean $\mu(\hat{f}_j)$ as a function of the parameter index $j = 1, 2, \ldots, n_p$ for $K = 9$ and $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2} and the true values for $(f_1 \quad f_2 \quad \cdots \quad f_8) = (0.1 \quad 0.2 \quad 0.4 \quad 0.8 \quad 1.6 \quad 3.2 \quad 6.4 \quad 12.8)$. For this case, the obtained values of the mean $\mu(\hat{f}_j)$ become rather inaccurate in case $\sigma_w$ = 5 or 10, therefore they have not been shown in the figure below.
- The normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for each of the parameter indices $j = 1, 2, \ldots, n_p$ as a function of $\sigma_w$ where both the horizontal and vertical axes are presented in a logarithmic scale.

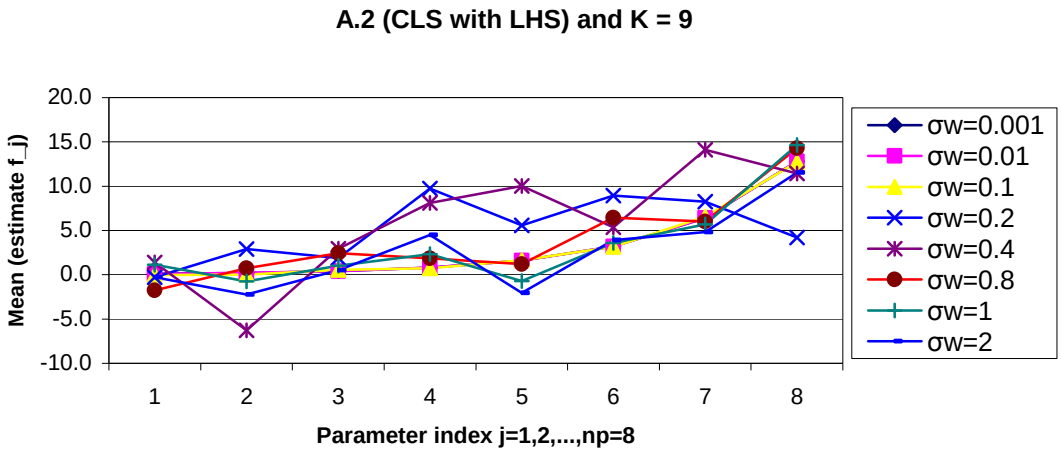


**Figure 9. Mean $\mu(\hat{f}_j)$ values as a function of j = 1,2,...,$n_p$ with algorithm A.2 (CLS with LHS) for $K$ = 9 and $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2}. The scale of the vertical axis in Figure 9 is larger than in Figure 3 or Figure 6.**

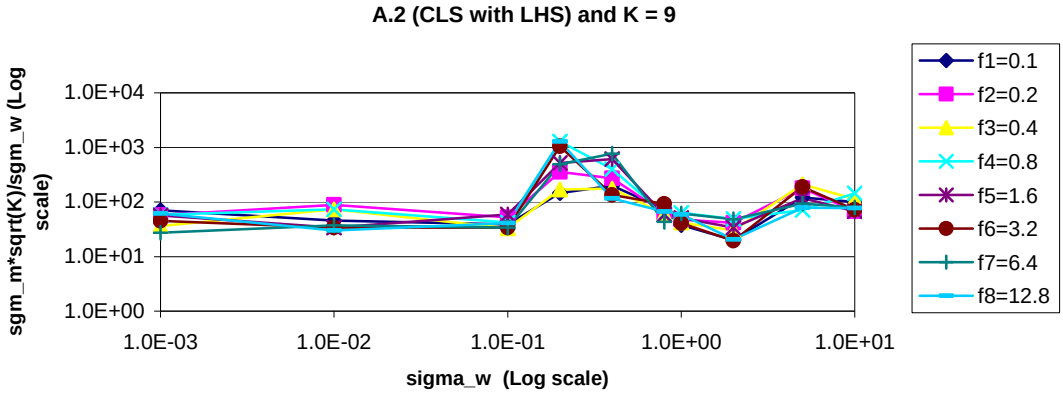


**Figure 10. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$n_p$ with algorithm A.2 (CLS with LHS) for $K$ = 9 as a function of $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10}.**

Consider algorithm A.1, i.e. for CLS with SRS, for a fixed value of $K$ and variation of standard deviation $\sigma_w$, as specified in the following table:

| **Algorithm** | **Values for *K*** | **Values for $\sigma_w$** |
|---|---|---|
| A.1 (CLS with SRS) | $K$ = 10 | $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10} |

The following two figures present:

- The mean $\mu(\hat{f}_j)$ as a function of the parameter index $j = 1,2,\ldots,n_p$ for each of the values of $\sigma_w$ as specified in the above table, and the true values for $(f_1 \quad f_2 \quad \cdots \quad f_8) = (0.1 \quad 0.2 \quad 0.4 \quad 0.8 \quad 1.6 \quad 3.2 \quad 6.4 \quad 12.8)$.
- The normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for each of the parameter indices $j = 1,2,\ldots,n_p$ as a function of $\sigma_w$ where both the horizontal and vertical axes are presented in a logarithmic scale.

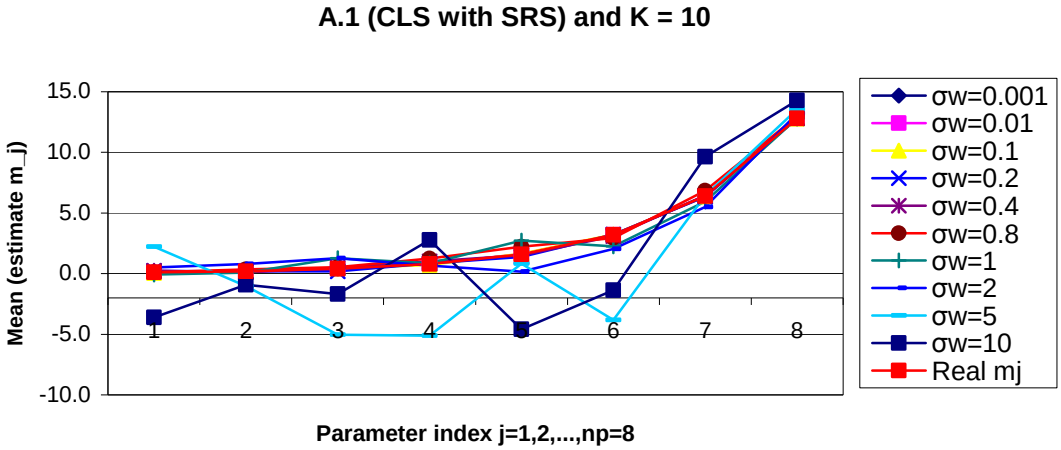


**Figure 11. Mean $\mu(\hat{f}_j)$ values as a function of j = 1,2,...,$n_p$ with algorithm A.1 (CLS with SRS) for $K$ = 10 and $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10}. The red line indicated as 'Real fj' represents the true values for $f_j$.**

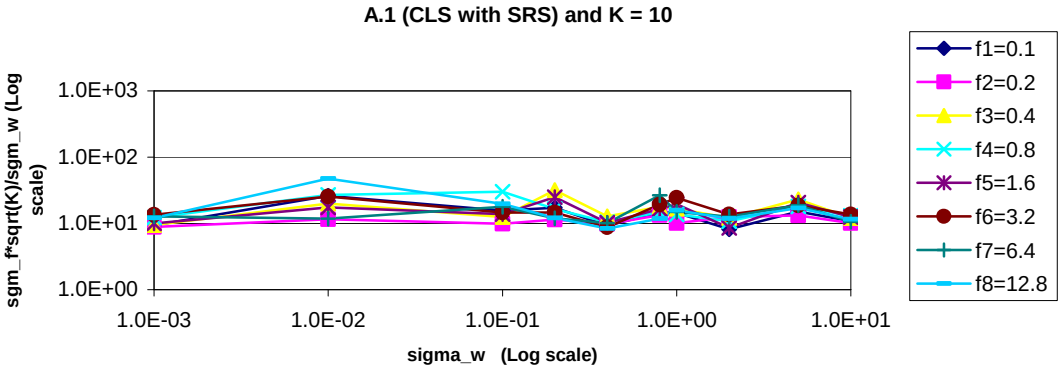


**Figure 12. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$n_p$ with algorithm A.1 (CLS with SRS) for $K$ = 10 as a function of $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10}.**

To show the effect of the type of sampling technique SRS or LHS, consider both algorithms A.1 and A.2, i.e. for CLS with SRS and LHS respectively, for the same fixed value of $K$ and variation of standard deviation $\sigma_w$, as specified in the table above, that is:

| **Algorithm** | **Values for *K*** | **Values for $\sigma_w$** |
|---|---|---|
| A.1 (CLS with SRS; black line) and A.2 (CLS with LHS; red line) | $K = 10$ | $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10} |

The following figure presents:

- The normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for $j = 8$, i.e. for parameter $f_8 = 12.8$, as a function of $\sigma_w$ for algorithm A.1 (i.e. with SRS; black line) and algorithm A.2 (i.e. with LHS; red line), and where both the horizontal and vertical axes are presented in a logarithmic scale.

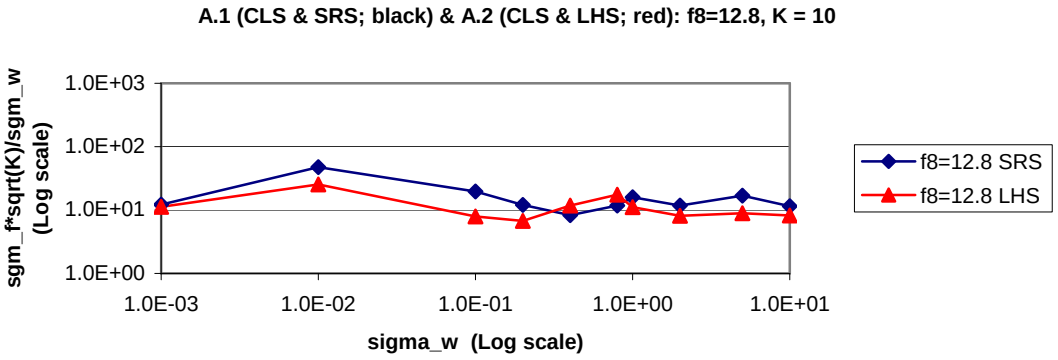


**Figure 13. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 8, i.e. for parameter $f_8$ = 12.8, with algorithms A.1 (LS-MP with SRS) and B.2 (LS-MP with SRS) for *K* = 10 as a function of $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10}.**

### 6.1.4 Discussion of results for CLS Algorithms A.1 and A.2

Here, a summarising discussion is given of the results presented in Subsections 6.1.1 through 6.1.3 for CLS algorithms A.1 and A.2.

**Mean $\mu(\hat{f}_j)$:**

- The results show that for most values of $K \geq 9$ and $\sigma_w$, the obtained values for the mean $\mu(\hat{f}_j)$ approach the true values of $f_j$ very accurately, though with the exception of $K$ being close to $n_p$ or of $\sigma_w$ being very large. For example for $K = 9$ in Figure 3 and in Figure 9 the obtained values for the mean $\mu(\hat{f}_j)$ become inaccurate, similarly for $\sigma_w = 5$ and $\sigma_w = 10$ in Figure 6 and Figure 11 respectively.

**Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$:**

- For $K$ close to $n_p$ (i.e. $K = 9$ and 10), the obtained values for the normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ sometimes become relatively large (see Figure 4 and Figure 12).
  - For $K = 9$ and $\sigma_w$ = { 0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10 } the values of the normalized standard deviations $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for all $j = 1, 2, \ldots, n_p$ in Figure 10 vary between 19 and 1290.
  - For $K = 10$ and $\sigma_w$ = { 0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10 } the values of the normalized standard deviations $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for all $j = 1, 2, \ldots, n_p$ in Figure 12 vary between 8 and 48. For $\sigma_w = 0.01$, the obtained values for $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ vary between 11 and 48, whereas for $\sigma_w = 0.001$, 0.4 and 10, the obtained values for $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ vary between 8 and 14.
  - For $K = 9$, large values of $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ occur more often than for $K = 10$.
- For $K = 16$ and $\sigma_w$ = { 0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10 } the values of the normalized standard deviations $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for all $j = 1, 2, \ldots, n_p$, fall all within the range of 3.0 to 4.7 (See Figure 7).
- For $K > 9$ (i.e. $K > n_p + 1$), the results show that the larger $K$ becomes, the smaller the obtained values for $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for $j = 1, 2, \ldots, n_p$ become. For example in Figure 4 with $\sigma_w = 0.5$, the obtained values for the normalized standard deviations $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for $K = 10$ fall within the range 7-15.5 and for $K = 100$ fall within the range 2.3-3.

**LHS versus SRS:**

- For all $K \geq 9$ and for all $\sigma_w$ as considered in the simulations, sometimes SRS yields smaller values of the normalized standard deviations $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$, and sometimes LHS does (See Figure 5, Figure 8 and Figure 13). However no systematic differences can be observed between results obtained with SRS and LHS.

## 6.2 Discussion of results for LS-MP Algorithms B.1 and B.2

Results for Least Squares with Moore-Penrose (LS-MP) with both sampling types SRS and LHS are presented in this subsection.

This subsection is organised as follows:

- In Subsection 6.2.1, we vary $K$ for a fixed low value of standard deviation $\sigma_w$.
- In Subsection 6.2.2, we vary $K$ for a fixed high value of standard deviation $\sigma_w$.
- In Subsection 6.2.3, we vary standard deviation $\sigma_w$ for a fixed $K$ (= 16).
- In Subsection 6.2.4, we vary standard deviation $\sigma_w$ for a fixed $K$ (= 10) close to the number of regression components $n_p$.
- In Subsection 6.2.5, the results obtained in Subsections 6.2.1 through 6.2.4 for LS-MP algorithms B.1 and B.2 are discussed.
- In Subsection 6.2.6, the results obtained for CLS algorithm A.1 and LS-MP algorithm B.1 (based on Subsections 6.1.1- 6.1.3 and Subsections 6.2.1- 6.2.4), both with SRS are discussed.

### 6.2.1 Variation of *K* and fixed low value of $\sigma_w$

Consider algorithm B.2, i.e. for LS-MP with LHS, for a fixed value of the standard deviation $\sigma_w$ and with variation of $K$, as specified in the following table:

| **Algorithm** | **Values for *K*** | **Values for $\sigma_w$** |
|---|---|---|
| B.2 (LS-MP with LHS) | $K$ = {4, 6, 8, 9, 10, 25, 50, 100, 500, 1000} | $\sigma_w$ = 0.01 |

The following two figures present:

- The mean $\mu(\hat{f}_j)$ as a function of the parameter index $j = 1, 2, \ldots, n_p$ for each of the values of $K$ as specified in the above table, and the true values for $(f_1 \quad f_2 \quad \cdots \quad f_8) = (0.1 \quad 0.2 \quad 0.4 \quad 0.8 \quad 1.6 \quad 3.2 \quad 6.4 \quad 12.8)$.
- The normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for each of the parameter indices $j = 1, 2, \ldots, n_p$ as a function of $K$ where both the horizontal and vertical axes are presented in a logarithmic scale.

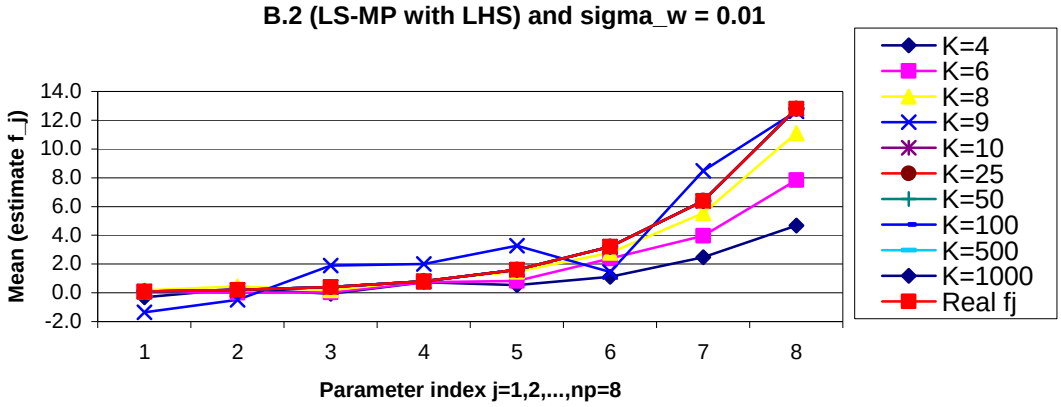


**Figure 14. Mean $\mu(\hat{f}_j)$ values as a function of j = 1,2,...,$n_p$ with algorithm B.2 (LS-MP with LHS) for $\sigma_w$ =0.01 and $K$ = {4, 6, 8, 9, 10, 25, 50, 100, 500, 1000}. The red line indicated as 'Real fj' represents the true values for $f_j$.**

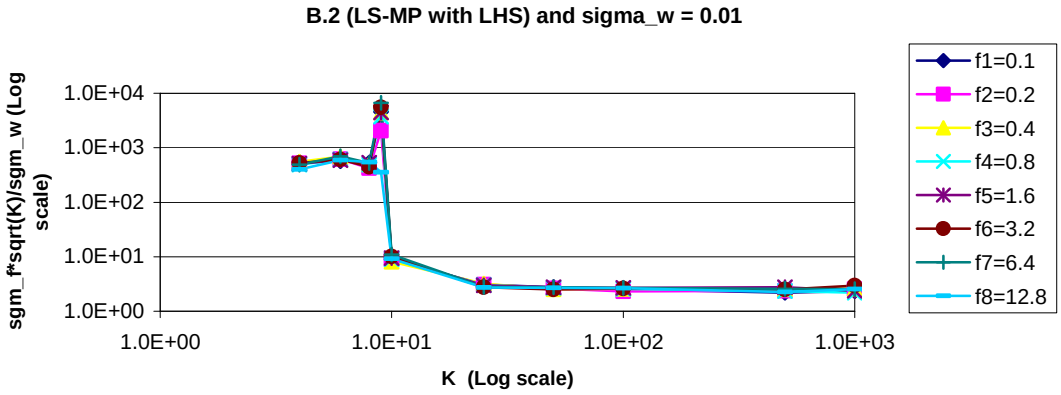


**Figure 15. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$n_p$ with algorithm B.2 (LS-MP with LHS) for $\sigma_w$ =0.01 as a function of $K$ = {4, 6, 8, 9, 10, 25, 50, 100, 500, 1000}.**

To show the effect of the type of sampling technique SRS or LHS, consider both algorithms B.1 and B.2, i.e. for LS-MP with SRS and LHS respectively, for the same fixed value of the standard deviation $\sigma_w$ and variation of $K$, as specified in the table above, that is:

| **Algorithm** | **Values for $K$** | **Values for $\sigma_w$** |
|---|---|---|
| B.1 (LS-MP with SRS; black line) and B.2 (LS-MP with LHS; red line) | $K$ = {4, 6, 8, 9, 10, 25, 50, 100, 500, 1000} | $\sigma_w$ = 0.01 |

The following figure presents:

- The normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for $j$ = 1, i.e. for parameter $f_1$ = 0.1, as a function of $K$ for algorithm B.1 (i.e. with SRS; black line) and algorithm B.2 (i.e. with LHS; red line), and where both the horizontal and vertical axes are presented in a logarithmic scale.

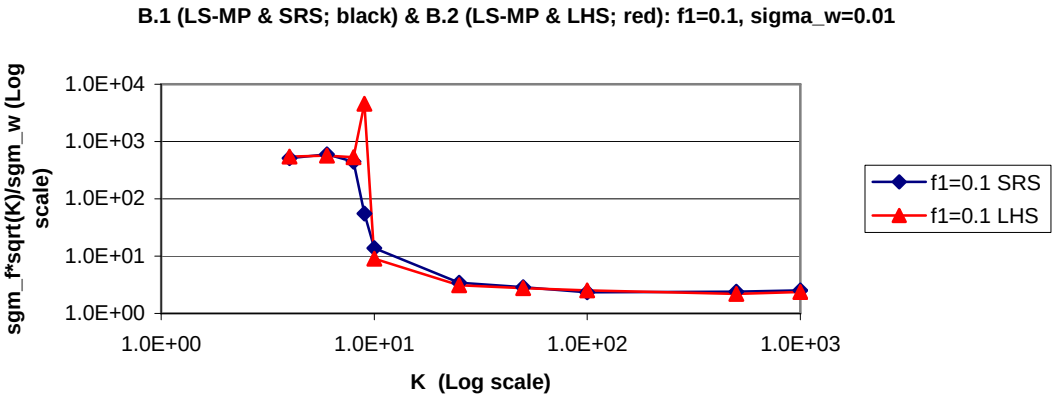


**Figure 16. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1, i.e. for parameter $f_1$ = 0.1, with algorithms B.1 (LS-MP with SRS) and B.2 (LS-MP with LHS) for $\sigma_w$ = 0.01 as a function of $K$ = {4, 6, 8, 9, 10, 25, 50, 100, 500, 1000}.**

### 6.2.2 Variation of *K* and fixed high value of $\sigma_w$

Consider algorithm B.1, i.e. for LS-MP with SRS, for a fixed value of the standard deviation $\sigma_w$ and with variation of $K$, as specified in the following table:

| Algorithm | Values for *K* | Values for $\sigma_w$ |
|---|---|---|
| B.1 (LS-MP with SRS) | $K$ = {2, 4, 6, 8, 9, 10, 11, 20, 35, 100} | $\sigma_w$ = 0.5 |

The following two figures present:

- The mean $\mu(\hat{f}_j)$ as a function of the parameter index $j = 1, 2, \ldots, n_p$ for each of the values of $K$ as specified in the above table, and the true values for $(f_1 \quad f_2 \quad \cdots \quad f_8) = (0.1 \quad 0.2 \quad 0.4 \quad 0.8 \quad 1.6 \quad 3.2 \quad 6.4 \quad 12.8)$.
- The normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for each of the parameter indices $j = 1, 2, \ldots, n_p$ as a function of $K$ where both the horizontal and vertical axes are presented in a logarithmic scale.

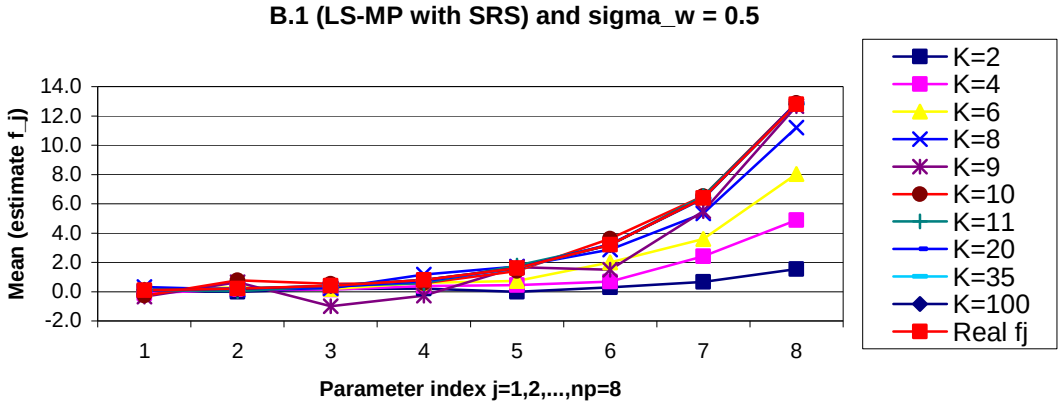


**Figure 17. Mean $\mu(\hat{f}_j)$ values as a function of j = 1,2,...,$n_p$ with algorithm B.1 (LS-MP with SRS) for $\sigma_w$ = 0.5 and $K$ = {2, 4, 6, 8, 9, 10, 11, 20, 35, 100}. The red line indicated as 'Real fj' represents the true values for $f_j$.**

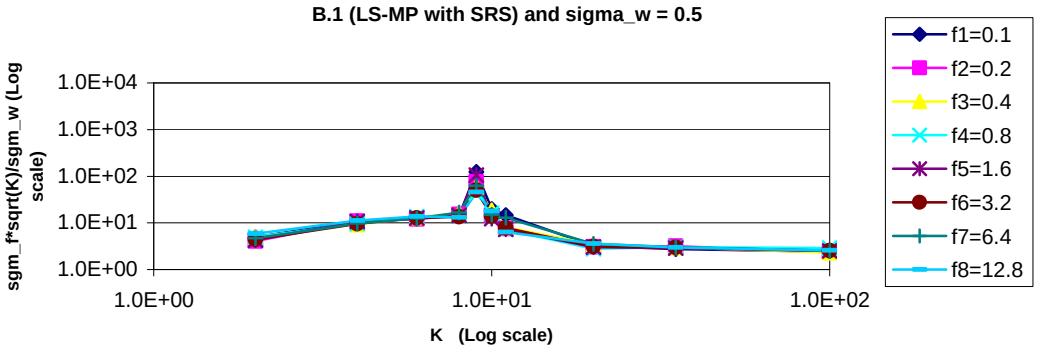


**Figure 18. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$n_p$ with algorithm B.1 (LS-MP with SRS) for $\sigma_w$ = 0.5 as a function of $K$ = {2, 4, 6, 8, 9, 10, 11, 20, 35, 100}.**

To show the effect of the type of sampling technique SRS or LHS, consider both algorithms B.1 and B.2, i.e. for LS-MP with SRS and LHS respectively, for the same fixed value of the standard deviation $\sigma_w$ and variation of $K$, as specified in the table above, that is:

| **Algorithm** | **Values for $K$** | **Values for $\sigma_w$** |
|---|---|---|
| B.1 (LS-MP with SRS; black line) and B.2 (LS-MP with LHS; red line) | $K$ = {2, 4, 6, 8, 9, 10, 11, 20, 35, 100} | $\sigma_w$ = 0.5 |

The following figure presents:

- The normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for $j$ = 1, i.e. for parameter $f_1$ = 0.1, as a function of $K$ for algorithm B.1 (i.e. with SRS; black line) and algorithm B.2 (i.e. with LHS; red line), and where both the horizontal and vertical axes are presented in a logarithmic scale

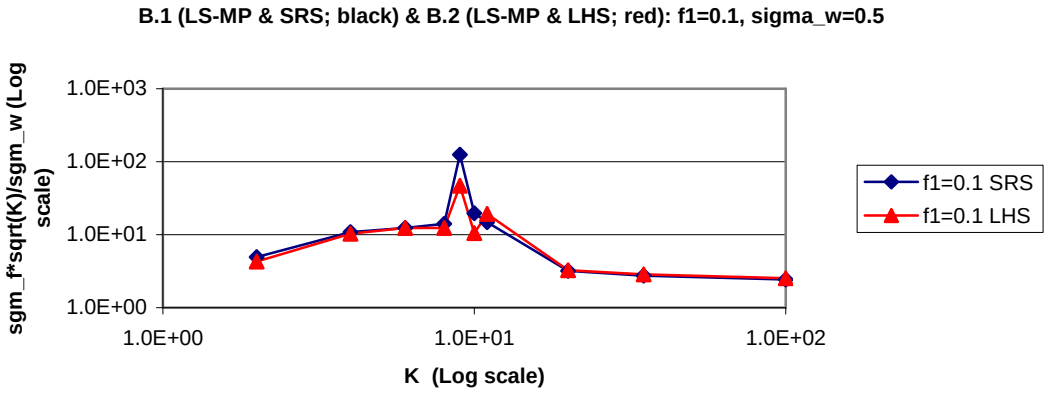


**Figure 19. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1, i.e. for parameter $f_1$ = 0.1, with algorithms B.1 (LS-MP with SRS) and B.2 (LS-MP with SRS) for $\sigma_w$ = 0.5 as a function of $K$ = {2, 4, 6, 8, 9, 10, 11, 20, 35, 100}.**

### 6.2.3 Variation of $\sigma_w$ value for $K$ = 16

Consider algorithm B.1, i.e. for LS-MP with SRS, for a fixed value of $K$ and variation of standard deviation $\sigma_w$, as specified in the following table:

| Algorithm | Values for $K$ | Values for $\sigma_w$ |
|---|---|---|
| B.1 (LS-MP with SRS) | $K$ = 16 | $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10} |

The following two figures present:

- The mean $\mu(\hat{f}_j)$ as a function of the parameter index $j = 1, 2, \ldots, n_p$ for each of the values of $\sigma_w$ as specified in the above table, and the true values for $(f_1 \quad f_2 \quad \cdots \quad f_8) = (0.1 \quad 0.2 \quad 0.4 \quad 0.8 \quad 1.6 \quad 3.2 \quad 6.4 \quad 12.8)$.
- The normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for each of the parameter indices $j = 1, 2, \ldots, n_p$ as a function of $\sigma_w$ where both the horizontal and vertical axes are presented in a logarithmic scale.

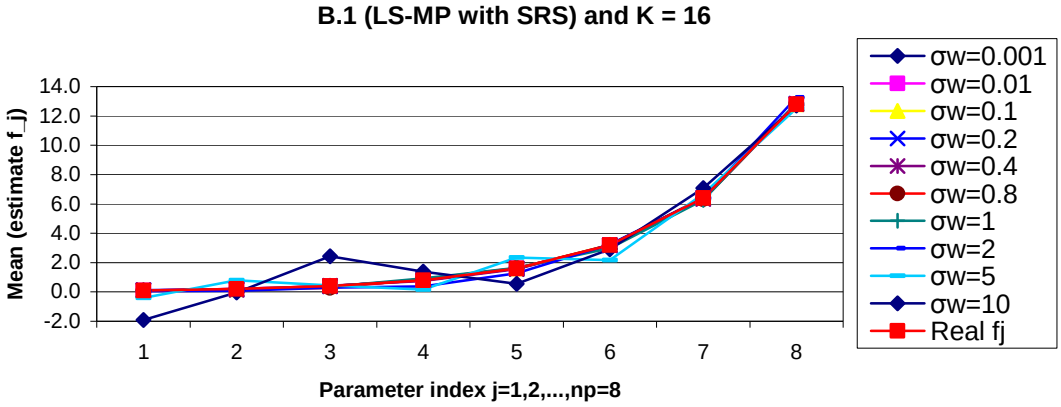


**Figure 20. Mean $\mu(\hat{f}_j)$ values as a function of j = 1,2,...,$n_p$ with algorithm B.1 (LS-MP with SRS) for $K$ = 16 and $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10}. The red line indicated as 'Real fj' represents the true values for $f_j$.**

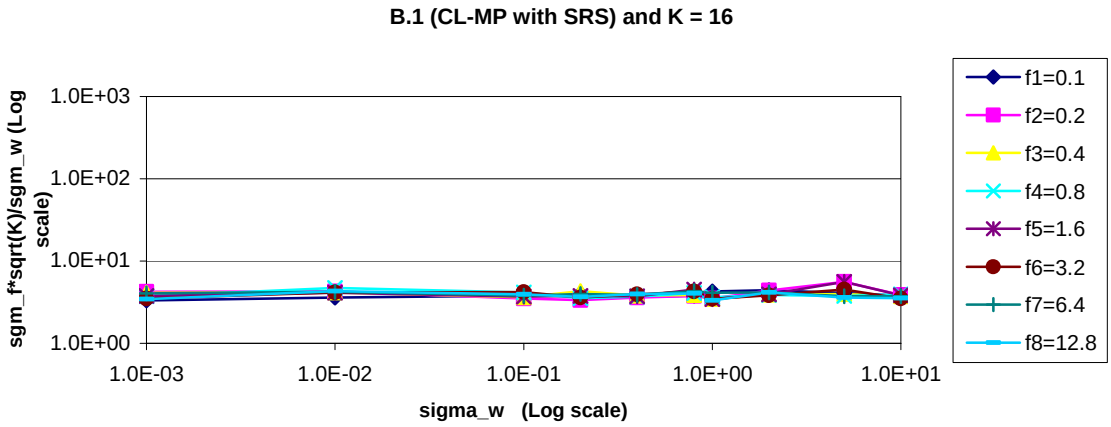


**Figure 21. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$n_p$ with algorithm B.1 (LS-MP with SRS) for $K$ = 16 as a function of $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10}.**

To show the effect of the type of sampling technique SRS or LHS, consider both algorithms B.1 and B.2, i.e. for LS-MP with SRS and LHS respectively, for the same fixed value of $K$ and variation of standard deviation $\sigma_w$, as specified in the table above, that is:

| **Algorithm** | **Values for $K$** | **Values for $\sigma_w$** |
|---|---|---|
| B.1 (LS-MP with SRS; black line) and B.2 (LS-MP with LHS; red line) | $K$ = 16 | $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10} |

The following figure presents:

- The normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for $j$ = 1, i.e. for parameter $f_1$ = 0.1, as a function of $\sigma_w$ for algorithm B.1 (i.e. with SRS; black line) and algorithm B.2 (i.e. with LHS; red line), and where both the horizontal and vertical axes are presented in a logarithmic scale.

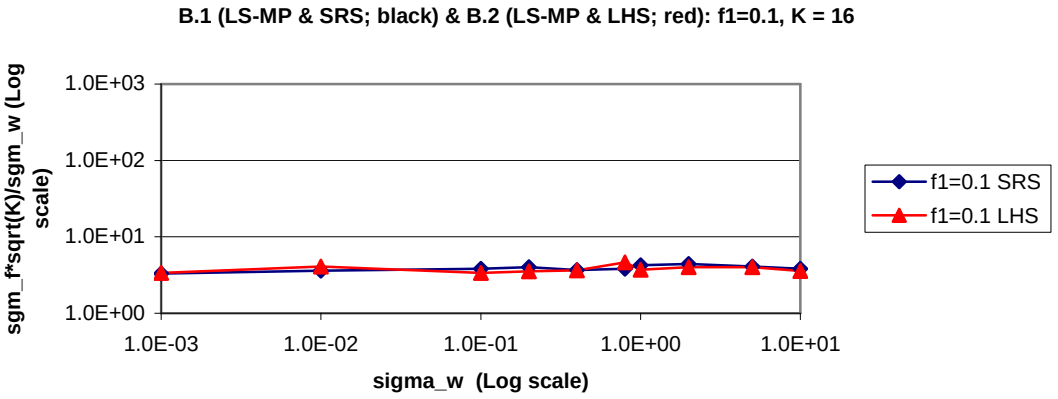


**Figure 22. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1, i.e. for parameter $f_1$ = 0.1, with algorithms B.1 (LS-MP with SRS) and B.2 (LS-MP with SRS) for *K* = 16 as a function of $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10}.**

### 6.2.4 Variation of $\sigma_w$ value for *K* = 10

Consider algorithm B.1, i.e. for LS-MP with SRS, for a fixed value of $K$ and variation of standard deviation $\sigma_w$, as specified in the following table:

| Algorithm | Values for *K* | Values for $\sigma_w$ |
|---|---|---|
| B.1 (LS-MP with SRS) | $K$ = 10 | $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10} |

The following two figures present:

- The mean $\mu(\hat{f}_j)$ as a function of the parameter index $j = 1, 2, \ldots, n_p$ for each of the values of $\sigma_w$ as specified in the above table, and the true values for $(f_1 \quad f_2 \quad \cdots \quad f_8) = (0.1 \quad 0.2 \quad 0.4 \quad 0.8 \quad 1.6 \quad 3.2 \quad 6.4 \quad 12.8)$.
- The normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for each of the parameter indices $j = 1, 2, \ldots, n_p$ as a function of $\sigma_w$ where both the horizontal and vertical axes are presented in a logarithmic scale.

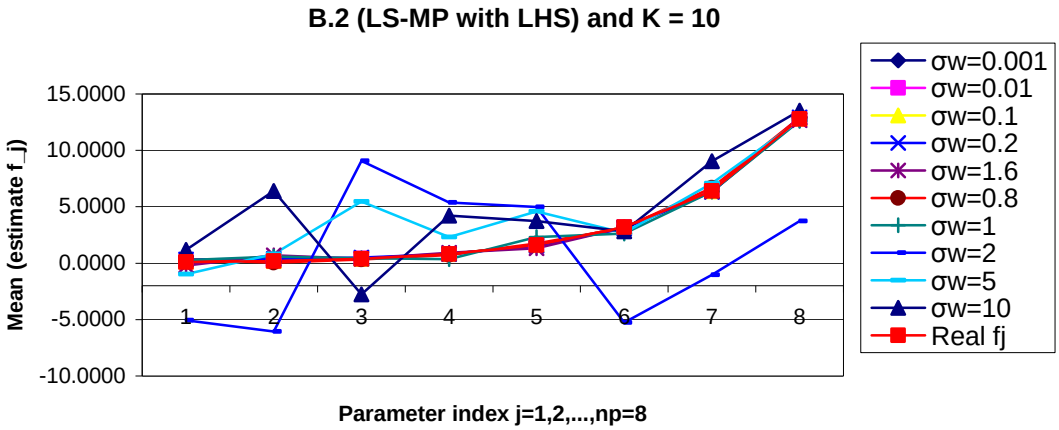


**Figure 23. Mean $\mu(\hat{f}_j)$ values as a function of j = 1,2,...,$n_p$ with algorithm B.2 (LS-MP with LHS) for $K$ = 10 and $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10}. The red line indicated as 'Real fj' represents the true values for $f_j$.**

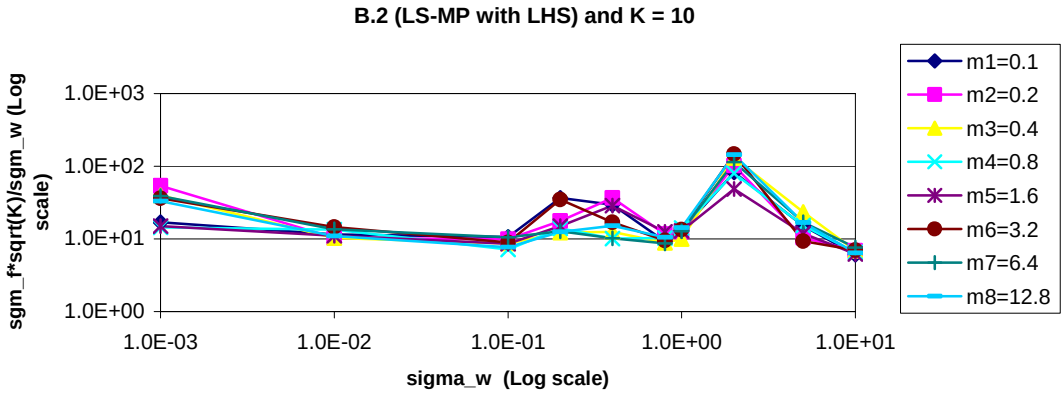


**Figure 24. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$n_p$ with algorithm B.2 (LS-MP with LHS) for $K$ = 10 as a function of $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10}.**

To show the effect of the type of sampling technique SRS or LHS, consider both algorithms B.1 and B.2, i.e. for LS-MP with SRS and LHS respectively, for the same fixed value of $K$ and variation of standard deviation $\sigma_w$, as specified in the table above, that is:

| **Algorithm** | **Values for *K*** | **Values for $\sigma_w$** |
|---|---|---|
| B.1 (LS-MP with SRS; black line) and B.2 (LS-MP with LHS; red line) | $K = 16$ | $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10} |

The following figure presents:

- The normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for $j = 8$, i.e. for parameter $f_8 = 12.8$, as a function of $\sigma_w$ for algorithm B.1 (i.e. with SRS; black line) and algorithm B.2 (i.e. with LHS; red line), and where both the horizontal and vertical axes are presented in a logarithmic scale.

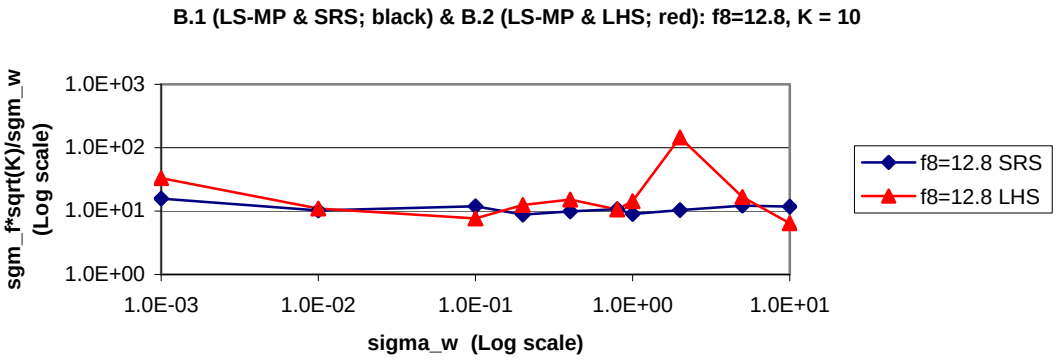


**Figure 25. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 8, i.e. for parameter $f_8$ = 12.8, with algorithms B.1 (LS-MP with SRS) and B.2 (LS-MP with SRS) for *K* = 10 as a function of $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10}.**

### 6.2.5 Discussion of results for LS-MP Algorithms B.1 and B.2

Here, a summarising discussion is given of the results presented in Subsections 6.2.1 through 6.2.4 for LS-MP algorithms B.1 and B.2.

**Mean $\mu(\hat{f}_j)$:**

- For $2 \le K \le n_p$ : The results show that the smaller $K$ is, the more inaccurate the mean $\mu(\hat{f}_j)$ for $j = 1, 2, \ldots, n_p$ becomes when compared to the true values of $f_j$ (See Figure 14 and Figure 17).
- Similarly as was observed for CLS Algorithms A.1 and A.2, the results show that for most values of $K \ge 9$ and $\sigma_w$, the obtained values for the mean $\mu(\hat{f}_j)$ approach the true values of $f_j$ very accurately, though with the exception of $K$ being too close to $N_p$ or of $\sigma_w$ being too large. For example for $K = 9$ in Figure 14 and in Figure 17 the obtained values for the mean $\mu(\hat{f}_j)$ become inaccurate, similarly for $\sigma_w = 5$ and $\sigma_w = 10$ in Figure 20 and Figure 23 respectively). The larger $\sigma_w$ becomes, the more inaccurate the mean $\mu(\hat{f}_j)$ for $j = 1, 2, \ldots, n_p$ becomes as can be observed from Figure 20 (i.e. for $\sigma_w = 5$ and $\sigma_w = 10$) and in Figure 23 (i.e. for $\sigma_w = 2, 5, 10$).

**Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$:**

- The obtained values for $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for the case that $2 \le K \le n_p$ and $\sigma_w$ is small ($\sigma_w = 0.0.1$) are much larger than for the case that $K$ is much larger than $n_p$ (See Figure 15). For example:
  - for $2 \le K \le n_p$, the obtained values for $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ are larger than $10^2$;
  - for $K \ge$ 25, the obtained values for $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ are smaller than 5.

  This effect does not occur for larger values of $\sigma_w$, see Figure 18 for $\sigma_w = 0.5$.
- Similarly as was observed for CLS Algorithms A.1 and A.2, the results show that for $K$ close to $n_p$ (i.e. for $K = 9$ and less often for $K = 10$), the obtained values for the normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ sometimes become relatively large (see Figure 15, Figure 18 and Figure 24) in combination with more inaccuracies in the mean $\mu(\hat{f}_j)$.
- For $K > 9$ (i.e. $K > n_p + 1$), the results show that the larger $K$ becomes the smaller the obtained values for $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for $j = 1, 2, \ldots, n_p$ become. For example in Figure 15 with $\sigma_w = 0.01$, the obtained values for the normalized standard deviations $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for $K = 10$ fall within the range 8-16 and for $K = 500$ fall within the range 2.2-2.8. In Figure 18 with $\sigma_w = 0.5$, the obtained values for

the normalized standard deviations for $K$ = 10 fall within the range 7-15 and for $K$ = 500 fall within the range 2.2-2.8.

- For larger values of $\sigma_w$ (= 0.5), and for $K$ = 9 and $K$ = 10 (i.e. for $K = n_p + 1$ and $K = n_p + 2$) sometimes the obtained values for the mean $\mu(\hat{f}_j)$ have large deviations from the true values of $f_j$ (See Figure 17)**,** and also the obtained values for the normalized standard deviations $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for $j = 1, 2, \ldots, n_p$ turn out to be relatively large (See Figure 18). The Monte Carlo simulations showed that this can happen for both SRS and for LHS. For smaller values of $\sigma_w$ (= 0.01) this did occur less for $K$ = 9 and not for $K$ = 10. These inaccuracies which sometimes occur for $K = n_p + 1$, and less often for $K = n_p + 2$, might be caused by matrix $X_0{}^T X_0$ having a high condition number.
- For $K$ = 16 and $\sigma_w$ = { 0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10 } the values of the normalized standard deviations $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for all $j = 1, 2, \ldots, n_p$, fall within the range 2.8-5.6 (See Figure 21).
- For $K$ = 10 and $\sigma_w$ = { 0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10 } the values of the normalized standard deviations $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for all $j = 1, 2, \ldots, n_p$ in Figure 24 varies between 6 and 147. For $\sigma_w$ = 2, the obtained values for $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ varies between 48 and 147, whereas for $\sigma_w$ = 0.1 and 10, the obtained values for $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ are much smaller as they vary between 6 and 11.

**LHS versus SRS:**

- For all $K$ and for all $\sigma_w$ as considered in the simulations, sometimes SRS yields smaller values of the normalized standard deviations $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$, and sometimes LHS does (See Figure 16, Figure 19, Figure 22 and Figure 25). However no systematic differences can be observed between results obtained with SRS and LHS.

### 6.2.6 Discussion of results for CLS Algorithm A.1 versus LS-MP Algorithm B.1

Comparison of simulation results of Algorithms A.1 (=CLS with SRS) and B.1 (=LS-MP with SRS) shows that:

- For $K > n_p$, the simulation results show that the algorithms with LS-MP gives the same values for the mean $\mu(\hat{f}_j)$ and normalized standard deviations $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ as CLS does. This can be explained as follows: In the Monte Carlo simulations for algorithms A.1 and B.1 (and similarly for algorithms A.2 and B.2) it turned out that matrix $X_0{}^T X_0$ had full rank for each $K > n_p$, which means that matrix $X_0{}^T X_0$ is invertible. If the matrix $X_0{}^T X_0$ is invertible, then the Moore-Penrose inverse and inverse coincide by definition, in which case it is obvious that the results for CLS and LS-MP coincide for $K > n_p$.
- For $K \leq n_p$, the CLS could not be used.

## 6.3 Discussion of results for PLS-N Algorithm C.1 versus LS-MP Algorithm B.1

Results for NIPALS based PLS (PLS-N) with sampling type SRS (i.e. algorithm C.1) are presented in this subsection and are compared with Least Squares with Moore-Penrose inverse (LS-MP) with sampling type SRS (i.e. algorithm B.1).

This subsection is organised as follows:

- In Subsection 6.3.1, we vary $K$ for a fixed value of standard deviation $\sigma_w$.
- In Subsection 6.3.2, the results obtained in Subsection 6.3.1 for PLS-N algorithm C.1 and LS-MP algorithm B.1 are discussed.

### 6.3.1 Variation of *K* and fixed $\sigma_w$

Consider algorithms B.1 and C.1 for a fixed value of the standard deviation $\sigma_w$ and with variation of $K$, as specified in the following table:

| **Algorithm** | **Values for *K*** | **Values for $\sigma_w$** |
|---|---|---|
| B.1 (LS-MP with SRS) | $K$ = {3, 4, 6, 8, 9, 10, 11, 20, 35, 100} | $\sigma_w = 0.5$ |
| C.1 (PLS-N with SRS) | $K$ = {3, 4, 6, 8, 9, 10, 11, 20, 35, 100} | $\sigma_w = 0.5$ |

For both of these algorithms, the following two figures present:

- The mean $\mu(\hat{f}_j)$ as a function of the parameter index $j = 1, 2, \ldots, n_p$ for each of the values of $K$ as specified in the above table, and the true values for $(f_1 \quad f_2 \quad \cdots \quad f_8) = (0.1 \quad 0.2 \quad 0.4 \quad 0.8 \quad 1.6 \quad 3.2 \quad 6.4 \quad 12.8)$.

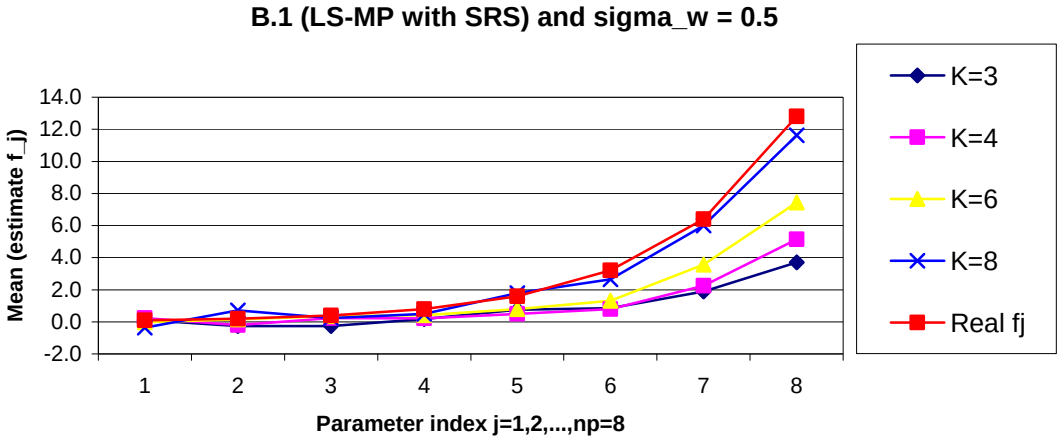


**Figure 26. Mean $\mu(\hat{f}_j)$ values as a function of j = 1,2,...,$n_p$ with algorithm B.1 (LS-MP with SRS) for $\sigma_w$ = 0.5 and $K$ = {3, 4, 6, 8}. The red line indicated as 'Real fj' represents the true values for $f_j$.**

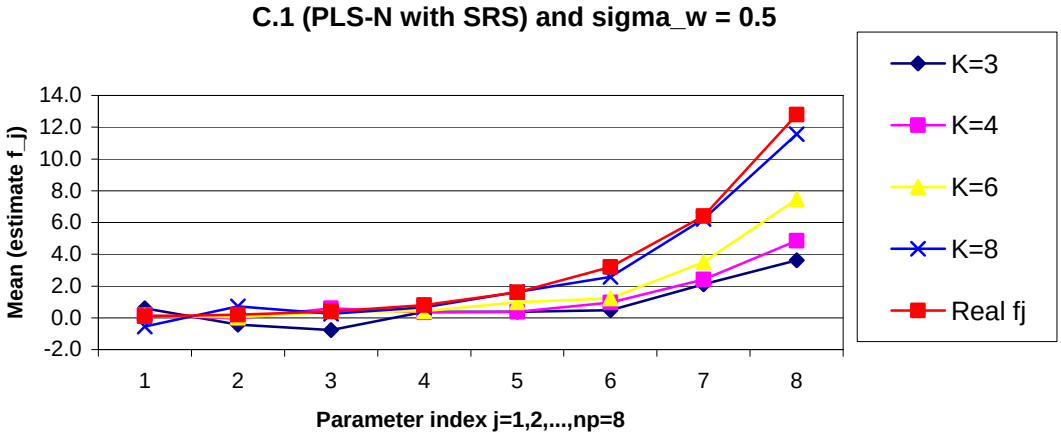


**Figure 27. Mean $\mu(\hat{f}_j)$ values as a function of j = 1,2,...,$n_p$ with algorithm C.1 (PLS-N with SRS) for $\sigma_w$ =0.5 and $K$ = {3, 4, 6, 8}. The red line indicated as 'Real fj' represents the true values for $f_j$.**

To compare the difference between the type of estimation approaches, consider both algorithms B.1 and C.1, i.e. LS-MP and PLS-N respectively and both with SRS, for the same fixed value of the standard deviation $\sigma_w$ and variation of $K$, as specified in the table above, that is:

| Algorithm | Values for *K* | Values for $\sigma_w$ |
|---|---|---|
| B.1 (LS-MP with SRS; black line)<br>C.1 (PLS-N with SRS; red line) | *K* = {3, 4, 6, 8, 9, 10, 11, 20, 35, 100} | $\sigma_w$ = 0.5 |

The following two figures present:

- The normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for $j$ = 5, and j = 8 i.e. for parameters $f_5$ = 1.6 and $f_8$ = 12.8, as a function of $K$ for algorithm B.1 (i.e. LS-MP with SRS; black line) and algorithm C.1 (i.e. PLS-N with SRS; red line), and where both the horizontal and vertical axes are presented in a logarithmic scale.

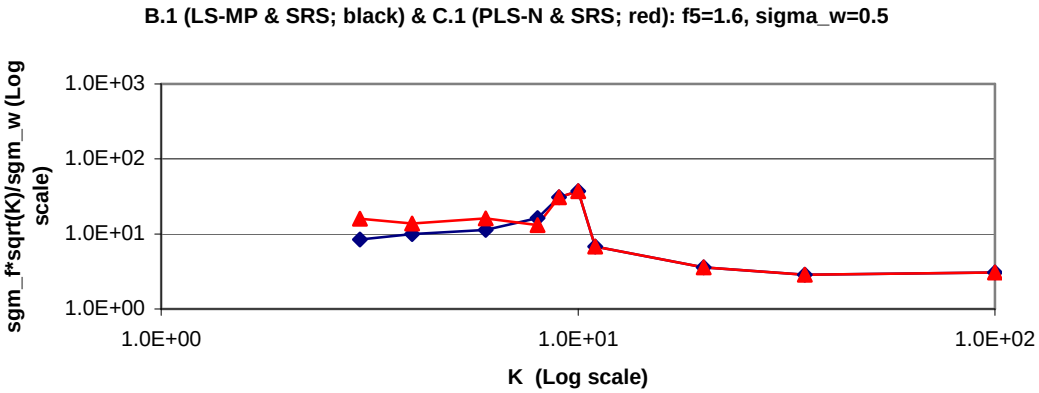


**Figure 28. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 5, i.e. for parameter $f_5$ = 1.6, with algorithms B.1 (LS-MP with SRS) and C.1 (PLS-N with SRS) for $\sigma_w$ =0.5 as a function of *K* = {3, 4, 6, 8, 9, 10, 11, 20, 35, 100}.**

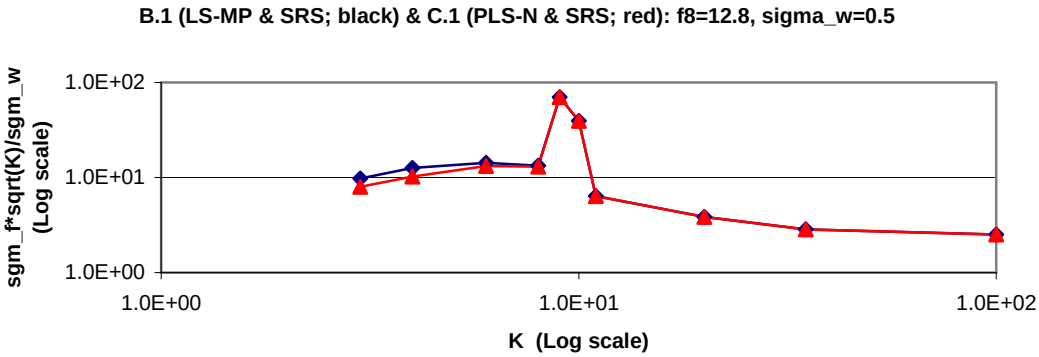


**Figure 29. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 8, i.e. for parameter $f_8$ = 12.8, with algorithms B.1 (LS-MP with SRS) and C.1 (PLS-N with SRS) for $\sigma_w$ = 0.5 as a function of *K* = {3, 4, 6, 8, 9, 10, 11, 20, 35, 100}.**

### 6.3.2 Discussion of results for PLS-N Algorithm C.1 versus LS-MP Algorithm B.1

Here, a summarising discussion is given of the results presented in Subsection 6.3.1 for PLS-N algorithm C.1 and LS-MP algorithm B.1.

**Algorithm C.1 (PLS-N with SRS) versus algorithm B.1 (LS-MP with SRS)**
The simulation results obtained show that:

- For $K < n_p$, sometimes algorithm B.1 yields smaller values of the normalized standard deviations $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$, and sometimes algorithm C.1 does. See for example Figure 28 which shows that algorithm B.1 (LS-MP with SRS) yields smaller values for $\sigma(\hat{f}_5)\sqrt{K}/\sigma_w$, whereas in Figure 29 algorithm C.1 (PLS-N with SRS) yields smaller values for $\sigma(\hat{f}_8)\sqrt{K}/\sigma_w$. The results also show that for $K < n_p$, algorithms B.1 and C.1 gives different numerical values for the mean $\mu(\hat{f}_j)$ (see Figure 26 and Figure 27).
- For $K \geq n_p$, LS-MP and PLS-N give the same numerical results for the normalized standard deviations $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ (see Figure 28 and Figure 29). Similarly for $K \geq n_p$, algorithms B.1 and C.1 gives the same numerical values for the mean $\mu(\hat{f}_j)$.

## *6.4 Comparison of PLS-S Algorithm D.1 versus LS-MP Algorithm B.1*

For the comparison of Algorithm D.1 (PLS-S with SRS) versus Algorithm B.1 (LS-MP with SRS) only the main findings are summarized.

The simulation results obtained (though not included in this report) have shown that:

- For $K < n_p$, algorithm B.1 (LS-MP with SRS) and algorithm D.1 (PLS-S with SRS) give the same numerical values for both the mean $\mu(\hat{f}_j)$ and the normalized standard deviations $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ (numerical differences are in the order $10^{-9}$ or smaller).
- Similarly, for $K \geq n_p$, algorithm B.1 (LS-MP with SRS) and algorithm D.1 (PLS-S with SRS) give the same numerical values for both the mean $\mu(\hat{f}_j)$ and the normalized standard deviations $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ (numerical differences are in the order $10^{-9}$ or smaller).

### 6.5 Discussion of results obtained

Classical Least Squares (CLS) estimation cannot be applied if the number of samples $K$ is smaller or equal than the number of regression components $n_p$. For the CLS algorithm with $K > n_p$, the Monte Carlo simulations show that:

- For $K$ close to $n_p$ (in particular for $K = n_p + 1$ and $K = n_p + 2$), the CLS results become inaccurate, i.e. the obtained values for the mean $\mu(\hat{f}_j)$ become inaccurate, and the obtained values for the normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ can become relativey large. This effect is even stronger when in combination with large values of $\sigma_w$. These inaccuracies are caused by an ill-conditioned matrix $X_0^T X_0$.
- For $K > n_p + 2$, the CLS results show that the mean $\mu(\hat{f}_j)$ for $j = 1, 2, \ldots, n_p$ approach the true values of $f_j$ very accurately and the larger $K$ is, the smaller the normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for $j = 1, 2, \ldots, n_p$ becomes. So for this case the CLS algorithm is suitable to determine estimator $\hat{F}^T = (\hat{f}_1, \hat{f}_2, \cdots, \hat{f}_{n_p})$.
- For all $K \geq n_p$, no systematic differences can be observed between results obtained with SRS and LHS, as sometimes SRS yields smaller values of the normalized standard deviations $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$, and sometimes LHS does.

Alternative approaches LS-MP, PLS-N and PLS-S have been applied, and the results show that for each value of $K > n_p$, algorithms with LS-MP, PLS-N and PLS-S give numerically the same results as CLS does. For LS-MP this also follows from theory, since in case the inverse of matrix $X_0^T X_0$ exists, then the Moore-Penrose inverse and inverse coincide by definition. And for $K \leq n_p$ the results show that

- In contrast to CLS, algorithms with LS-MP, PLS-N and PLS-S can also be applied for $K \leq n_p$, and the smaller $K$ is, the more the mean $\mu(\hat{f}_j)$ deviates from real value of $f_j$ for $j = 1, 2, \ldots, n_p$.
- Both LS-MP and PLS-S algorithms give the same numerical values for the mean $\mu(\hat{f}_j)$ and normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$, for $K \leq n_p$.
- The PLS-N algorithm gives different values for the mean $\mu(\hat{f}_j)$ and normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for $K \leq n_p$ when compared to LS-MP and PLS-S. No systematic differences can be observed between the results of PLS-N when compared to LS-MP and PLS-S, as sometimes PLS-N yields smaller values of the normalized standard deviations $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$, and sometimes LS-MP and PLS-S do.

Regarding sampling types SRS and LHS the results show that

- For all values of *K*, no systematic differences can be observed between results obtained with SRS and LHS, as sometimes SRS yields smaller values of the normalized standard deviations $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$, and sometimes LHS does.

Solely based on the numerical results for LS-MP, PLS-N and PLS-S, no specific conclusion can be drawn which of the three approaches is preferred. Another aspect that can be considered is the calculation speed of the three approaches:

- The Moore-Penrose inverse is time-consuming for large matrices [Courrieu, 2005]; see also Subsection 3.4 in this report.
- The SIMPLS approach directly finds weight vectors which are applied to the original matrix *X*, and without explicit computation of matrix inverses.
- SIMPLS is also faster than PLS-N ([Alin, 2009]; see also Subsection 3.5 in this report.
- The PLS-S algorithm is the fastest when compared to LS-MP [Courrieu, 2005] and PLS-N [De Jong, 1993], [Alin, 2009].

Thus PLS-S is computationally the best. PLS-S with SRS (i.e. Algorithm D.1) will be considered in more detail in Section 7 with the aim to consider the effect of the values for the standard deviation $\sigma_w$ and the effect of *K* on the mean $\mu(\hat{f}_j)$ and the normalized standard deviations $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$.

# 7 Effect of *K* and $\sigma_w$

In this section, the effect of standard deviation $\sigma_w$ and $K$, on the normalized standard deviations $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for $j=1,2,\ldots,n_p$, will be analysed for PLS-S with sampling type SRS (i.e. Algorithm D.1). In this section the following is discussed:

- Results for PLS-S with SRS algorithm D.1 with variation of $K$ in Subsection 7.1.
- Results for PLS-S with SRS algorithm D.1 with variation of standard deviation $\sigma_w$ in Subsection 7.2.
- A summarising discussion about the effect of standard deviation $\sigma_w$ and number of samples *K* on PLS-S algorithm D.1 is given in Subsection 7.3.

## *7.1 PLS-S Algorithm D.1 and variation of K*

Results for algorithm D.1, i.e. PLS-S with SRS, for fixed value of the standard deviation $\sigma_w$ and with variation of $K$, are presented in the three subsections hereafter for low, high and very high values for $\sigma_w$ (0.01, 0.5 and 10) respectively.

This subsection is organised as follows:

- In Subsection 7.1.1, we vary $K$ for a fixed low value of standard deviation $\sigma_w$.
- In Subsection 7.1.2, we vary $K$ for a fixed high value of standard deviation $\sigma_w$.
- In Subsection 7.1.3, we vary $K$ for a fixed very high value of standard deviation $\sigma_w$.

### 7.1.1 Variation of *K* and fixed low value of $\sigma_w$

Consider algorithm D.1, i.e. PLS-S with SRS, for a fixed value of the standard deviation $\sigma_w$ and with variation of $K$, as specified in the following table:

| **Algorithm** | **Values for *K*** | **Values for $\sigma_w$** |
|---|---|---|
| D.1 (PLS-S with SRS) | *K* = {4, 6, 8, 9, 10, 25, 50, 100, 500, 1000} | $\sigma_w = 0.01$ |
| D.1 (PLS-S with SRS) | *K* = {2, 4, 6, 8, 9, 10, 11, 20, 35, 100} | $\sigma_w = 0.01$ |
| D.1 (PLS-S with SRS) | *K* = {100, 300, 600, 800, 1000, 3000, 5000, 10000, 30000, 50000} | $\sigma_w = 0.01$ |

The following three figures present:

- The normalized standard deviation of $\sigma(\hat{f}_j)$, i.e. $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$, for each of the parameter indices $j=1,2,\ldots,n_p$ as a function of $K$ where both the horizontal and

vertical axes are presented in a logarithmic scale, and with assumed values for $(f_1 \quad f_2 \quad \cdots \quad f_8) = (0.1 \quad 0.2 \quad 0.4 \quad 0.8 \quad 1.6 \quad 3.2 \quad 6.4 \quad 12.8)$.

The three figures present the results for three different scales of the horizontal axis, representing the axis for the $K$ values.

For $\sigma_w$ = 0.01 and $K$ = {4, 6, 8, 9, 10, 25, 50, 100, 500, 1000}:

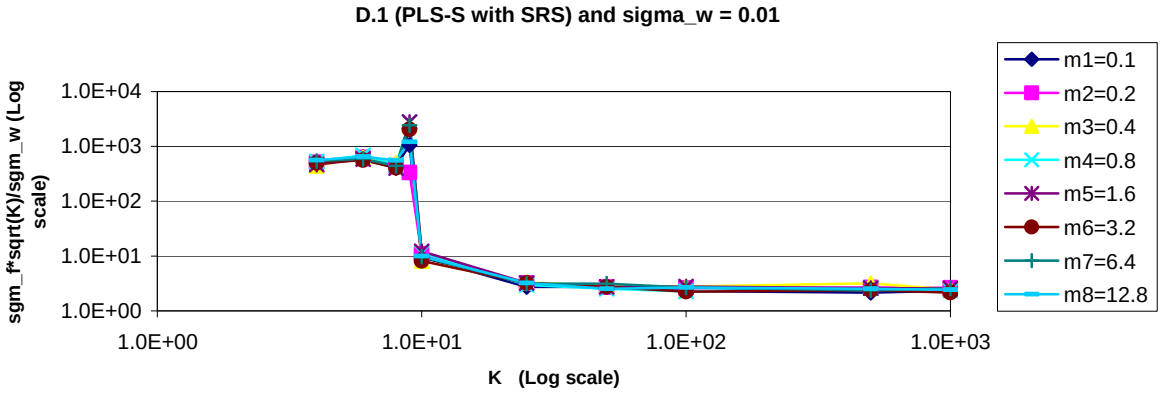


**Figure 30. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$n_p$ with algorithm D.1 (PLS-S with SRS) for $\sigma_w$ = 0.01 and $K$ = {4, 6, 8, 9, 10, 25, 50, 100, 500, 1000}.**

Similar, though on a smaller horizontal axis, i.e. $\sigma_w$ = 0.01 and $K$ = {2, 4, 6, 8, 9, 10, 11, 20, 35, 100}:

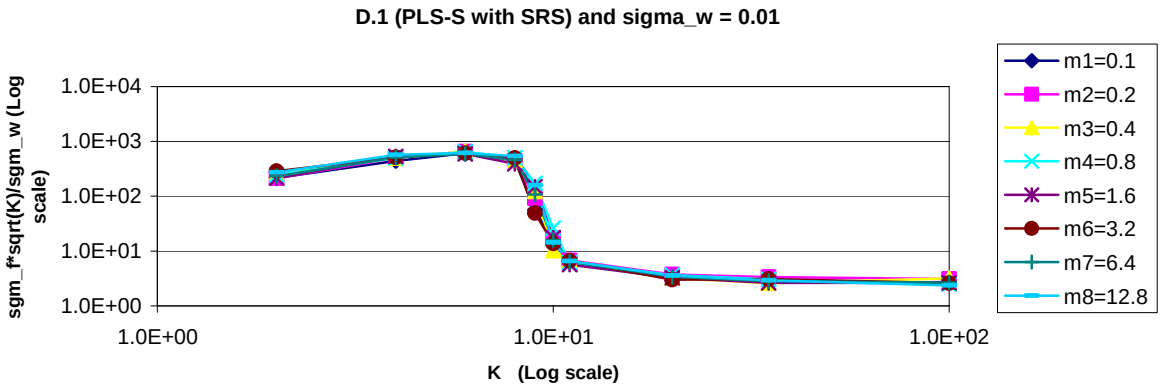


**Figure 31. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$n_p$ with algorithm D.1 (PLS-S with SRS) for $\sigma_w$ = 0.01 and $K$ = {2, 4, 6, 8, 9, 10, 11, 20, 35, 100}.**

Similar, though on a larger horizontal axis, i.e. for $\sigma_w$ = 0.01 and $K$ = {100, 300, 600, 800, 1000, 3000, 5000, 10000, 30000, 50000}:

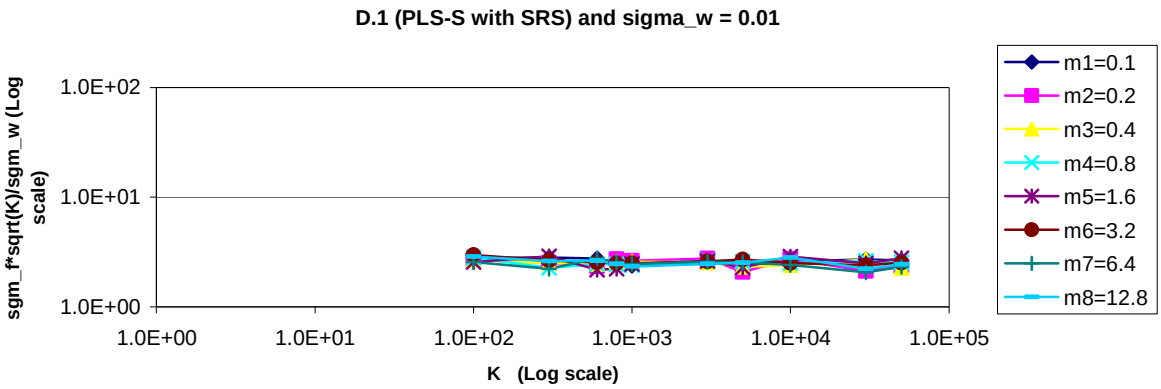


**Figure 32. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$n_p$ with algorithm D.1 (PLS-S with SRS) for $\sigma_w$ = 0.01 and $K$ = {100, 300, 600, 800, 1000, 3000, 5000, 10000, 30000, 50000}.**

**Discussion of results for small value $\sigma_w$ = 0.01**

The simulations for algorithm D.1 (PLS-S with SRS) with $\sigma_w$ = 0.01 show that:

- For $2 \leq K \leq 8$ (i.e. $K \leq n_p$), the values of $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ are within the range of 180-780.
- For $K$ = 9 (i.e. $K = n_p+1$), the values of $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ variy from from 23 to very high (in the figure above the maximum is around $1.8 \cdot 10^3$, though other simulations show that it can even be around $3.2 \cdot 10^4$).
- For $K > 9$ (i.e. $K > n_p+1$), the values of $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ do not really converge, although the range in which they fall is small, i.e. all fall within the range 2.2-3.1.

### 7.1.2 Variation of *K* and fixed high value of $\sigma_w$

Consider algorithm D.1, i.e. PLS-S with SRS, for a fixed value of the standard deviation $\sigma_w$ and with variation of $K$, as specified in the following table:

| Algorithm | Values for *K* | Values for $\sigma_w$ |
|---|---|---|
| D.1 (PLS-S with SRS) | *K* = {4, 6, 8, 9, 10, 25, 50, 100, 500, 1000} | $\sigma_w = 0.5$ |
| D.1 (PLS-S with SRS) | *K* = {2, 4, 6, 8, 9, 10, 11, 20, 35, 100} | $\sigma_w = 0.5$ |
| D.1 (PLS-S with SRS) | *K* = {100, 300, 600, 800, 1000, 3000, 5000, 10000, 30000, 50000} | $\sigma_w = 0.5$ |

The following three figures present:

- The normalized standard deviation of $\sigma(\hat{f}_j)$, i.e. $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$, for each of the parameter indices $j = 1, 2, \ldots, n_p$ as a function of $K$ where both the horizontal and vertical axes are presented in a logarithmic scale, and with assumed values for $(f_1 \quad f_2 \quad \cdots \quad f_8) = (0.1 \quad 0.2 \quad 0.4 \quad 0.8 \quad 1.6 \quad 3.2 \quad 6.4 \quad 12.8)$.

  The three figures present the results for three different scales of the horizontal axis, representing the axis for the $K$ values.

For $\sigma_w$ = 0.5 and *K* = {4, 6, 8, 9, 10, 25, 50, 100, 500, 1000}:

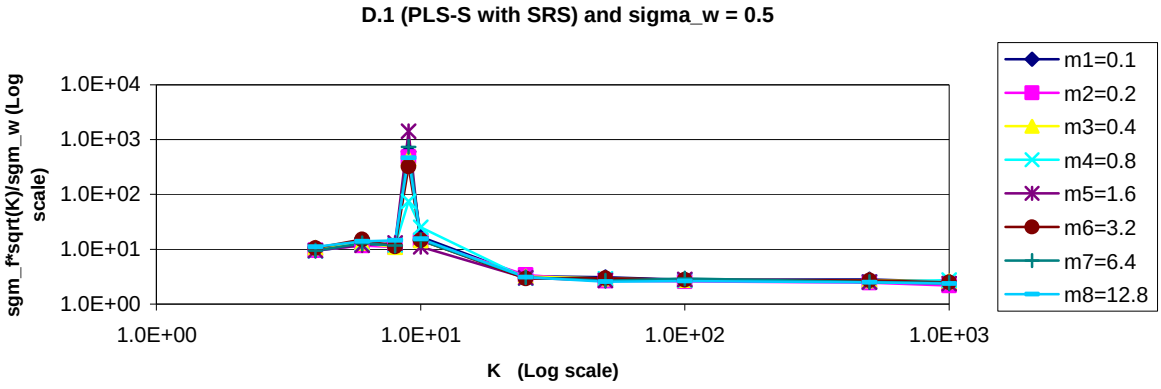


**Figure 33. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$n_p$ with algorithm D.1 (PLS-S with SRS) for $\sigma_w$ = 0.5 and *K* = {4, 6, 8, 9, 10, 25, 50, 100, 500, 1000}.**

Similar, though on a smaller horizontal axis, i.e. for $\sigma_w$ = 0.5 and $K$ = {2, 4, 6, 8, 9, 10, 11, 20, 35, 100}:

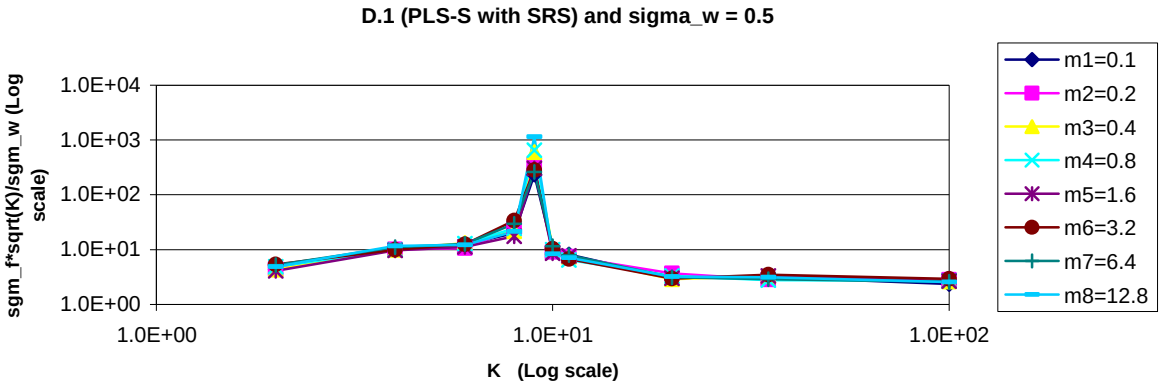


**Figure 34. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$N_p$ with algorithm D.1 (PLS-S with SRS) for $\sigma_w$ = 0.5 and $K$ = {2, 4, 6, 8, 9, 10, 11, 20, 35, 100}.**

Similar, though on a larger horizontal axis, i.e. for $\sigma_w$ = 0.5 and $K$ = {100, 300, 600, 800, 1000, 3000, 5000, 10000, 30000, 50000}:

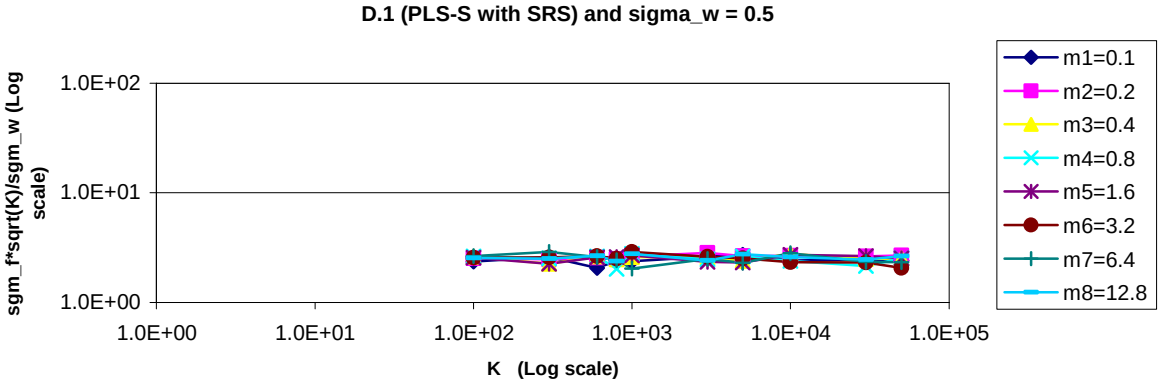


**Figure 35. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$n_p$ with algorithm D.1 (PLS-S with SRS) for $\sigma_w$ = 0.5 and $K$ = {100, 300, 600, 800, 1000, 3000, 5000, 10000, 30000, 50000}.**

**Discussion of results for high value $\sigma_w$ = 0.5**

The simulations for algorithm D.1 (PLS-S with SRS) with $\sigma_w$ = 0.5 show that:

- For $2 \leq K \leq 8$ (i.e. $K \leq n_p$), the values of $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ are within the range of 4-33.
- For $K = 9$ (i.e. $K = n_p+1$), the values of $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ varies from from 17 to very high (in the figure above the maximum is around $1.8 \cdot 10^3$).
- For $K > 9$ (i.e. $K > n_p+1$), the values of $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ do not really converge to some fixed number, but instead converge to a range and within this range they can vary, the range is 2.0-2.9.

### 7.1.3 Variation of *K* and fixed very high value of $\sigma_w$

Consider algorithm D.1, i.e. PLS-S with SRS, for a fixed value of the standard deviation $\sigma_w$ and with variation of $K$, as specified in the following table:

| Algorithm | Values for *K* | Values for $\sigma_w$ |
|---|---|---|
| D.1 (PLS-S with SRS) | $K$ = {4, 6, 8, 9, 10, 25, 50, 100, 500, 1000} | $\sigma_w = 10$ |
| D.1 (PLS-S with SRS) | $K$ = {2, 4, 6, 8, 9, 10, 11, 20, 35, 100} | $\sigma_w = 10$ |
| D.1 (PLS-S with SRS) | $K$ = {100, 300, 600, 800, 1000, 3000, 5000, 10000, 30000, 50000} | $\sigma_w = 10$ |

The following three figures present:

- The normalized standard deviation of $\sigma(\hat{f}_j)$, i.e. $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$, for each of the parameter indices $j = 1, 2, \ldots, n_p$ as a function of $K$ where both the horizontal and vertical axes are presented in a logarithmic scale, and with assumed values for $(f_1 \quad f_2 \quad \cdots \quad f_8) = (0.1 \quad 0.2 \quad 0.4 \quad 0.8 \quad 1.6 \quad 3.2 \quad 6.4 \quad 12.8)$.

  The three figures present the results for three different scales of the horizontal axis, representing the axis for the $K$ values.

For $\sigma_w$ = 10 and $K$ = {4, 6, 8, 9, 10, 25, 50, 100, 500, 1000}:

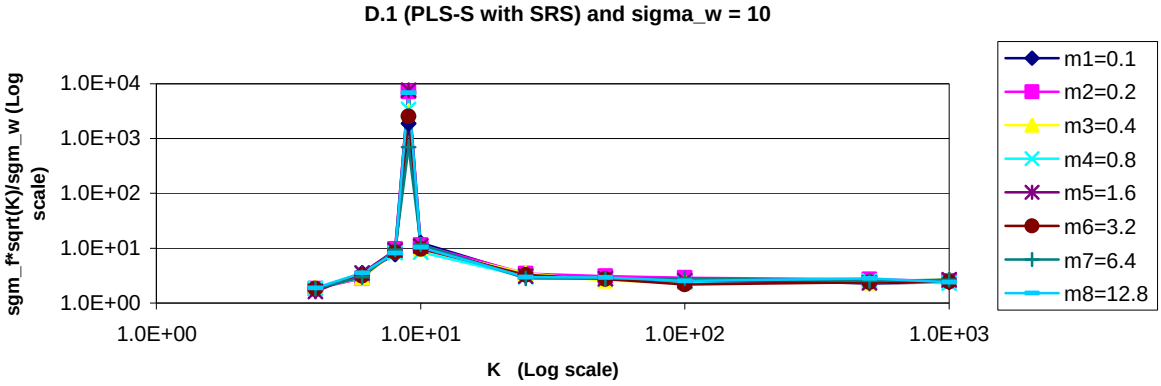


**Figure 36. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$n_p$ with algorithm D.1 (PLS-S with SRS) for $\sigma_w$ = 10 and *K* = {4, 6, 8, 9, 10, 25, 50, 100, 500, 1000}.**

Similar, though on a smaller horizontal axis, i.e. for $\sigma_w$ = 10 and $K$ = {2, 4, 6, 8, 9, 10, 11, 20, 35, 100}:

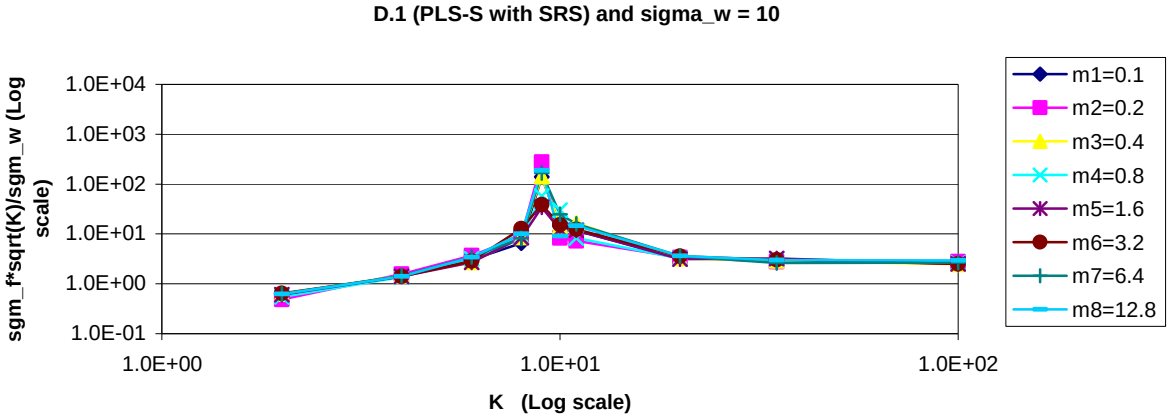


**Figure 37. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$n_p$ with algorithm D.1 (PLS-S with SRS) for $\sigma_w$ = 10 and $K$ = {2, 4, 6, 8, 9, 10, 11, 35, 100}.**

Similar, though on a larger horizontal axis, i.e. for $\sigma_w$ = 10 and $K$ = {100, 300, 600, 800, 1000, 3000, 5000, 10000, 30000, 50000}:

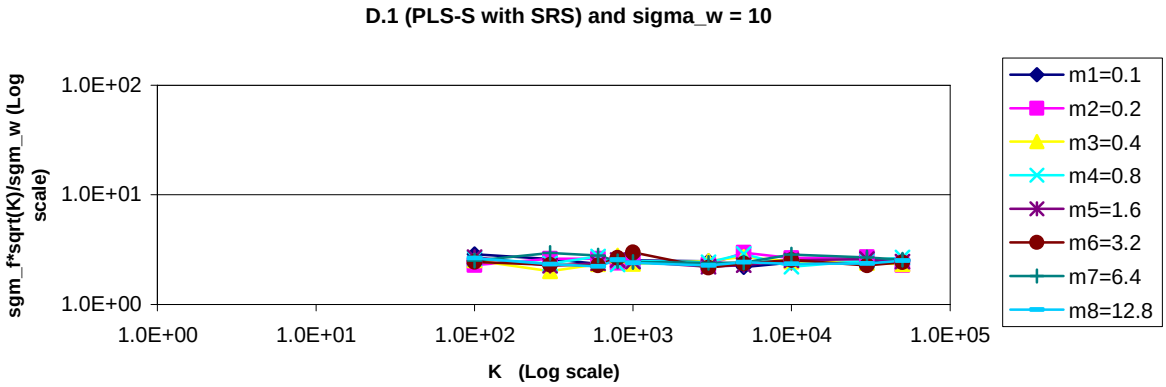


**Figure 38. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$n_p$ with algorithm D.1 (PLS-S with SRS) for $\sigma_w$ = 10 and $K$ = {100, 300, 600, 800, 1000, 3000, 5000, 10000, 30000, 50000}.**

**Discussion of results for very high value $\sigma_w$ = 10**

The simulations for algorithm D.1 (PLS-S with SRS) with $\sigma_w$ = 20 show that:

- For $2 \leq K \leq 8$ (i.e. $K \leq n_p$), the values of $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ are within the range of 0.4-20.
- For $K = 9$ (i.e. $K = n_p+1$), the values of $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ varies from from 17 to very high (in the figure above the maximum is around $7.7 \cdot 10^3$).
- For $K > 9$ (i.e. $K > n_p+1$), the values of $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ do not really converge to some fixed number, but instead converge to a range and within this range they can vary, the range is 2.0-3.0.

## 7.2 PLS-S Algorithm D.1 and variation of $\sigma_w$

Results for algorithm D.1, i.e. PLS-S with SRS, for fixed value of $K$ and with variation of the standard deviation $\sigma_w$, are presented in the two subsections hereafter for $K \geq n_p + 1$ and for $K < n_p$ respectively.

This subsection is organised as follows:

- In Subsection 7.2.1, we vary standard deviation $\sigma_w$ for a fixed $K$.
- In Subsection 7.2.2, we vary standard deviation $\sigma_w$ for a fixed low value of $K$.

For an overall discussion about the effects of both the standard deviation $\sigma_w$ and the number of samples *K* on PLS-S algorithm D.1 the reader is referred to Subsection 7.3.

### 7.2.1 Variation of $\sigma_w$ and fixed value $K \geq n_p+1$

Consider algorithm D.1, i.e. PLS-S with SRS, for a fixed value of *K* and with variation of the standard deviation $\sigma_w$, as specified in the following table:

| **Algorithm** | **Values for *K*** | **Values for $\sigma_w$** |
|---|---|---|
| D.1 (PLS-S with SRS) | $K$ = 9 | $\sigma_w$ = { 0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 } |
| D.1 (PLS-S with SRS) | $K$ = 10 | $\sigma_w$ = { 0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 } |
| D.1 (PLS-S with SRS) | $K$ = 11 | $\sigma_w$ = { 0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 } |
| D.1 (PLS-S with SRS) | $K$ = 12 | $\sigma_w$ = { 0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 } |
| D.1 (PLS-S with SRS) | $K$ = 32 | $\sigma_w$ = { 0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 } |
| D.1 (PLS-S with SRS) | $K$ = 1000 | $\sigma_w$ = { 0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 } |

The following figures present:

- The normalized standard deviation of $\sigma(\hat{f}_j)$, i.e. $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$, for fixed values of $K \geq n_p+1$ as specified in the above table for each of the parameter indices $j = 1, 2, \ldots, n_p$ as a function of $\sigma_w$ where both the horizontal and vertical axes are presented in a logarithmic scale, and with assumed values for $(f_1 \quad f_2 \quad \cdots \quad f_8) = (0.1 \quad 0.2 \quad 0.4 \quad 0.8 \quad 1.6 \quad 3.2 \quad 6.4 \quad 12.8)$.

- For $K = 32$ two additional figures are presented, i.e. the corresponding standard deviation $\sigma(\hat{f}_j)$ for each $j = 1, 2, \ldots, n_p$ as a function of $\sigma_w$ where both the horizontal and vertical axes are presented in a logarithmic scale; and the corresponding mean $\mu(\hat{f}_j)$ as a function of the parameter number $j = 1, 2, \ldots, n_p$.

For $K$ = 9 and $\sigma_w$ = { 0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 } the values of $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ can vary from low to very high (in the figure this is from $2 \cdot 10^1$ to $2 \cdot 10^3$):

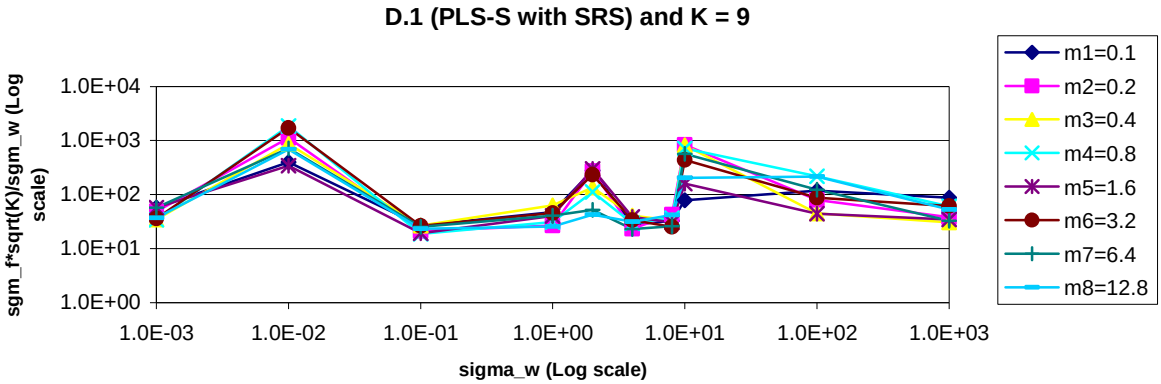


**Figure 39. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$n_p$ with algorithm D.1 (PLS-S with SRS) for $K$ = 9 and $\sigma_w$ = {0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 }.**

For $K$ = 10 and $\sigma_w$ = { 0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 } the values of $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ vary from 8.4 to 35.7 in the figure below:

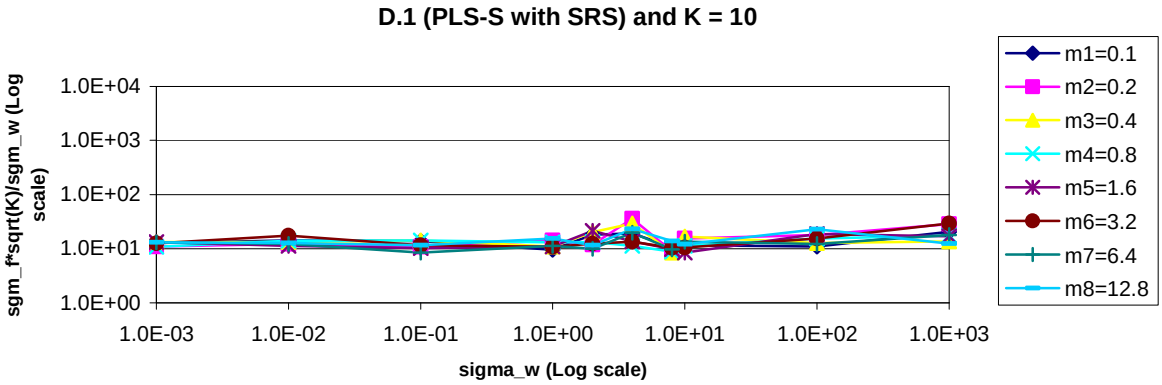


**Figure 40. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$n_p$ with algorithm D.1 (PLS-S with SRS) for $K$ = 10 and $\sigma_w$ = {0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 }.**

For $K$ = 11 and $\sigma_w$ = { 0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 } the values of $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ vary between 5-19:

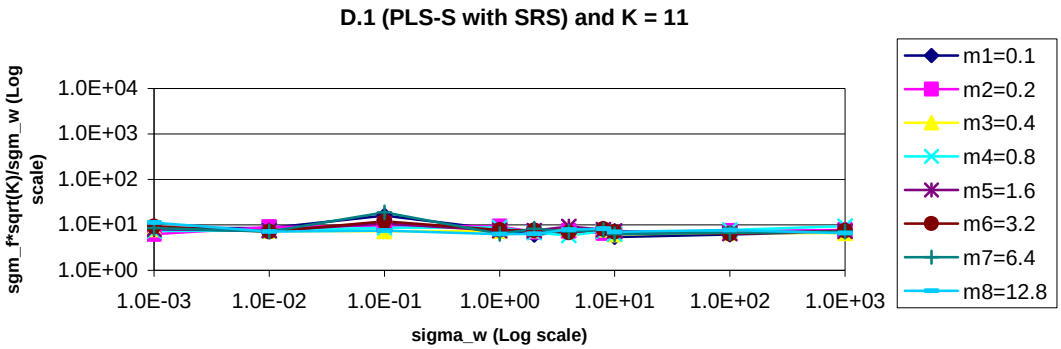


**Figure 41. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$n_p$ with algorithm D.1 (PLS-S with SRS) for $K$ = 11 and $\sigma_w$ = {0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 }.**

For $K$ = 12 and $\sigma_w$ = { 0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 } the values of $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ vary between 4.1-11.5:

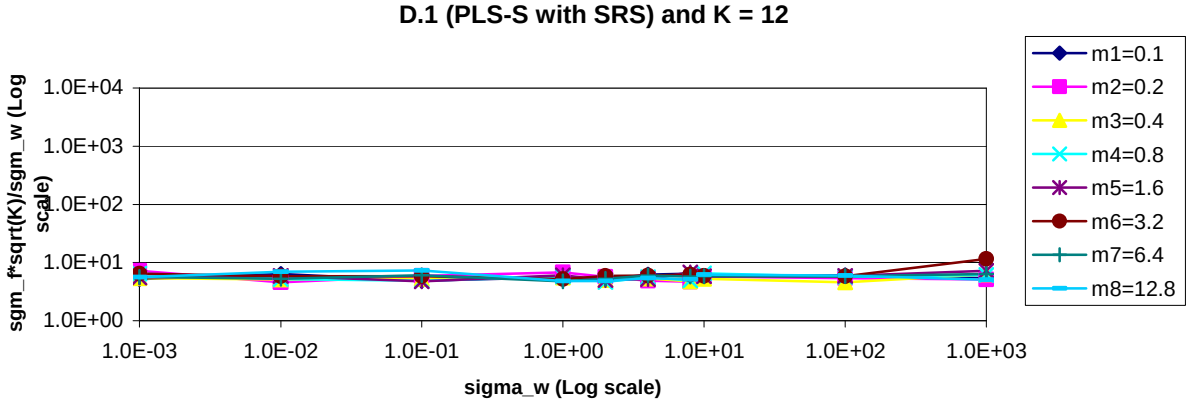


**Figure 42. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$n_p$ with algorithm D.1 (PLS-S with SRS) for $K$ = 12 and $\sigma_w$ = {0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 }.**

For $K$ = 32 and $\sigma_w$ = { 0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 } the values of $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ vary between 2.4-3.6:

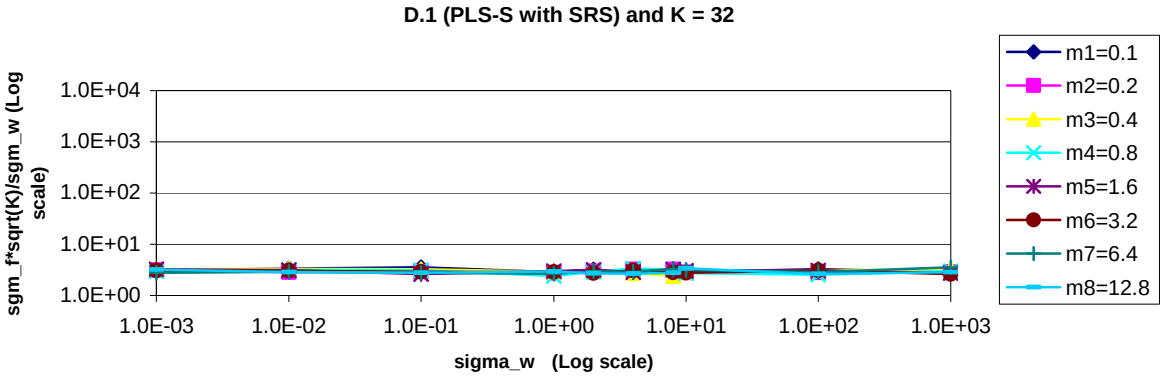


**Figure 43. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$n_p$ with algorithm D.1 (PLS-S with SRS) for $K$ = 32 and $\sigma_w$ = {0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 }.**

and corresponding standard deviation $\sigma(\hat{f}_j)$ is

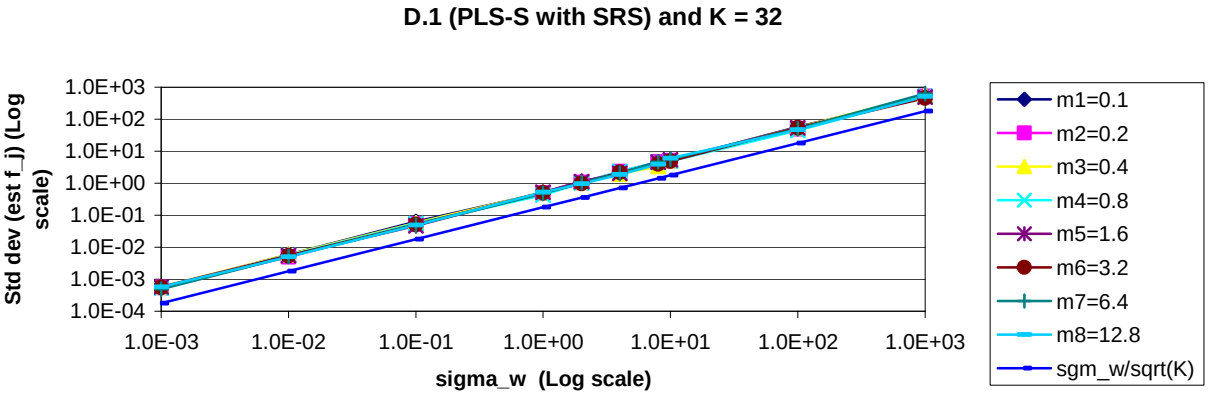


**Figure 44. Standard deviation $\sigma(\hat{f}_j)$ for j = 1,2,...,$n_p$ with algorithm D.1 (PLS-S with SRS) for $K$ = 32 and $\sigma_w$ = { 0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 }.**

and corresponding mean $\mu(\hat{f}_j)$ is

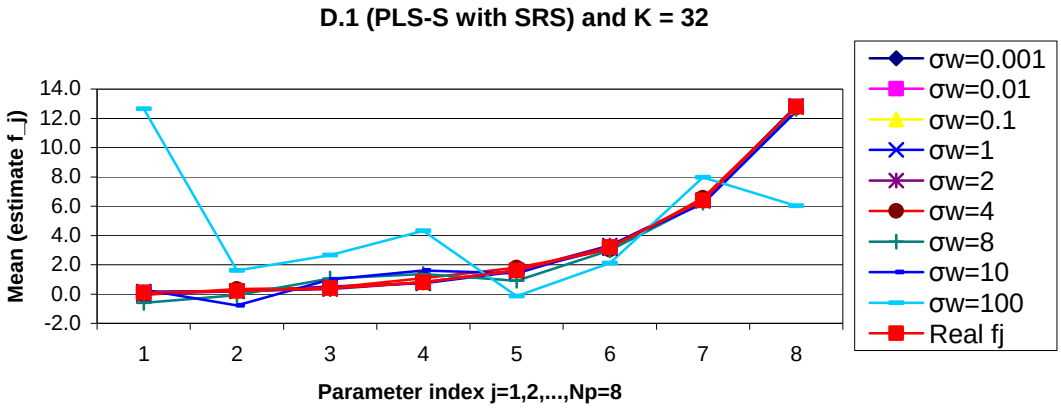


**Figure 45. Mean $\mu(\hat{f}_j)$ values as a function of j = 1,2,...,$n_p$ with algorithm D.1 (PLS-S with SRS) for $K$ = 32 and $\sigma_w$ = {0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100}. The red line indicated as 'Real fj' represents the true values for $f_j$.**

Note that in Figure 45 the mean $\mu(\hat{f}_j)$ values for $\sigma_w$ = 1000 are not shown since they are very inaccurate and the figure would become unreadable.

For $K$ = 1000 and $\sigma_w$ = { 0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 } the values of $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ vary between 2.1 -2.9:

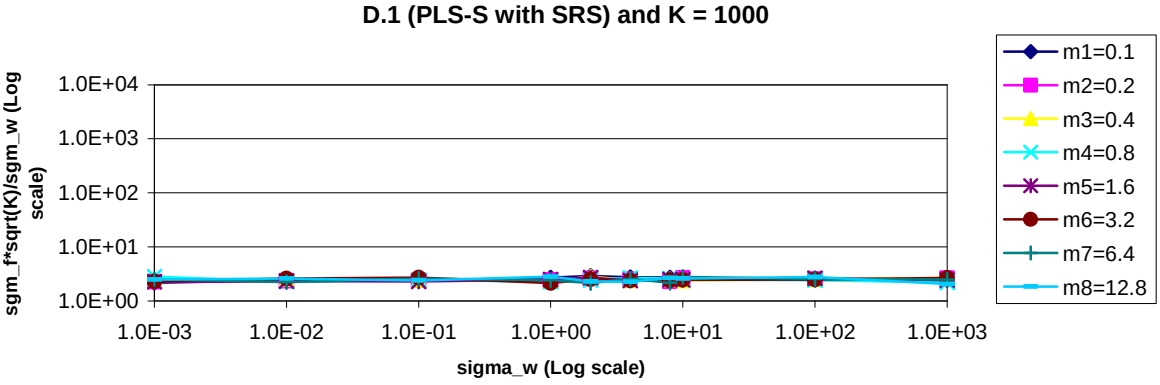


**Figure 46. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$n_p$ with algorithm D.1 (PLS-S with SRS) for $K$ = 1000 and $\sigma_w$ = {0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 }.**

### 7.2.2 Variation of $\sigma_w$ and low value $K = 6$ ($< n_p$)

Consider algorithm D.1, i.e. PLS-S with SRS, for a fixed value of $K$ and with variation of the standard deviation $\sigma_w$, as specified in the following table:

| Algorithm | Values for $K$ | Values for $\sigma_w$ |
|---|---|---|
| D.1 (PLS-S with SRS) | $K = 6$ | $\sigma_w$ = { 0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 } |
| D.1 (PLS-S with SRS) | $K = 6$ | $\sigma_w$ = { 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100 } |
| D.1 (PLS-S with SRS) | $K = 6$ | $\sigma_w$ = { 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1 } |

For each of these three cases, two figures are presented below:

- The normalized standard deviation of $\sigma(\hat{f}_j)$, i.e. $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$, for $K = 6$ for each of the parameter indices $j = 1, 2, \ldots, n_p$ as a function of $\sigma_w$ where both the horizontal and vertical axes are presented in a logarithmic scale, and with assumed values for
  $(f_1 \quad f_2 \quad \cdots \quad f_8) = (0.1 \quad 0.2 \quad 0.4 \quad 0.8 \quad 1.6 \quad 3.2 \quad 6.4 \quad 12.8)$.
- The corresponding standard deviation $\sigma(\hat{f}_j)$ for each $j = 1, 2, \ldots, n_p$ as a function of $\sigma_w$ where both the horizontal and vertical axes are presented in a logarithmic scale.

For $K = 6$ and $\sigma_w$ = { 0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 }:

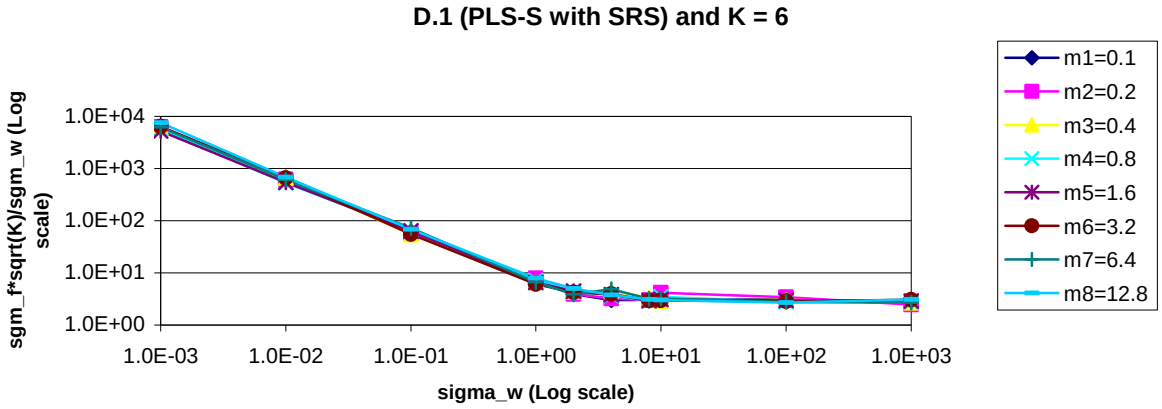


**Figure 47. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$n_p$ with algorithm D.1 (PLS-S with SRS) for $K$ = 6 and $\sigma_w$ = { 0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 }.**

and corresponding figure for standard deviations of $\sigma(\hat{f}_j)$ is

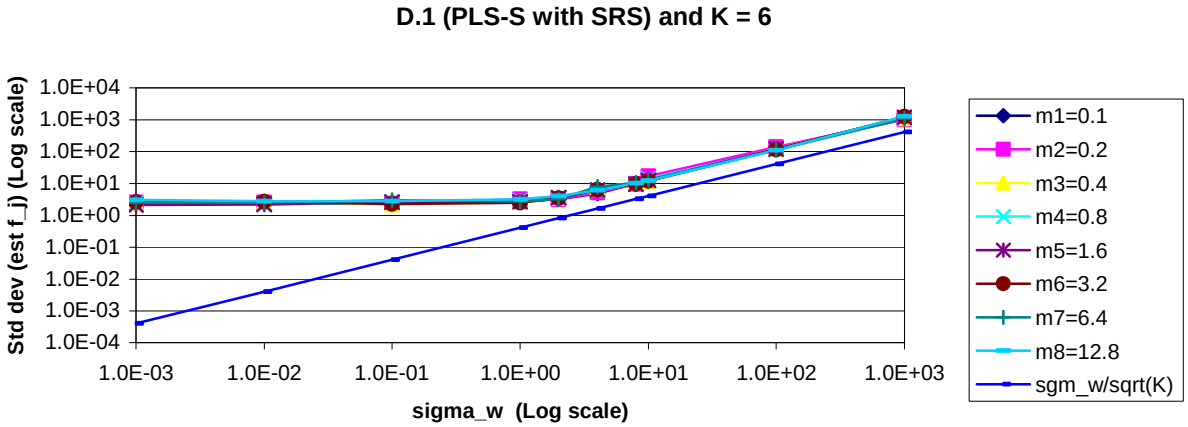


**Figure 48. Standard deviation $\sigma(\hat{f}_j)$ for j = 1,2,...,$n_p$ with algorithm D.1 (PLS-S with SRS) for $K$ = 6 and $\sigma_w$ = { 0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 }.**

Similar, though on a smaller horizontal axis, i.e. for $K$ = 6 and $\sigma_w$ = { 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100 }:

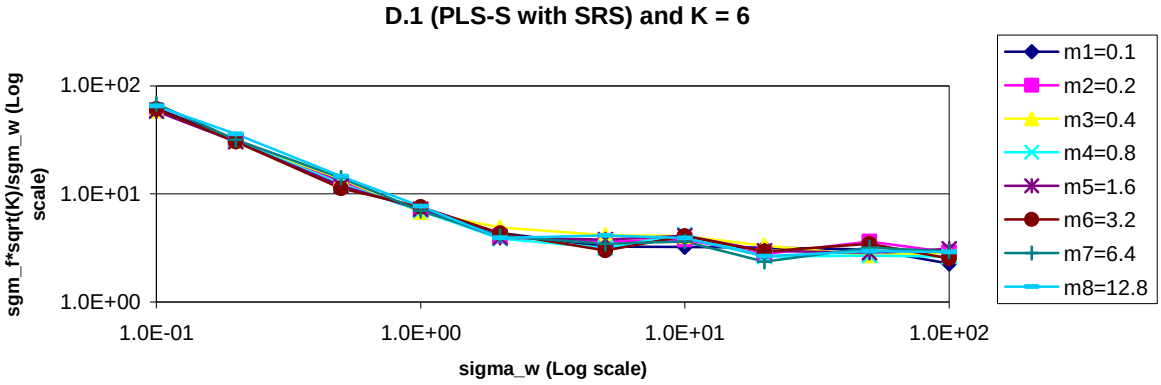


**Figure 49. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$n_p$ with algorithm D.1 (PLS-S with SRS) for $K$ = 6 and $\sigma_w$ = { 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100 }.**

and corresponding figure for standard deviations of $\sigma(\hat{f}_j)$ is

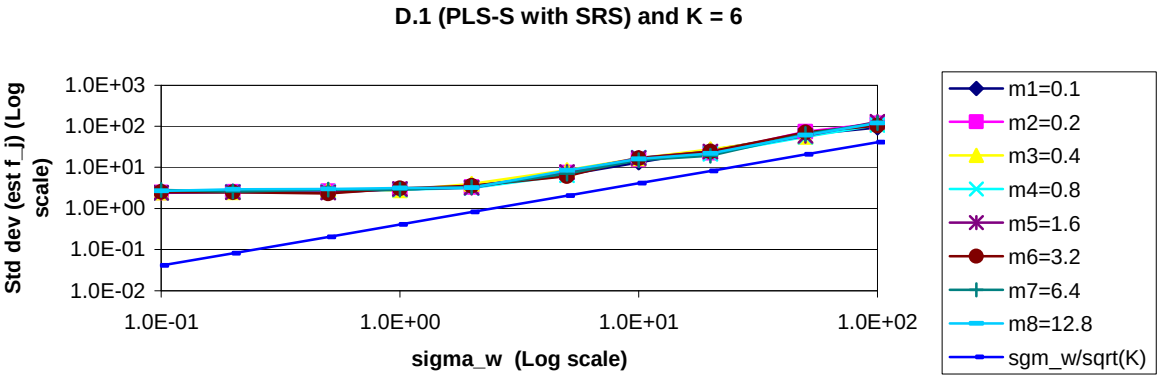


**Figure 50. Standard deviation $\sigma(\hat{f}_j)$ for j = 1,2,...,$N_p$ with algorithm D.1 (PLS-S with SRS) for $K$ = 6 and $\sigma_w$ = { 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100 }.**

Similar, though for $\sigma_w \leq 1$, i.e. for $\sigma_w$ = { 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1 }

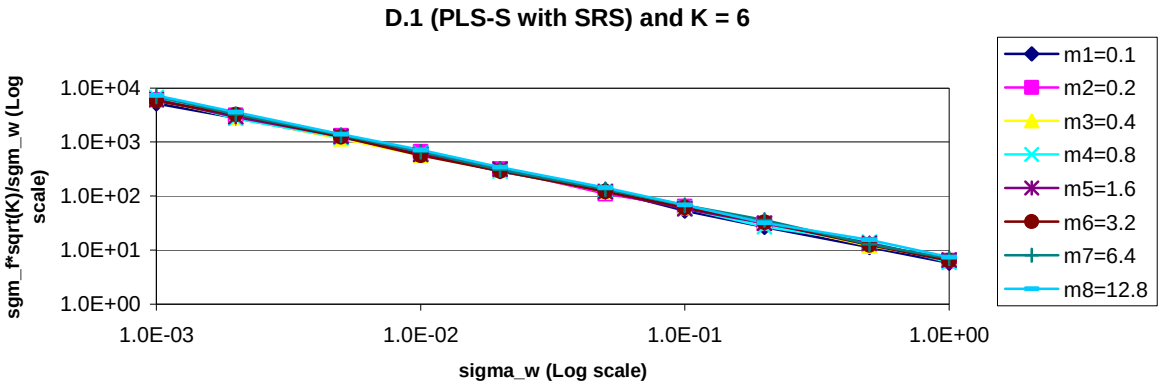


**Figure 51. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$n_p$ with algorithm D.1 (PLS-S with SRS) for $K$ = 6 and $\sigma_w$ = { 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1 }.**

and corresponding figure for standard deviations of $\sigma(\hat{f}_j)$ is

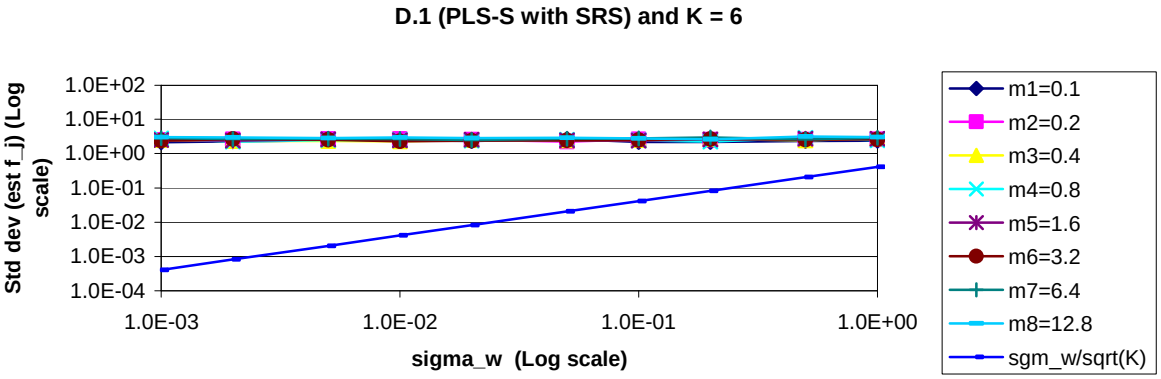


**Figure 52. Standard deviation $\sigma(\hat{f}_j)$ for j = 1,2,...,$n_p$ with algorithm D.1 (PLS-S with SRS) for $K$ = 6 and $\sigma_w$ = { 0.001, 0.002, 0.005, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1}.**

**Discussion of results for $K$ = 6**

The simulations for algorithm D.1 (PLS-S with SRS) with $K$ = 6, the results show that:

- For $\sigma_w \leq 1$, the obtained values for the standard deviations of $\sigma(\hat{f}_j)$ all fall within the range 2.1-3.2 (see Figure 52), and the obtained values for $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ increase with decreasing $\sigma_w$ (see Figure 51).
- It turns out that for $\sigma_w$ getting smaller, $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ becomes extremely high. This can be explained by a division by $\sigma_w$ for $\sigma_w \to 0$ with $K$ = 6 and finite values for $\sigma(\hat{f}_j)$, which implies that $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w \to \infty$.
- For $\sigma_w$ > 10, the values for $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ do not really converge to some fixed number, but instead converge to a range and within this range they can vary, the range is 2.3-4.2 (See Figure 47 and Figure 49).

## *7.3 Discussion of results obtained*

The effect of standard deviation $\sigma_w$ and number of samples $K$ on the normalized standard deviation $\sigma(\hat{F})\sqrt{K}/\sigma_w$ where $F^T = \begin{pmatrix} f_1 & f_2 & \cdots & f_{n_p} \end{pmatrix}$, has been analysed for PLS-S Algorithm D.1.

The main findings are:

- For $K$ large enough and such that $K > n_p + 1$, the Monte Carlo simulations showed that the standard deviations $\sigma(\hat{f}_j)$ for all $j = 1, 2, \ldots, n_p$ are proportional to standard deviation $\sigma_w$ and inversely proportional to the square root of $K$. The values of $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for $K$ large enough all fall within some 'converging area' (i.e. for $K$ = 1000, the range is 2.1-2.9), though within this area there is no convergence; the 'converging area' is independent of the value of standard deviation $\sigma_w$. Only in case $\sigma_w$ has extremely high values (e.g. $\sigma_w$ = 100 and $\sigma_w$ = 1000 in Figure 45), the mean $\mu(\hat{f}_j)$ becomes inaccurate.
- For $K = n_p + 1$, the values for $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ can become rather high (i.e. more than $10^3$) and this is independent of the value of $\sigma_w$, although for higher values of $\sigma_w$ this effect occurred more often. Similarly, for $K = n_p + 2$, the values for $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ can also become rather high, though this occurred less often compared to $K = n_p + 1$.
- For $K \leq n_p$, it turned out that the smaller $K$ is, the more inaccurate the obtained values for the mean $\mu(\hat{f}_j)$ for $j = 1, 2, \ldots, n_p$ become when compared to the true

values of $f_j$. Considering values of $\sigma_w$ for $K \le n_p$, it turns out that there is a difference between small values of $\sigma_w$ (say smaller than 1) or larger values of values of $\sigma_w$ (say larger than 10):

For small values of $\sigma_w$, i.e. for $\sigma_w \le 1$:

- the obtained values for the standard deviations $\sigma(\hat{f}_j)$ for $j = 1, 2, \ldots, n_p$ for the case that $K \le n_p$ all fall within a low range (e.g. for $K$ = 6, the range is 2.1-3.2), and
- the obtained values for $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ increase with decreasing $\sigma_w$. The smaller $\sigma_w$ becomes for $K$ = 6, it turns out that $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ becomes extremely high.

For large values of $\sigma_w$, i.e. for $\sigma_w$ > 10:

- the obtained values for values for $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ do not really converge to some fixed number, but instead converge to a range and within this range they can vary, for example for $K$ = 6, the range is is 2.3-4.2.

# 8 Effect of large value for $n_p$

In this section, the effect of a large value for the number of parameters $n_p$ will be analysed for PLS-S with sampling type SRS (i.e. Algorithm D.1) and how it relates to LS-MP with sampling type SRS (i.e. Algorithm B.1).
In this section it is assumed that the number of parameters $n_p$ = 200, for this value the following is discussed:

- Results for Algorithms B.1 versus D.1 with variation of $K$ in Subsection 8.1.
- Results for Algorithms B.1 versus D.1 with variation of standard deviation $\sigma_w$ in Subsection 8.2.
- A summarising discussion is given in Subsection 8.3.

## *8.1 Algorithm B.1 (LS-MP) vs. Algorithm D.1 (PLS-S) and variation of K*

Results for algorithms B.1 (LS-MP with SRS) and D.1 (PLS-S with SRS) with $n_p = 200$, for fixed value of the standard deviation $\sigma_w$ (i.e. $\sigma_w = 0.5$) and with variation of $K$, are presented in the subsection hereafter.

### 8.1.1 Variation of *K* and fixed value of $\sigma_w$ (=0.5)

Consider algorithms B.1 and D.1, for a fixed value of the standard deviation $\sigma_w$ and with variation of $K$, as specified in the following table:

| **Algorithm** | **Values for *K*** | **Values for $\sigma_w$** |
|---|---|---|
| B.1 (LS-MP wirh SRS)<br>D.1 (PLS-S with SRS) | *K* = {10, 20, 40, 50, 100, 160, 175, 200, 250, 500} | $\sigma_w$ = 0.5 |
| B.1 (LS-MP wirh SRS)<br>D.1 (PLS-S with SRS) | *K* = {150, 160, 170, 180, 190, 200, 201, 220, 500, 1000} | $\sigma_w$ = 0.5 |

The following three figures present the normalized standard deviation of $\sigma(\hat{f}_j)$, i.e. $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$, as a function of $K$ where both the horizontal and vertical axes are presented in a logarithmic scale,

- for Algorithm B.1 for a selection of parameter indices, i.e. for
  $j = 1, 27, 53, 79, 105, 131, 157, 183, n_p (= 200)$ and with assumed values for $f_j$
  $(f_1 \quad f_{27} \quad f_{53} \quad \cdots \quad f_{200}) = (0.1 \quad 1.7 \quad 3.2 \quad 4.8 \quad 6.3 \quad 7.9 \quad 9.4 \quad 11 \quad 12)$;
  for Algorithm D.1 for the same selection of parameter indices and the assumed values for $f_j$;
- for Algorithms B.1 and D.1 for parameter index $j = n_p = 200$.

For $\sigma_w$ = 0.5 and $K$ = {10, 20, 40, 50, 100, 160, 175, 200, 250, 500}:

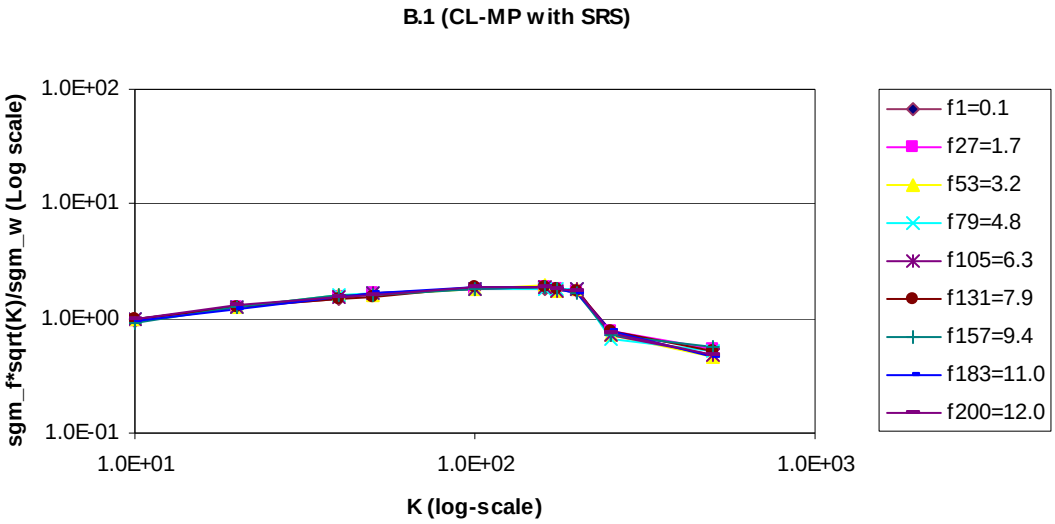


**Figure 53. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for ( $f_1$ $f_{27}$ $f_{53}$ … $f_{200}$ ) = (0.1 1.7 3.2 4.8 6.3 7.9 9.4 11 12) with algorithm B.1 (LS-MP with SRS) for $\sigma_w$ = 0.5 and $K$ = {10, 20, 40, 50, 100, 160, 175, 200, 250, 500}.**

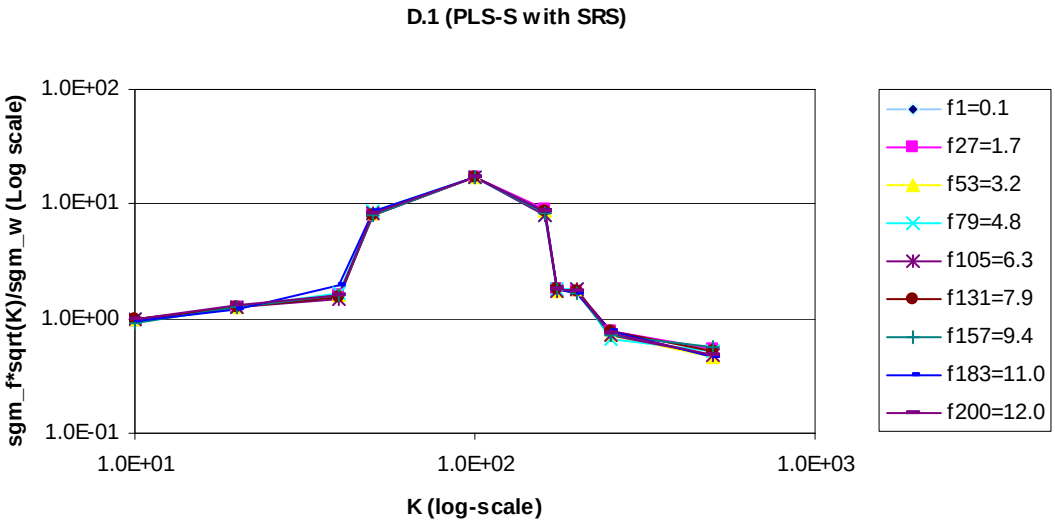


**Figure 54. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for ( $f_1$ $f_{27}$ $f_{53}$ … $f_{200}$ ) = (0.1 1.7 3.2 4.8 6.3 7.9 9.4 11 12) with algorithm D.1 (PLS-S with SRS) for $\sigma_w$ = 0.5 and $K$ = {10, 20, 40, 50, 100, 160, 175, 200, 250, 500}.**

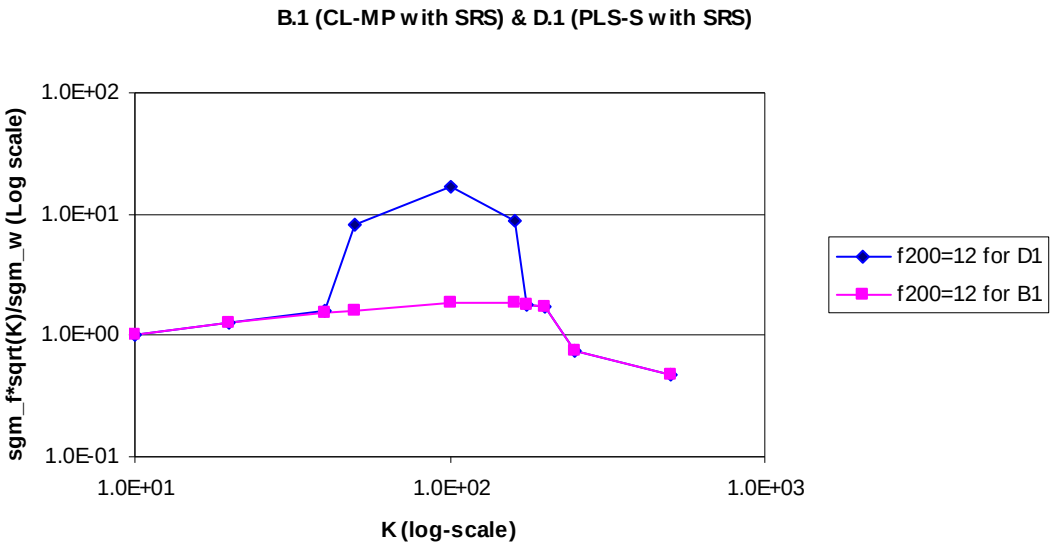


**Figure 55. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for parameter $f_{200}$ = 12.0 with algorithms B.1 and D.1 for $\sigma_w$ = 0.5 and $K$ = {10, 20, 40, 50, 100, 160, 175, 200, 250, 500}.**

These three figures (Figure 53-Figure 55) show that for $K$ = 50, 100 and 160 the values for the normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for algorithm D.1 deviates from those for algorithm B.1 (e.g for $K$ = 100, the normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for B.1 is 1.9, whereas for D.1 this is 17.0). This deviation between B.1 and D.1 occurs for some values for which $K < n_p$, though not for all as it does not occur for the lowest values $K$ = 10 and 20, and for high values $K$ = 175 and larger. For all values $K \geq 175$, both algorithms B.1 and D.1 give the same numerical results (i.e. numerical differences for $K = 175 < n_p$ are in the order of $10^{-5}$ or smaller; numerical differences for $K$ = 200, 250, 500 (i.e. values for which $K \geq n_p$) are in the order of $10^{-10}$ or smaller).

The following figure presents:

- The mean $\mu(\hat{f}_j)$ as a function of the parameter index $j = 1, 27, 53, \ldots, n_p (= 200)$ for each of the values of $K$ as specified in the above table, i.e. $K$ = {10, 20, 40, 50, 100, 160, 175, 200, 250, 500} and the true values for $(f_1 \quad f_{27} \quad f_{53} \quad \cdots \quad f_{200}) = (0.1 \quad 1.7 \quad 3.2 \quad 4.8 \quad 6.3 \quad 7.9 \quad 9.4 \quad 11 \quad 12)$.

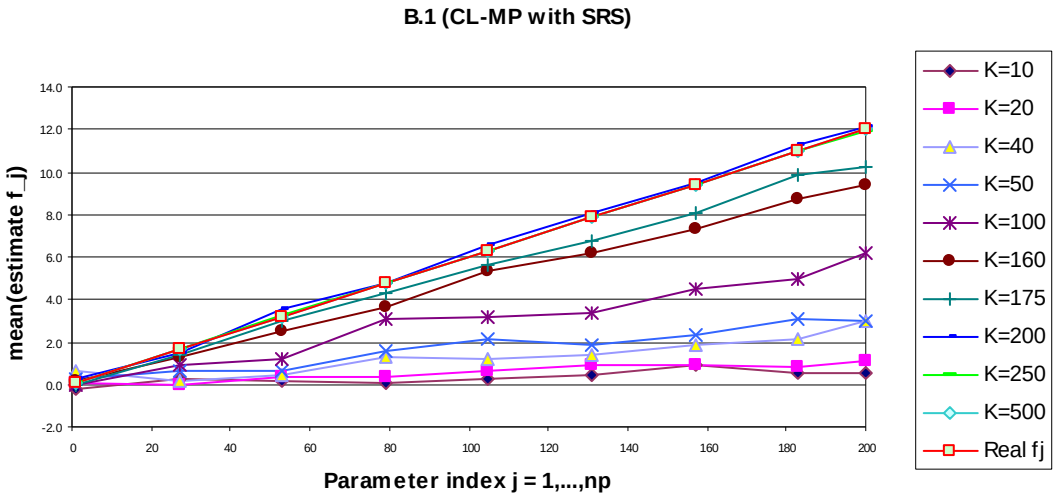


**Figure 56. Mean $\mu(\hat{f}_j)$ values as a function of j = 1,27,53,...,$n_p$ (=200) with algorithm B.1 (LS-MP with SRS) for $\sigma_w$ = 0.5 and $K$ = {10, 20, 40, 50, 100, 160, 175, 200, 250, 500}. The red line indicated as 'Real fj' represents the true values for $f_j$.**

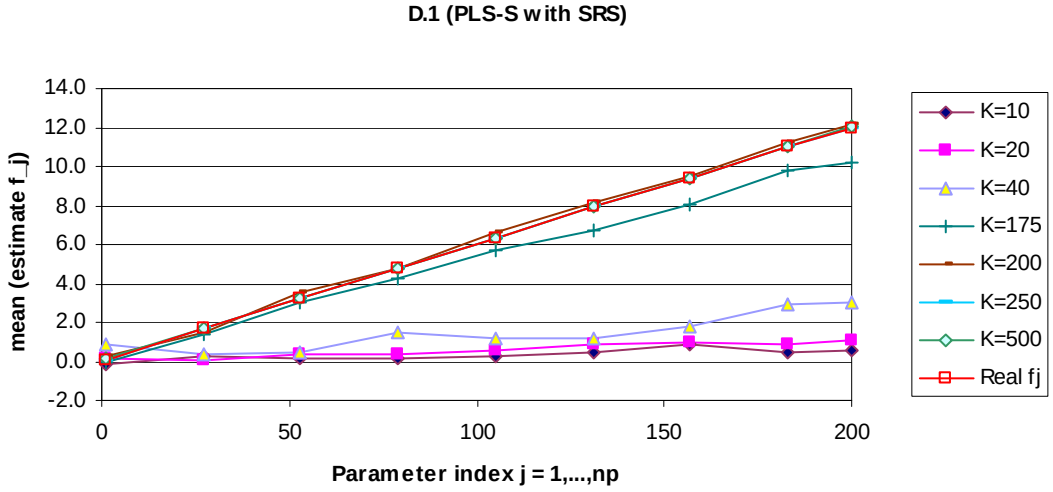


**Figure 57. Mean $\mu(\hat{f}_j)$ values as a function of j = 1,27,53,...,$n_p$ (=200) with algorithm D.1 (PLS-S with SRS) for $\sigma_w$ = 0.5 and $K$ = {10, 20, 40, 175, 200, 250, 500}. The red line indicated as 'Real fj' represents the true values for $f_j$.**

Figure 57 does not include the mean $\mu(\hat{f}_j)$ values for $K = \{50, 100, 160\}$ with algorihtm D.1 as their absolute values become extremely large, i.e. they vary between $-6 \cdot 10^{+14}$ and $+8 \cdot 10^{+14}$. This also explains that for $K = \{50, 100, 160\}$ the normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ with algorithm D.1 deviates from those of algorithm B.1.

Similar, though with other values for the horizontal axis, i.e. for $K = \{150, 160, 170, 180, 190, 200, 201, 220, 500, 1000\}$, i.e. including $K = n_p$ and $K = n_p + 1$, and again with $\sigma_w = 0.5$:

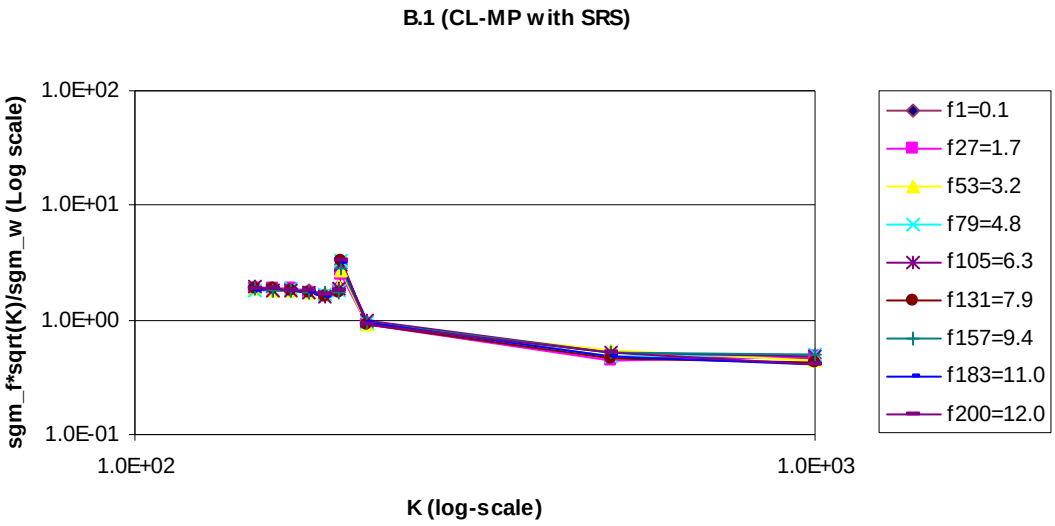


**Figure 58. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for ( $f_1$ $f_{27}$ $f_{53}$ … $f_{200}$ ) = (0.1 1.7 3.2 4.8 6.3 7.9 9.4 11 12) with algorithm B.1 (LS-MP with SRS) for $\sigma_w$ = 0.5 and $K$ = {150, 160, 170, 180, 190, 200 (=$n_p$), 201, 220, 500, 1000}.**

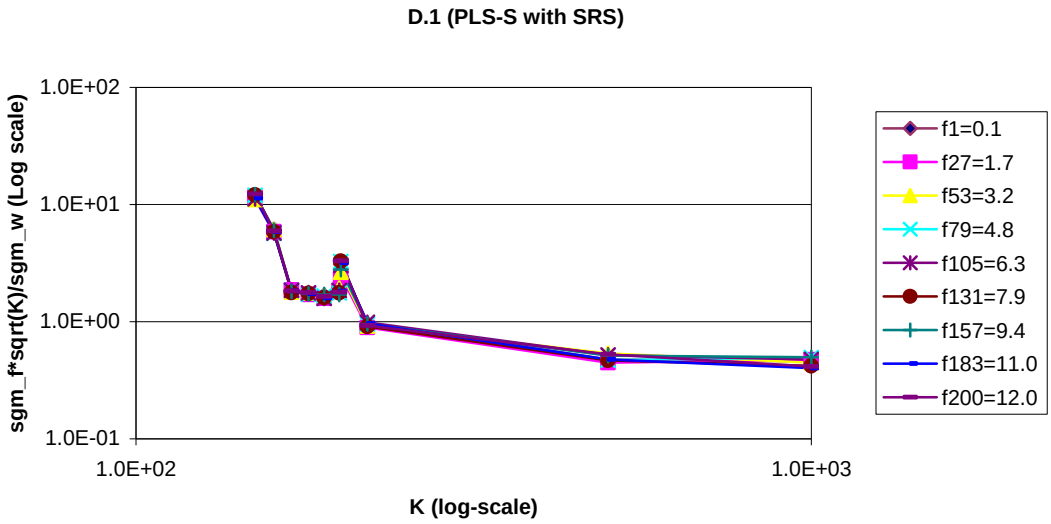


**Figure 59. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for ( $f_1$ $f_{27}$ $f_{53}$ … $f_{200}$ ) = (0.1 1.7 3.2 4.8 6.3 7.9 9.4 11 12) with algorithm D.1 (PLS-S with SRS) for $\sigma_w$ = 0.5 and $K$ = {150, 160, 170, 180, 190, 200 (=$n_p$), 201, 220, 500, 1000}.**

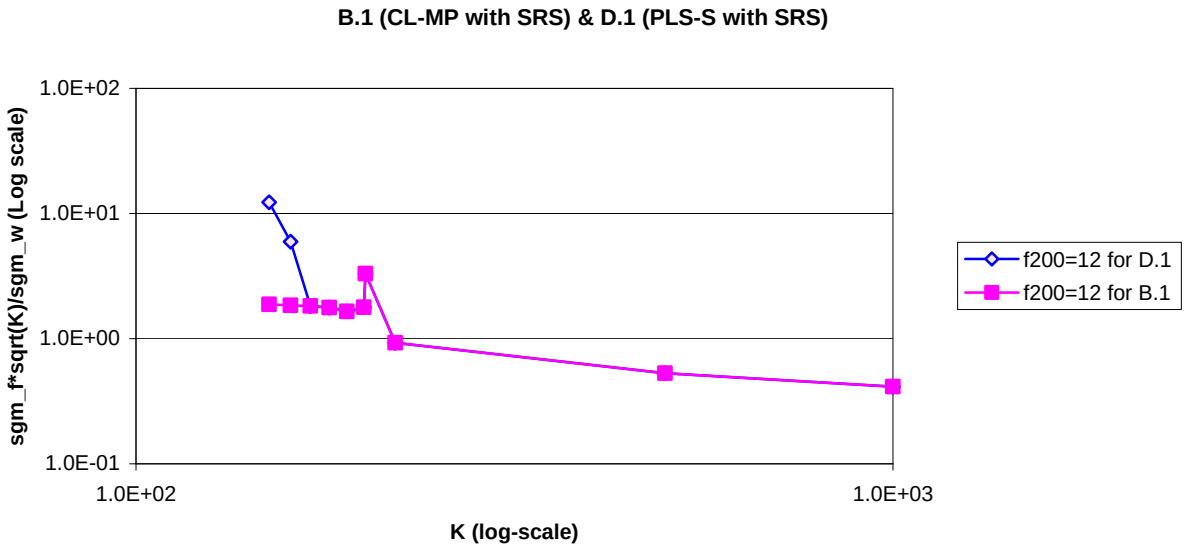


**Figure 60. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for parameter $f_{200}$ = 12.0 with algorithms B.1 and D.1 for $\sigma_w$ = 0.5 and $K$ = {150, 160, 170, 180, 190, 200 (=$n_p$), 201, 220, 500, 1000}.**

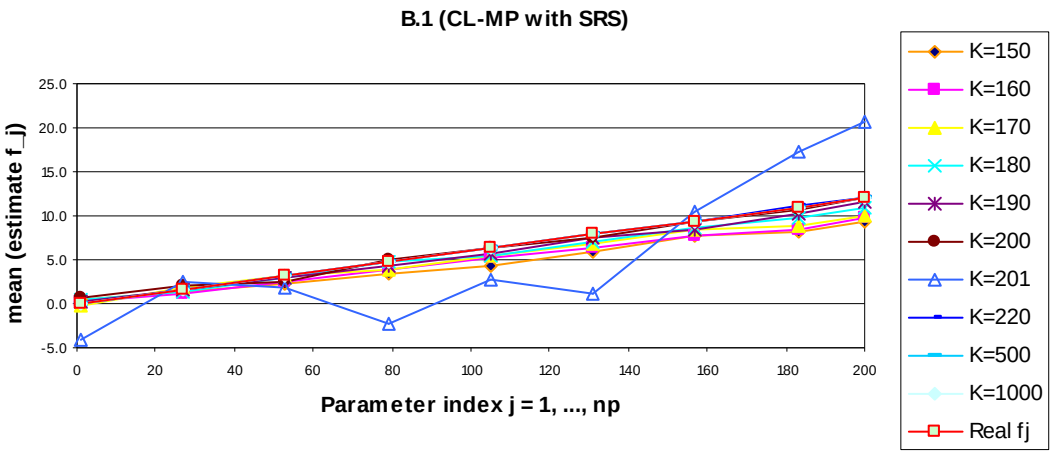


**Figure 61. Mean $\mu(\hat{f}_j)$ values as a function of j = 1,2,...,$n_p$ with algorithm B.1 (LS-MP with SRS) for $\sigma_w$ = 0.5 and $K$ = {150, 160, 170, 180, 190, 200 (=$n_p$), 201, 220, 500, 1000}. The red line indicated as 'Real fj' represents the true values for $f_j$.**

These figures (see Figure 59 and Figure 60) show that for $K$ = 150 and 160 the values for the normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for algorithm D.1 deviates from those for algorithm B.1. Related to this also the mean $\mu(\hat{f}_j)$ values for $K$ = 150 and 160 with algorihtm D.1 are extremely large in absolute sense (i.e. for $K$ = 150 they vary between between $-8 \cdot 10^{+9}$ and $5 \cdot 10^{+9}$ , and for $K$ = 160 they vary between between $-6 \cdot 10^{+3}$ and $4 \cdot 10^{+3}$).

For all values $K \geq 180$, both algorithms B.1 and D.1 give the same numerical results (i.e. numerical differences for $K$ = 201 are in the order of $10^{-7}$ or smaller, and for all other values $K \geq 180$, numerical differences are in the order of $10^{-10}$ or smaller).

For $K = n_p + 1$ (= 201), there is a peak in the normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$, both for algorithms B.1 and D.1; in addition there are inaccuracies in the corresponding mean $\mu(\hat{f}_j)$ (see Figure 61). Therefore this case is further analysed in Subsection 8.2.2 for different values of standard deviation $\sigma_w$.

**Discussion of results for $n_p$ = 200 and $\sigma_w$ = 0.5**

The simulations for algorithms B.1 (LS-MP with SRS) and D.1 (PLS-S with SRS) with $n_p = 200$ and $\sigma_w$ = 0.5 show that:

- For $K$ = 40, 50, ..., 160, 170, the values for the normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for algorithm D.1 deviates from those for algorithm B.1, and the corresponding mean $\mu(\hat{f}_j)$ algorithm D.1 becomes inaccurate compared to algorithm B.1. For $K$ = 40 and $K$ = 170, the deviations are still small, but not for the other values in this interval. For example for $K$ = 100, with algorithm D.1 the normalised standard deviations fall within the range 16.8-17.0, which corresponds to very high values of the standard deviation $\sigma(\hat{f}_j)$, i.e. in the order of $10^{+15}$, and with very inaccurate mean $\mu(\hat{f}_j)$ in the order of $10^{+14}$; whereas with algorithm B.1 the normalised standard deviations fall within the range 1.8-1.9 and the standard deviation $\sigma(\hat{f}_j)$ fall within the range 3.0-3.8.
- For $K$ = 10, 20 and for all $K \geq 175$, both algorithms give the same numerical results. (i.e. numerical differences for $K$ = 175 are in the order of $10^{-5}$ or smaller; numerical differences for $K \geq n_p$ = 200 are in the order of $10^{-10}$ or smaller). The values of $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ do not really converge to some fixed number, but instead converge to a range and within this range they can vary, the range is 0.3-0.5.
- For $10 \leq K \leq 200$ (i.e. $K \leq n_p$), the values of $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ with algorithm B.1 are within the range of 0.9 - 1.9.
- For $K$ = 201 (i.e. $K = n_p+1$), both algorithms B.1 and D.1 give the same numerical results; though the values of $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ show a peak, with values between 2.5 to 3.3, whereas for $K$ = 200 and $K$ > 201 the values of $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ are smaller (than 1.9).

## 8.2 Algorithm B.1 (LS-MP) vs. Algorithm D.1 (PLS-S) and variation of $\sigma_w$

Results for algorithms B.1 (LS-MP with SRS) and D.1 (PLS-S with SRS) with $n_p = 200$, for fixed value of $K$ and with variation of the standard deviation $\sigma_w$, are presented in the three subsections hereafter for three different values of *K* such that $K > n_p + 1$, $K = n_p + 1$ and $K < n_p$.

This subsection is organised as follows:

- In Subsection 8.2.1, we vary standard deviation $\sigma_w$ for $K$ = 250 (> $n_p$+1)
- In Subsection 8.2.2, we vary standard deviation $\sigma_w$ for $K$ = 201 (= $n_p$+1)
- In Subsection 8.2.3, we vary standard deviation $\sigma_w$ for $K$ = 195 (< $n_p$)

For an overall discussion about the effects of both the standard deviation $\sigma_w$ and the number of samples *K* on PLS-S algorithm D.1 the reader is referred to Subsection 8.3.

### 8.2.1 Variation of $\sigma_w$ and fixed value $K = 250$ (> $n_p$+1)

Consider algorithms B.1 and D.1, for a fixed value of *K* and with variation of the standard deviation $\sigma_w$, as specified in the following table:

| **Algorithm** | **Values for *K*** | **Values for $\sigma_w$** |
|---|---|---|
| B.1 (LS-MP wirh SRS)<br>D.1 (PLS-S with SRS) | $K$ = 250 | $\sigma_w$ = { 0.001, 0.01, 0.1, 0.2, 0.4,0.8, 1, 2, 5, 10 } |

The following figures present:

- The normalized standard deviation of $\sigma(\hat{f}_j)$, i.e. $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$, for fixed value of $K$ = 250 > $n_p$+1 as specified in the above table for parameter indices $j = 1, 27, 53, 79, 105, 131, 157, 183, n_p (= 200)$ as a function of $\sigma_w$ where both the horizontal and vertical axes are presented in a logarithmic scale, and with assumed values for $f_j$
  $(f_1 \quad f_{27} \quad f_{53} \quad \cdots \quad f_{200}) = (0.1 \quad 1.7 \quad 3.2 \quad 4.8 \quad 6.3 \quad 7.9 \quad 9.4 \quad 11 \quad 12)$;
- The corresponding standard deviation $\sigma(\hat{f}_j)$ as a function of $\sigma_w$ where both the horizontal and vertical axes are presented in a logarithmic scale.
- The corresponding mean $\mu(\hat{f}_j)$ as a function of the parameter number $j = 1, 27, 53, 79, 105, 131, 157, 183, n_p (= 200)$.

As it turned out that both algorithms B.1 and D.1 give numericaly the same results, the results are only presented for algorithm B.1 (LS-MP with SRS).

For $K$ = 250 and $\sigma_w$ = { 0.001, 0.01, 0.1, 0.2, 0.4,0.8, 1, 2, 5, 10 }the values of $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ vary between 4.3-6.7

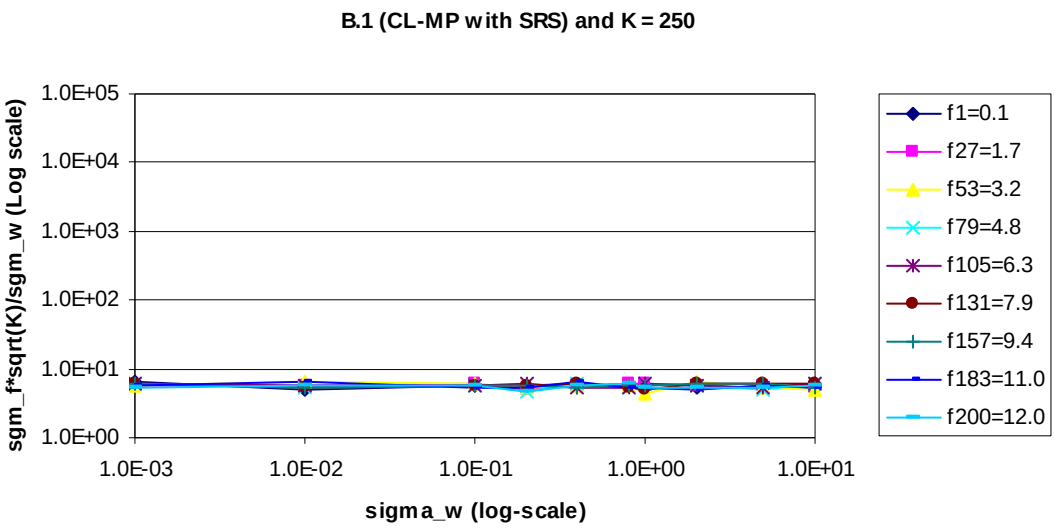


**Figure 62. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for ( $f_1$ $f_{27}$ $f_{53}$ … $f_{200}$ ) = (0.1 1.7 3.2 4.8 6.3 7.9 9.4 11 12) with algorithm B.1 (LS-MP with SRS) for $K$ = 250 and $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10 }.**

and corresponding standard deviation $\sigma(\hat{f}_j)$ is

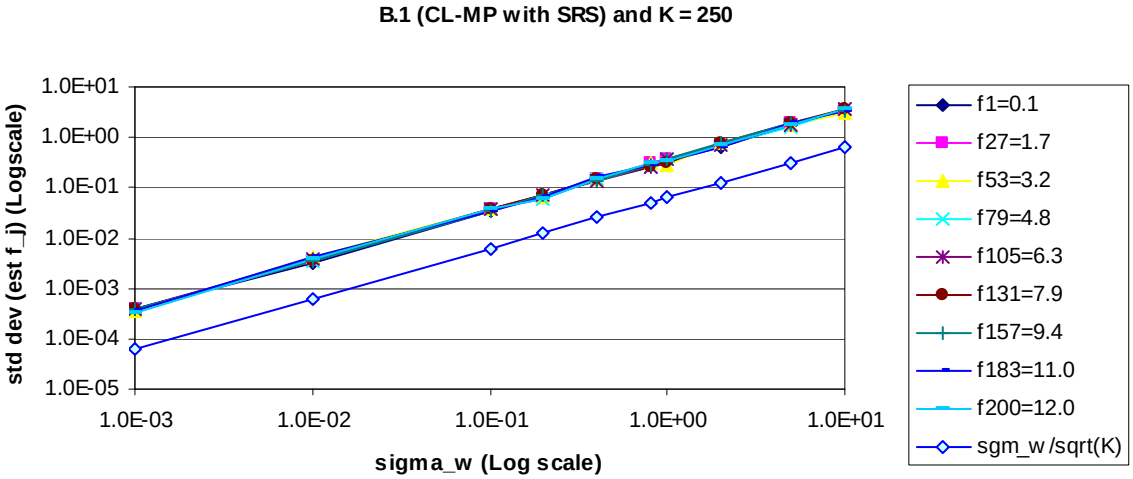


**Figure 63. Standard deviation $\sigma(\hat{f}_j)$ for ( $f_1$ $f_{27}$ $f_{53}$ … $f_{200}$ ) = (0.1 1.7 3.2 4.8 6.3 7.9 9.4 11 12) with algorithm B.1 (LS-MP with SRS) for $K$ = 250 and $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10 }.**

and corresponding mean $\mu(\hat{f}_j)$ is

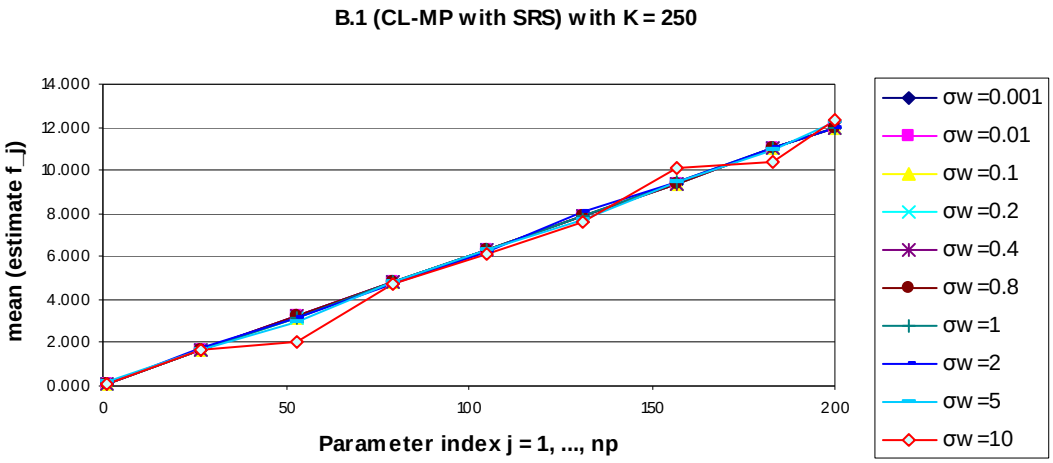


**Figure 64. Mean $\mu(\hat{f}_j)$ values as a function of j = 1,2,...,$n_p$ with algorithm B.1 (LS-MP with SRS) for $K$ = 250 and $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10 }. The red line indicated as 'Real fj' represents the true values for $f_j$.**

### 8.2.2 Variation of $\sigma_w$ and fixed value $K = 201 (= n_p+1)$

Consider algorithms B.1 and D.1, for a fixed value of $K$ and with variation of the standard deviation $\sigma_w$, as specified in the following table:

| **Algorithm** | **Values for $K$** | **Values for $\sigma_w$** |
|---|---|---|
| B.1 (LS-MP wirh SRS)<br>D.1 (PLS-S with SRS) | $K = 201$ | $\sigma_w$ = { 0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 } |

The following figures present:

- The normalized standard deviation of $\sigma(\hat{f}_j)$, i.e. $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$, for fixed value of $K = 201 = n_p + 1$ as specified in the above table for parameter indices $j = 1, 27, 53, 79, 105, 131, 157, 183, n_p (= 200)$ as a function of $\sigma_w$ where both the horizontal and vertical axes are presented in a logarithmic scale, and with assumed values for $f_j$
  $(f_1 \quad f_{27} \quad f_{53} \quad \cdots \quad f_{200}) = (0.1 \quad 1.7 \quad 3.2 \quad 4.8 \quad 6.3 \quad 7.9 \quad 9.4 \quad 11 \quad 12)$;
- The corresponding standard deviation $\sigma(\hat{f}_j)$ as a function of $\sigma_w$ where both the horizontal and vertical axes are presented in a logarithmic scale.
- The corresponding mean $\mu(\hat{f}_j)$ as a function of the parameter number $j = 1, 27, 53, 79, 105, 131, 157, 183, n_p (= 200)$.

As it turned out that both algorithms B.1 and D.1 give numericaly the same results, the results are only presented for algorithm B.1 (LS-MP with SRS).

For $K$ = 201 and $\sigma_w$ = { 0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 } the values of $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ can vary from low to very high (in the figure this is from $9.2 \cdot 10^1$ to $1.1 \cdot 10^4$):

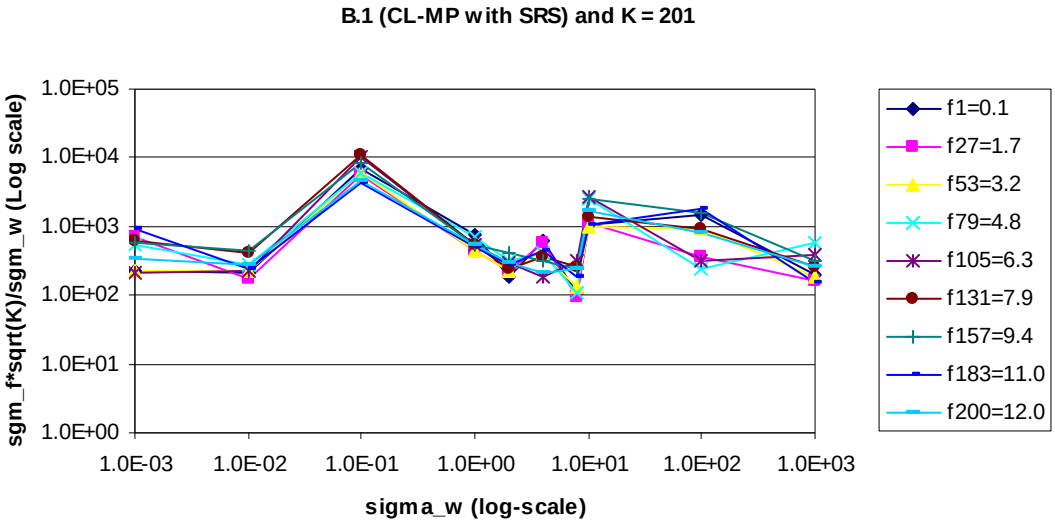


**Figure 65. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for ( $f_1$ $f_{27}$ $f_{53}$ … $f_{200}$ ) = (0.1 1.7 3.2 4.8 6.3 7.9 9.4 11 12) with algorithm B.1 (LS-MP with SRS) for $K$ = 201 and $\sigma_w$ = {0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 }.**

and corresponding standard deviation $\sigma(\hat{f}_j)$ is

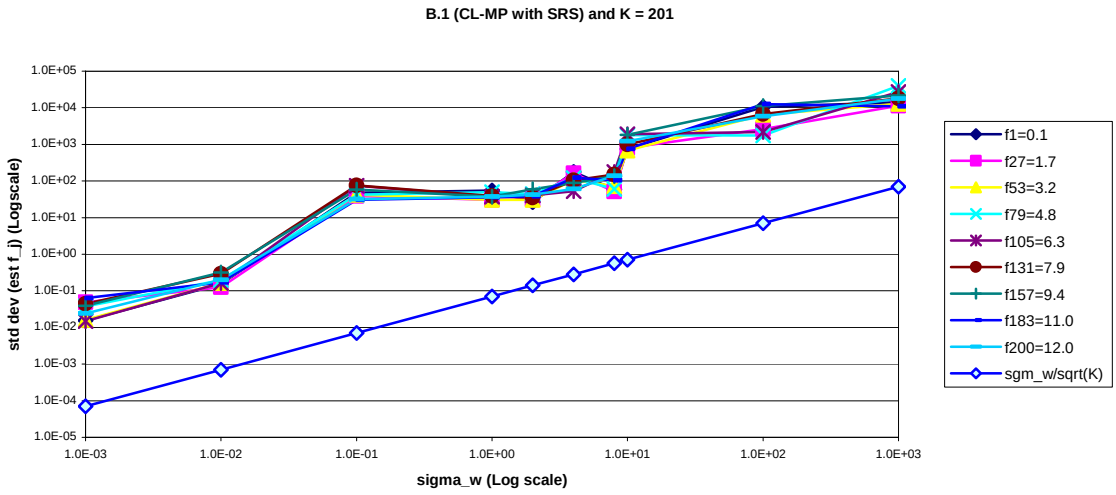


**Figure 66. Standard deviation $\sigma(\hat{f}_j)$ for ( $f_1$ $f_{27}$ $f_{53}$ … $f_{200}$ ) = (0.1 1.7 3.2 4.8 6.3 7.9 9.4 11 12) with algorithm B.1 (LS-MP with SRS) for $K$ = 201 and $\sigma_w$ = {0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100, 1000 }.**

### 8.2.3 Variation of $\sigma_w$ and fixed value $K$ =195 (< $n_p$)

Consider algorithms B.1 and D.1, for a fixed value of $K$ and with variation of the standard deviation $\sigma_w$, as specified in the following table:

| Algorithm | Values for $K$ | Values for $\sigma_w$ |
|---|---|---|
| B.1 (LS-MP wirh SRS)<br>D.1 (PLS-S with SRS) | $K$ = 195 | $\sigma_w$ = { 0.001, 0.01, 0.1, 0.2, 0.4,0.8, 1, 2, 5, 10 } |

The following figures present:

- The normalized standard deviation of $\sigma(\hat{f}_j)$, i.e. $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$, for fixed value of $K = 195 < n_p$ as specified in the above table for parameter indices $j = 1, 27, 53, 79, 105, 131, 157, 183, n_p (= 200)$ as a function of $\sigma_w$ where both the horizontal and vertical axes are presented in a logarithmic scale, and with assumed values for $f_j$
  $(f_1 \quad f_{27} \quad f_{53} \quad \cdots \quad f_{200}) = (0.1 \quad 1.7 \quad 3.2 \quad 4.8 \quad 6.3 \quad 7.9 \quad 9.4 \quad 11 \quad 12)$;
- The corresponding standard deviation $\sigma(\hat{f}_j)$ as a function of $\sigma_w$ where both the horizontal and vertical axes are presented in a logarithmic scale.
- The corresponding mean $\mu(\hat{f}_j)$ as a function of the parameter number $j = 1, 27, 53, 79, 105, 131, 157, 183, n_p (= 200)$.

As it turned out that both algorithms B.1 and D.1 give numericaly the same results, the results are only presented for algorithm B.1 (LS-MP with SRS).

For $K$ = 195 and $\sigma_w$ = { 0.001, 0.01, 0.1, 0.2, 0.4,0.8, 1, 2, 5, 10 }:

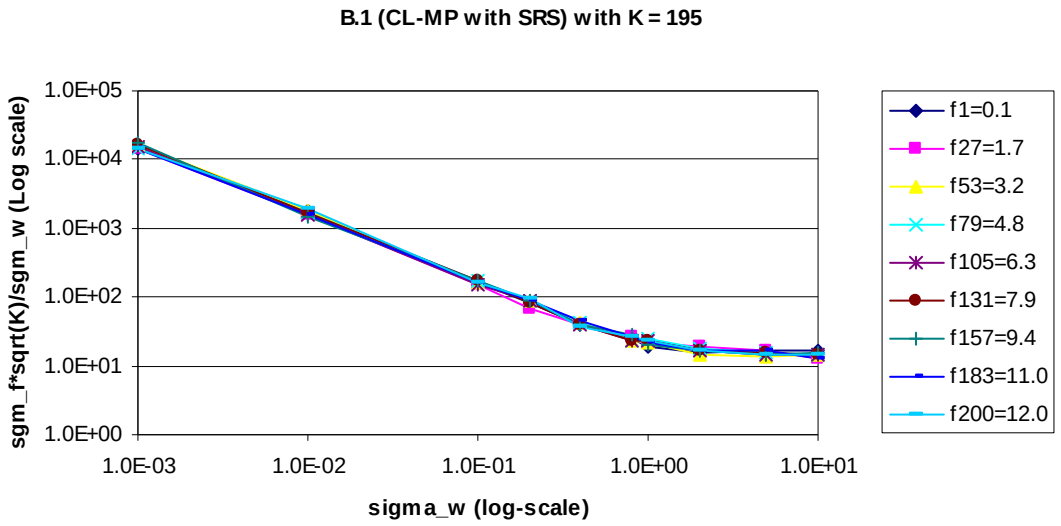


**Figure 67. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for ( $f_1$ $f_{27}$ $f_{53}$ ... $f_{200}$ ) = (0.1 1.7 3.2 4.8 6.3 7.9 9.4 11 12) with algorithm B.1 (LS-MP with SRS) for $K$ = 195 and $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10 }.**

and corresponding standard deviation $\sigma(\hat{f}_j)$ is

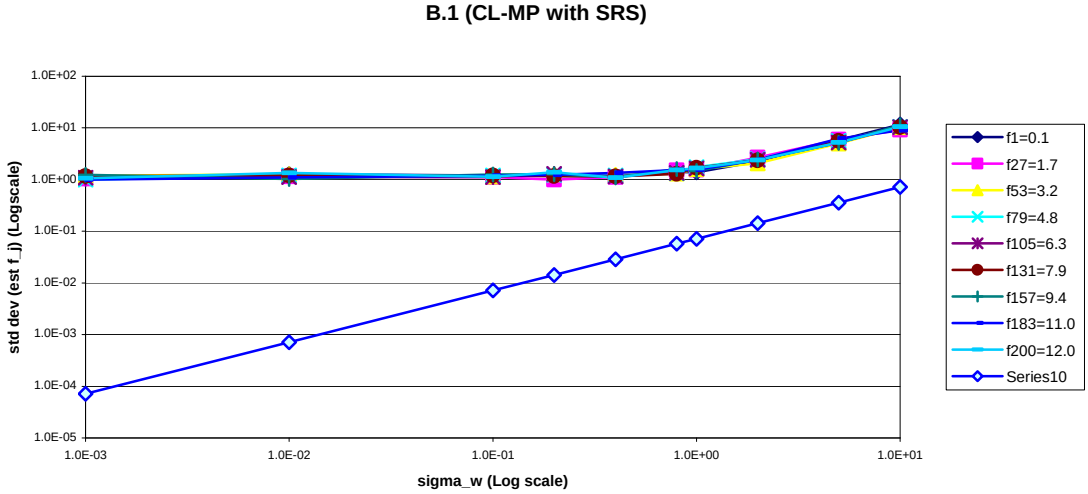


**Figure 68. Standard deviation $\sigma(\hat{f}_j)$ for ( $f_1$ $f_{27}$ $f_{53}$ ... $f_{200}$ ) = (0.1 1.7 3.2 4.8 6.3 7.9 9.4 11 12) with algorithm B.1 (LS-MP with SRS) for $K$ = 195 and $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10 }.**

and corresponding mean $\mu(\hat{f}_j)$ is

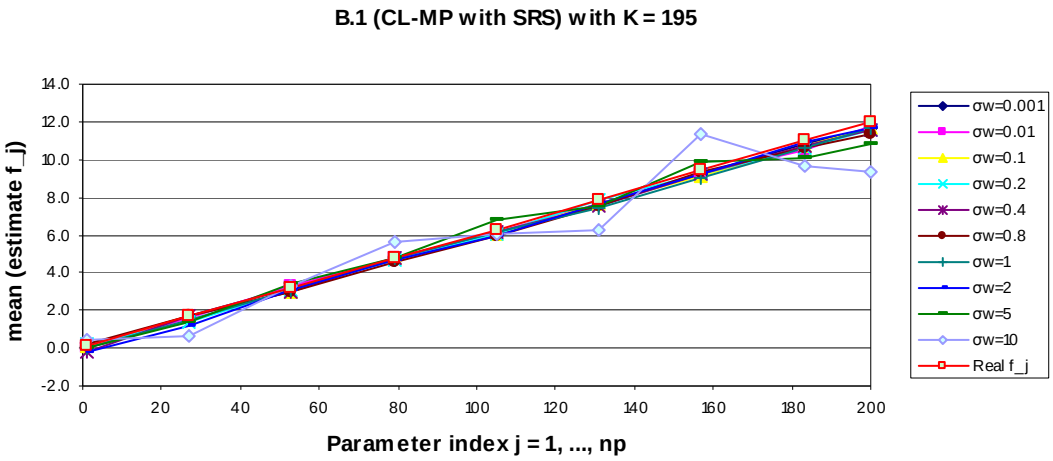


**Figure 69. Mean $\mu(\hat{f}_j)$ for ( $f_1$ $f_{27}$ $f_{53}$ ... $f_{200}$ ) = (0.1 1.7 3.2 4.8 6.3 7.9 9.4 11 12) with algorithm B.1 (LS-MP with SRS) for $K$ = 195 and $\sigma_w$ = {0.001, 0.01, 0.1, 0.2, 0.4, 0.8, 1, 2, 5, 10 }.**

### Discussion of results for $K$ = 195

The results of the simulations with $K$ = 195 show that:

- For $\sigma_w \leq 1$, the obtained values for the standard deviations of $\sigma(\hat{f}_j)$ all fall within the range 1.0-1.7 (see Figure 68), and the obtained values for $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ increase with decreasing $\sigma_w$ (see Figure 67).
- It turns out that for $\sigma_w$ getting smaller, $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ becomes extremely high. This can be explained by a division by $\sigma_w$ for $\sigma_w \to 0$ with $K$ = 195 and finite values for $\sigma(\hat{f}_j)$, which implies that $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w \to \infty$.
- For $\sigma_w$ = 10, the values for $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ do not really converge to some fixed number, but instead converge to a range and within this range they can vary, the range is 12.6-16.3 (see Figure 67).

### 8.3 Discussion of results obtained for large $n_p$

The effect of standard deviation $\sigma_w$ and number of samples *K* on the normalized standard deviation $\sigma(\hat{F})\sqrt{K}/\sigma_w$ where $F^T = \begin{pmatrix} f_1 & f_2 & \cdots & f_{n_p} \end{pmatrix}$ with large value $n_p = 200$, has been analysed for LS-MP with SRS and PLS-S with SRS.

The main findings are:

- For $K \le n_p$ and algorithm LS-MP with SRS, it turned out that the smaller *K* is, the more inaccurate the obtained values for the mean $\mu(\hat{f}_j)$ for $j = 1, 2, \ldots, n_p$ become when compared to the true values of $f_j$.
- For a subset of *K* values such that $K < n_p$, the results showed that algorithm LS-MP with SRS gives better results than algorithm PLS-S with SRS. This is illustrated by Figure 55, where for *K* = 50, ..., 160 and $n_p = 200$ the normalised standard deviations are larger for algorithm PLS-S with SRS than for algorithm LS-MP with SRS. For example for *K* = 100, with algorithm PLS-S with SRS the normalised standard deviations fall within the range 16.8-17.0, which corresponds to very high values of the standard deviation $\sigma(\hat{f}_j)$, i.e. in the order of $10^{+15}$, whereas with algorithm LS-MP with SRS the standard deviations fall within the range 1.8-1.9 and the standard deviation $\sigma(\hat{f}_j)$ fall within the range 3.0-3.8.

  In addition the obtained values for the mean $\mu(\hat{f}_j)$ for $j = 1, 2, \ldots, n_p$ for this set of *K* values (i.e. for *K* = 50, ..., 160 and $n_p = 200$) with algorihm PLS-S with SRS is very inaccurate when compared to the true values of $f_j$.

  This shows that for such a subset of *K* values such that $K < n_p$ with large $n_p$, algorithm PLS-S with SRS is not suitable.
- For $K \ge 175$, both algorithm LS-MP with SRS and PLS-S with SRS give numericaly the same results.

The next main findings apply to both algorithm LS-MP with SRS and algorithm PLS-S with SRS:

- For *K* large enough and such that $K > n_p + 1$, the Monte Carlo simulations showed that the standard deviations $\sigma(\hat{f}_j)$ for all $j = 1, 2, \ldots, n_p$ are proportional to standard deviation $\sigma_w$ and inversely proportional to the square root of *K*. The values of $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for *K* large enough all fall within some 'converging area' (i.e. for *K* = 250, the range is 4.3-6.7, see Figure 62), though within this area there is no convergence; the 'converging area' is independent of the value of standard deviation $\sigma_w$. Only in case $\sigma_w$ has extremely high values (e.g. $\sigma_w$ = 100 and

$\sigma_w$ = 1000), the mean $\mu(\hat{f}_j)$ becomes inaccurate. (For $\sigma_w$ = 10 some smaller inaccucaries are shown in Figure 69).

- For $K = n_p + 1$, the values for $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ can become rather high (i.e. more than $10^4$) and this is independent of the value of $\sigma_w$ (see Figure 65).
- Considering values of $\sigma_w$ for $K = 195 \leq n_p$, it turns out that there is a difference between small values of $\sigma_w$ (say smaller than 1) or larger values of values of $\sigma_w$ (say larger than 2):
  For small values of $\sigma_w$, i.e. for $\sigma_w \leq 1$:
  - the obtained values for the standard deviations $\sigma(\hat{f}_j)$ for $j = 1, 2, \ldots, n_p$ for the case that $K = 195 \leq n_p$ all fall within a low range (e.g. between 1.0-1.7), and
  - the obtained values for $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ increase with decreasing $\sigma_w$. The smaller $\sigma_w$ becomes for $K = 195 \leq n_p$, it turns out that $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ becomes extremely high.

  For large values of $\sigma_w$, i.e. for $\sigma_w$ > 2:
  - the obtained values for values for $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ do not really converge to some fixed number, but instead converge to a range and within this range they can vary, for example for *K* = 195 the range is is 12.6-16.3 (for $\sigma_w$ = 10).

# 9 Concluding remarks

In this report sensitivity analysis has been considered for IPS based collision risk estimation in ATM. The sensitivity estimation problem has been formulated as a multi-dimensional linear regression problem which estimates the multi-dimensional regression coefficients from IPS obtained input and output data sequences.

Several algorithms have been applied to the linear version of the sensitivity estimation problem. The algorithms are determined by the type of multi-dimensional regression method and the type of sampling method.

- The following types of multi-dimensional regression methods have been considered: Classical Least Squares (CLS), Least Squares with Moore-Penrose (LS-MP), Partial Least Squares based on NIPALS (PLS-N) and Partial Least Squares based on SIMPLS (PLS-S).
- Two sampling methods have been applied to draw samples for the input sequence, i.e. Standard Random Sampling (SRS) and Latin Hypercube Sampling (LHS).
- The output of the model is generated by assuming a random variable noise with Gaussian distribution.

It has been investigated which of the algorithms can best be applied to the multi-dimensional linear regression problem considered for collision risk estimation in ATM.

Based on simulations with a small number of regression components, i.e. with $n_p = 8$, it turned out that regarding sampling types SRS and LHS the results show that for all values of *K*, no systematic differences can be observed between results obtained with SRS and LHS. Figure 70 for example shows the normalized standard deviation for one of the regression parameters for algorithm LS-MP with both sampling types SRS and LHS.

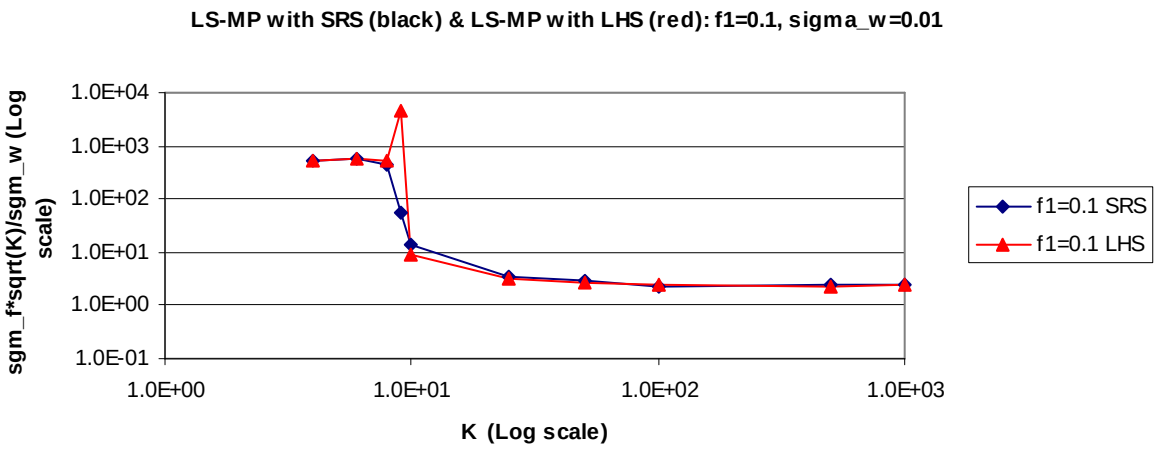


**Figure 70. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1, i.e. for parameter $f_1$ = 0.1, with algorithm LS-MP with SRS and algorithm LS-MP with LHS for $\sigma_w$ = 0.01 as a function of *K* = {4, 6, 8, 9, 10, 25, 50, 100, 500, 1000} for $n_p$ = 8 (See also Figure 16).**

It also has become clear that algorithms with CLS estimation can only be applied if the number of samples $K$ is larger than the number of regression components $n_p$, and for this case the simulations show that algorithms with LS-MP, PLS-N and PLS-S all give numerically the same results as CLS does. Figure 71 for example shows that LS-MP and PLS-N with $n_p = 8$ give the same results for $K \geq 9 (= n_p + 1)$.

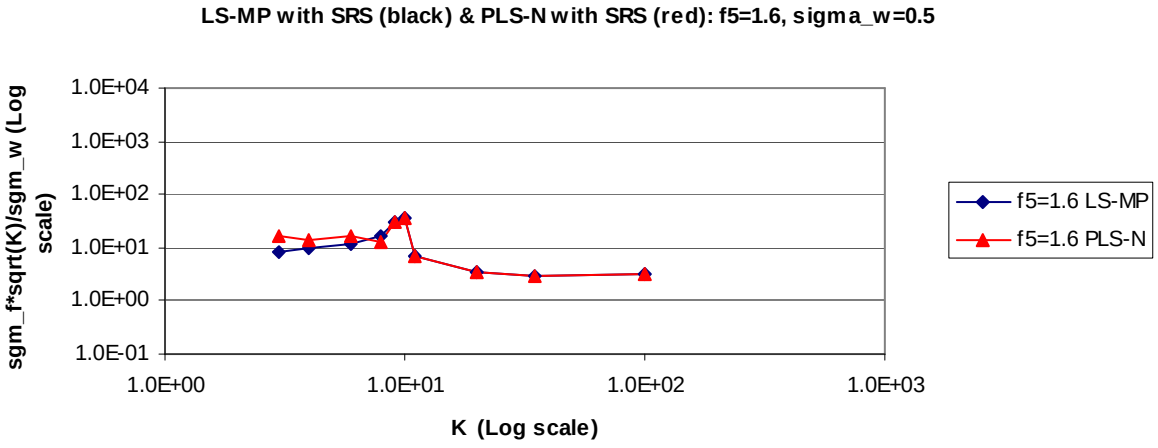


**Figure 71. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 5, i.e. for parameter $f_5$ = 1.6, with algorithm LS-MP with SRS and algorithm PLS-N with SRS for $\sigma_w$ =0.5 as a function of *K* = {3, 4, 6, 8, 9, 10, 11, 20, 35, 100} for $n_p$ = 8 (See also Figure 28).**

In contrast to CLS, algorithms with LS-MP, PLS-N and PLS-S can also be applied for the case that $K \le n_p$. This is for example shown in Figure 71 where LS-MP and PLS-N with $n_p = 8$ give different results for $K \le 8 (= n_p)$. Both LS-MP and PLS-S algorithms give the same numerical values for the mean $\mu(\hat{f}_j)$ and normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for $K \le n_p$. The PLS-N algorithm gives different values for the mean $\mu(\hat{f}_j)$ and normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for $K \le n_p$ when compared to LS-MP and PLS-S. Though no systematic differences can be observed between the results of PLS-N when compared to LS-MP and PLS-S.

As explained above, for a small value of $n_p$, it turned out that for all values of *K*, algorithms with LS-MP and PLS-S both give numerically the same results for the mean and standard deviations of the regression coefficient. Figure 72 shows that this is not the case for a large value $n_p$ of the number of regression components, for example with $n_p = 200$ it turned out that for a subset of *K* values in $K < n_p$, the results showed that the algorithm with LS-MP gives better results than the algorithm with PLS-S. In the simulations it turned out that for the subset *K* = 50, ..., 160 and $n_p = 200$ the PLS-S with SRS algorithm resulted in very high values of the standard deviation $\sigma(\hat{f}_j)$, i.e. even in the order of $10^{+15}$, and in very inaccurate values for the mean $\mu(\hat{f}_j)$, whereas the LS-MP with SRS algorithm did not result in such inaccuracies. For all other values of *K* than the above subset, both LS-MP and PLS-S give numerically the same results for the mean and standard deviations of the regression coefficient.

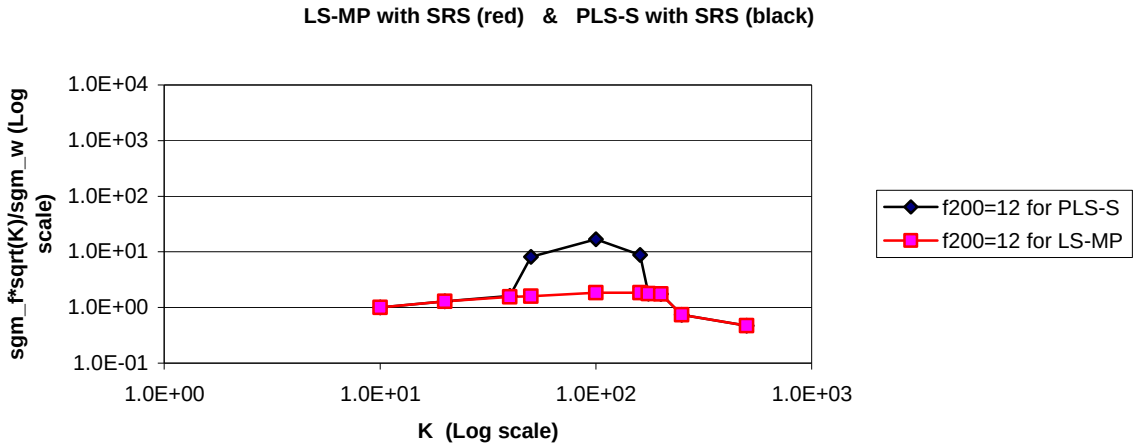


**Figure 72. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for parameter $f_{200}$ = 12.0 with algorithm LS-MP with SRS and algorithm PLS-S with SRS for $\sigma_w$ = 0.5 and $K$ = {10, 20, 40, 50, 100, 160, 175, 200, 250, 500} for $n_p$ = 200 (See also Figure 55).**

Based on the results obtained, the algorithm LS-MP with SRS is the preferred algorithm. For this algorithm, the effect of the standard deviation $\sigma_w$ of the noise and the number of samples $K$ on the normalized standard deviation of the estimation of the regression coefficient has been analysed (e.g. see Figure 73). In the Monte Carlo simulations, the resulting values for the mean and standard deviation of the $n_p$-dimensional regression coefficient have been considered. The mean values of the estimations of the regression coefficient have been compared to the assumed values of the regression coefficient, and the standard deviation values of the regression coefficient normalized by the standard deviation of the noise and the inverse of the square root of $K$ have been analysed.

Based on simulations with a small number of regression components, i.e. with $n_p = 8$, it turned out that for $K \leq n_p$, the smaller $K$ is, the more inaccurate the obtained mean values of the $n_p$-dimensional regression coefficient become when compared to the assumed values of the regression coefficient. On the other hand for $K > n_p + 2$, the obtained mean values of the regression coefficient approach the assumed values very accurately.
In addition, for $K$ larger than and close to $n_p$, the normalized standard deviations of the regression coefficient can sometimes (though not always) become rather high, e.g. more than $10^3$ as is for example shown in Figure 73 for $K = 9 (= n_p + 1)$.

For $K = n_p + 1$ this occurred more often than for $K = n_p + 2$, and in combination with higher values of $\sigma_w$ this effect occurred more often.

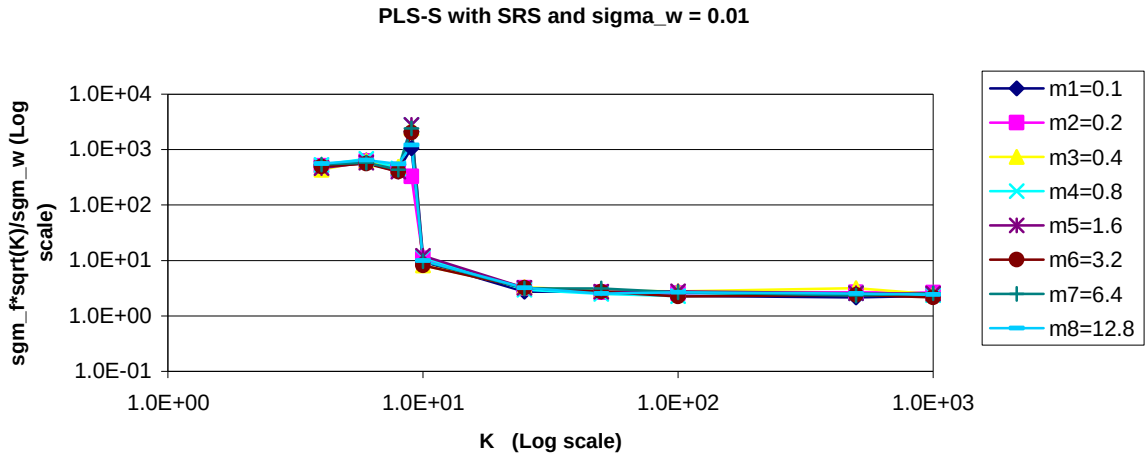


**Figure 73. Normalized standard deviation $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ for j = 1,2,...,$n_p$ with algorithm LS-MP with SRS for $\sigma_w$ = 0.01 and $K$ = {4, 6, 8, 9, 10, 25, 50, 100, 500, 1000} for $n_p$ = 8 (Similar to Figure 30 for algorithm PLS-S with SRS).**

Also for a small number of regression components, i.e. with $n_p = 8$ and $K$ large enough such that $K > n_p + 1$ (seeFigure 73) the normalized standard deviations of the regression coefficient do not really converge to some fixed number, but instead converge to a range (in the simulations, the size of this range is below 3) though within this range they can vary. This range is independent of the value of the standard deviation $\sigma_w$ of the noise, though it is dependent of the value of $K$, since the larger $K$ is, the smaller the values of this range are.

Only for extremely high values of the standard deviation $\sigma_w$ of the noise, the mean values of the estimations of the regression coefficient become very inaccurate when compared to the assumed values of the regression coefficient. Figure 74, for example shows the inaccuracies for $\sigma_w = 100$, whereas for $\sigma_w = 1000$ they have not even been shown in the figure, since they are very inaacurate and the figure would become unreadable. Related to these high values for $\sigma_w$, also the standard deviations of the regression coefficient become very large.

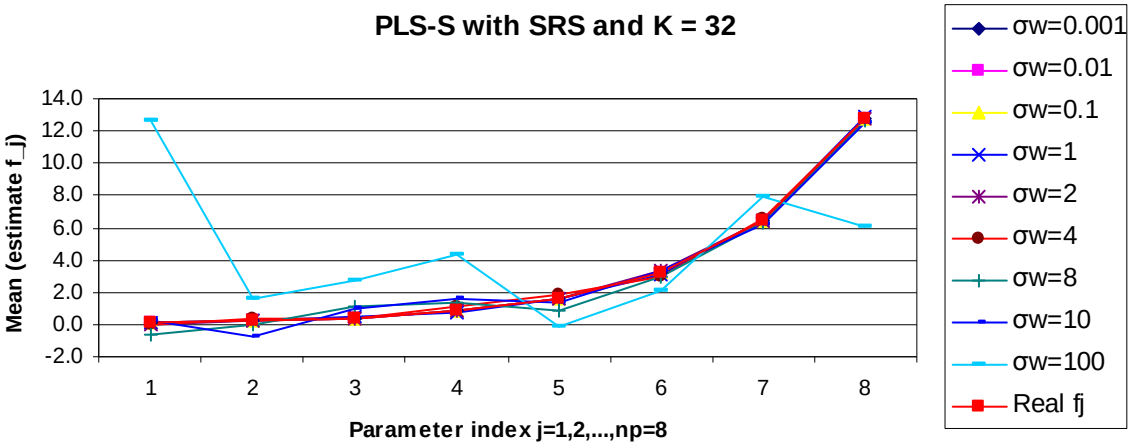


**Figure 74. Mean $\mu(\hat{f}_j)$ values as a function of j = 1,2,...,$n_p$ with algorithm LS-MP with SRS for $K$ = 32 and $\sigma_w$ = {0.001, 0.01, 0.1, 1, 2, 4, 8, 10, 100} for $n_p$ = 8. The red line indicated as 'Real fj' represents the true values for $f_j$ } (Similar to Figure 45 for algorithm PLS-S with SRS).**

These results confirm that LS-MP with SRS is the preferred algorithm. But even LS-MP with SRS cannot do the impossible as is shown in Figure 74, when the $K$ is smaller than or equal to the dimension $n_p$ of the regression coefficient or when the standard deviation $\sigma_w$ of the noise becomes unreasonably high, the normalized estimation errors tend to increase. Therefore it is best to choose a value $K$ that is larger than though not too close to $n_p$, and moreover very high values for the standard standard deviation $\sigma_w$ of the noise should be avoided.

## References


[Abdi, 2003] Abdi, H. (2003). Partial regression coefficients. In M. Lewis-Beck, A. Bryman, T. Futing (Eds): Encyclopedia for research methods for the social sciences. Thousand Oaks (CA): Sage. pp.978-982.

[Abdi, 2007] Abdi, H. (2007). Partial least square regression (PLS regression). In N.J. Salkind (Ed.): Encyclopedia of Measurement and Statistics. Thousand Oaks (CA): Sage. pp. 740-744. http://www.utdallas.edu/~herve/Abdi-PLSR2007-pretty.pdf

[Alin, 2009] Alin, A., Comparison of PLS algorithms when number of objects is much larger than number of variables, Statistical papers, Vol 50, No. 4,pp. 711-720, 2009.

[Blom et al., 2007] Blom, H.A.P., J. Krystul, G.J. Bakker, M.B. Klompstra, B. Klein Obbink, Free flight collision risk estimation by sequential Monte Carlo simulation, Eds: C.G. Cassandras and J. Lygeros, Stochastic hybrid systems, Taylor & Francis/CRC Press, 2007, chapter 10, pp. 249-281

[Blom et al., 2009] Blom, H.A.P., G.J. Bakker, J. Krystul, Rare event estimation for a large scale stochastic hybrid system with air traffic application, Eds: G. Rubino and B. Tuffin, Rare event simulation using Monte Carlo methods, J.Wiley, forthcoming 2009

[Boulesteix & Strimmer, 2006]. Boulesteix, A.-L. and K. Strimmer, Partial least squares: a versatile tool for the analysis of high-dimensional genomic data, Briefings In Bioinformatics. Vol 8. No 1. 32, Advance Access publication May 26, 2006

[Cacuci, 2003] Cacuci, D.G., Sensitivity and uncertainty analysis, Vol 1, Theory, Chapman & Hall/CRC, 2003.

[Campbell and Meyer, 1979] Campbell, S.L. and C.D. Meyer, Generalized inverses of linear transformations, Pitman, London; San Francisco, 1979.

[Courrieu, 2005] Courrieu, P., Fast computation of Moore-Penrose Inverse Matrices, Neural Information Processing - Letters and Reviews Vol.8, No.2, , Pp 25-29, 2005, August 2005.

[De Jong, 1993] De Jong, S., SIMPLS: an alternative approach to partial least squares regression. Chemometrics and Intelligent Laboratory Systems, 18, pp. 251-263, 1993

[DeRocquigny et al, 2008] De Rocquigny, E., N. Devictor and S. Tarantola (eds.), Uncertainty in industrial practice, Wiley, 2008.

[Everdij & Blom, 2005] Everdij, M.H.C., H.A.P. Blom, Bias and uncertainty modelling in accident risk assessment, NLR Report CR-2006-284, HYBRIDGE Project Deliverable PD14 (D8.4), 2005

[Everdij & Blom, 2002] Everdij, M.H.C., G.J. Bakker and H.A.P. Blom, Estimating safe separation criteria, CARE/ASAS Activity 3: Airborne Separation Minima, WP3 final report, Version 2.0, January 2002

[Everdij et al, 2006] Everdij, M.H.C., H.A.P. Blom, S.H. Stroeve, Structured assessment of bias and uncertainty in Monte Carlo simulated accident risk, Proc. 8th Int. Conf. on Probabilistic Safety Assessment and Management (PSAM8), New Orleans, USA, May 2006.

[Geladi & Kowalski, 1986] Geladi, P, and Kowalski, B. Partial least-squares regression: a tutorial, Analytica Chimica Acta, Volume 185, Pages 1-17, 1986.

[Golub and van Loan, 1996] Golub G., and C. van Loan. *Matrix Computations*, 3rd edition. Johns Hopkins University Press, 1996.

[Helland, 1988] Helland, I.S. On the structure of PLS regresssion, Communications in Statistics. Part B-Simulation Computat 17 581-607, 1988.

[Helton, 1993] Helton, J. C., Uncertainty and sensitivity analysis techniques for use in performance assessment for radioactive waste disposal Reliability Engineering and System Safety, 42, 327-367, 1993.

[Höskuldsson, 1988] Höskuldsson, A., PLS regression methods. Journal of Chemometrics, 2:211-228, 1988

[iFly D7.2a] iFly Deliverable D7.2.a, *Review of risk assessment status for air traffic*. Editors H.A.P. Blom, J. Krystul, P. Lezaud, M.B. Klompstra, V1.0 of 14 January 2009. Available on iFly website http://ifly.nlr.nl/

[iFly D7.2b] iFly Deliverable D7.2.b, Trans-dimensional simulation for rare-events estimation on stochastic hybrid systems by N. Kantas and J.M. Maciejowski, 1 May 2009. Available on iFly website http://ifly.nlr.nl/

[iFly D7.2c] iFly Deliverable D7.2.c, Interim Report on Importance sampling of multi aircraft encounter geometries.

[iFly D7.2d] iFly Deliverable D7.2.d, IPS extension to multiple aircraft Periodic Boundary Condition in Large Scale Random Air Traffic Scenarios, Version: 0.7, by A. Goswami and G.J. Bakker.

[iFly D7.2e] iFly Deliverable D7.2.e, Rare event estimation for a large scale stochastic hybrid system with air traffic application - Interacting particle system (IPS) extension to large hybrid systems - by H.A.P. Blom, G.J. Bakker and J. Krystul, 28 Jan 2009. Available on iFly website http://ifly.nlr.nl/

[Kurowicka & Cooke, 2006] Kurowicka D. and R. Cooke, Uncertainty analysis with high dimensional dependence modelling, Wiley, 2006.

[Manne, 1987] Manne, R., 1987. Analysis of two partial-least-squares algorithms for multivariate calibration. Chemometrics and Intelligent Laboratory Systems, 2:187-197.

[Mevik & Wehrens, 2007] Mevik, B.H. and R. Wehrens, The PLS Package: Principal Component and Partial Least Squares Regression in R, Journal of Statistical Software, January 2007, Volume 18, Issue 2. http://www.jstatsoft.org/

[Morgan & Henrion, 1990] Morgan M.G. and M. Henrion, Uncertainty, Cambridge University Press, 1990.

[Morris, 1991] Morris, M.D.**,** Factorial sampling plans for preliminary computational experiments. Technometrics, Vol. 33, pp. 161–174, 1991.

[Ränner et al, 1994] Rännar, S., F. Lindgren, P. Geladi and S. Wold, A PLS kernel algorithm for data sets with many variables and few objects. Part I: Theory and algorithm. Journal of Chemometrics, Volume 8, Pages 111 – 125, 1994

[Rosipal & Krämer, 2006] Rosipal, R. and N. Krämer. "Overview and Recent Advances in Partial Least Squares, in Subspace, Latent Structure and Feature Selection Techniques, pp 34–51", 2006

[Rosipal & Trejo, 2001] Rosipal R., L.J. Trejo Kernel Partial Least Squares Regression in Reproducing Kernel Hilbert Space. Journal of Machine Learning Research, Vol 2, Pp.97-123, December 2001.

[Saltelli, 2002] Saltelli, A., Making best use of model valuations to compute sensitivity indices, Computer Physics Communications, Vol. 145, pp. 280-297, 2002.

[Saltelli et al, 2008] Saltelli, A., M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana and S. Tarantola, Global Sensitivity Analysis: The Primer, Wiley, 2008.

[Sobol, 1993] Sobol, I. M., Sensitivity analysis for non-linear mathematical model, Mathematical Modelling Computational Experiment, Vol. 1, pp. 407-414, 1993.

[Strang, 1980] Strang, G., Linear Algebra and its Applications, second edition 1980.

[Swiler & Giunta, 2007] Swiler, L.P. and Giunta, A.A., Aleatory and epistemic uncertainty quantification for engineering applications, Proceedings of the Joint Statistical Meetings, July 29 - Aug. 2, 2007, Denver, CO. (http://dakota.sandia.gov/papers/JSM_Aug2007_Swiler_Format1.pdf)

[Wold, 1966] Wold H., Estimation of principal components and related models by iterative least squares. In: Krishnaiah PR (ed). Multivariate Analysis. New York: Academic Press,Pages 391–420, 1966.

[Wold, 1973] Wold H., Nonlinear Iterative Partial Least Squares (NIPALS) modeling: some current developments. In: Krishnaiah PR (ed). Multivariate Analysis. New York: Academic Press, Pages 383–407, 1973.

[Wold, 1975] Wold H., Path models with latent variables: the NIPALS approach. In: Blalock HM (ed). Quantitative Sociology: International Perspectives on Mathematical and Statistical Model Building. New York: Academic Press, 1975.

[Wold et al, 1984] Wold, S., A. Ruhe, H. Wold, and W. J. Dunn, Collinearity problem in linear regression. The partial least squares (PLS) approach to generalized inverses. SIAM Journal of Scientific and Statistical Computing; Vol. 5, Issue 3, Pages 735–43, September 1984.

# Appendix A Partial Least Squares regression

The Partial Least Squares (PLS) regression method is explained in this appendix. It first starts with the Multi-dimensional Linear Regression (MLR) problem, and then describes the PLS regression.

## A.1 Multi-dimensional Linear Regression (MLR)

In [Geladi & Kowalski, 1986] it is assumed that the values of each variable are used in the mean-centered form. The MLR problem in terms of the mean-centered $X_0$ and mean-centered $y_0$ is described as follows [Geladi & Kowalski, 1986].

Consider

$$y_0 = X_0 b + e \tag{A.1}$$

or written in full:

$$\begin{pmatrix} y_1^0 \\ y_2^0 \\ \vdots \\ y_n^0 \end{pmatrix} = \begin{pmatrix} x_{1,1}^0 & \cdots & x_{1,m}^0 \\ \vdots & & \vdots \\ x_{n,1}^0 & \cdots & x_{n,m}^0 \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{pmatrix} + \begin{pmatrix} e_1 \\ e_2 \\ \vdots \\ e_n \end{pmatrix}$$

which describes multi-linear dependencies for $n$ samples, where

- $y_0$: $n\times1$, $y_0$ is the column vector for $n$ samples (also called *one dependent variable* or *one response variable*)
- $X_0$: $n\times m$, ($X_0$ : independent variable)
- $b$: $m\times1$ (sensitivities; regression coefficients)
- $e$: $n\times1$ (error or residual)
- $n$: is the number of samples
- $m$: is the number of independent variables

The least squares solution of (A.1) is

$$\left(X_0^T X_0\right)\hat{b} = X_0^T y_0 \tag{A.2}$$

or

$$\hat{b} = \left(X_0^T X_0\right)^{-1} X_0^T y_0 \quad \text{if inverse } \left(X_0^T X_0\right)^{-1} \text{ exists}$$

The Multi-dimensional Linear Regression (MLR) formula with more than one dependent variable is described as follows [Geladi & Kowalski, 1986] in terms of the mean-centered $X_0$ and mean-centered $Y_0$ is as follows

$$Y_0 = X_0 B + \mathcal{E} \tag{A.3}$$

or written in full:

$$\begin{pmatrix} y_{1,1}^0 & \cdots & y_{1,p}^0 \\ \vdots & & \vdots \\ y_{n,1}^0 & \cdots & y_{n,p}^0 \end{pmatrix} = \begin{pmatrix} x_{1,1}^0 & \cdots & x_{1,m}^0 \\ \vdots & & \vdots \\ x_{n,1}^0 & \cdots & x_{n,m}^0 \end{pmatrix} \begin{pmatrix} b_1^1 & \cdots & b_1^p \\ \vdots & & \vdots \\ b_m^1 & \cdots & b_m^p \end{pmatrix} + \begin{pmatrix} e_{1,1} & \cdots & e_{1,p} \\ \vdots & & \vdots \\ e_{n,1} & \cdots & e_{n,p} \end{pmatrix}$$

where

- $Y_0$ : $n \times p$ matrix $Y_0$ is referred to as the dependent variable, with $p > 1$, i.e. *at least two response variables*); is also referred to as *matrix of responses* or called *matrix of dependent variables* [Abdi, 2003]
- $X_0$ : $n \times m$ matrix $X_0$ is referred to as independent variable; is also referred to as *matrix of predictors* or *regressors* [Abdi, 2003]
- $B$: $m \times p$ (matrix of sensitivities for the MLR method)
- $\mathcal{E}$: $n \times p$ (matrix of error or residual)
- $n$: is the number of samples
- $m$: is the number of independent variables
- $p$: is the number of dependent variables

Estimator $\hat{B}$ satisfies the following equation

$$\left(X_0^T X_0\right)\hat{B} = X_0^T y_0 \tag{A.4}$$

or

$$\hat{B} = \left(X_0^T X_0\right)^{-1} X_0^T y_0 \text{ if inverse } \left(X_0^T X_0\right)^{-1} \text{ exists}$$

Consider three cases:

1. $n < m$, i.e. more variables ($m$) than samples ($n$)
   In this case rank $(X_0^T X_0) \leq n < m$, hence matrix $X_0^T X_0$ is not invertible. Classical Least Squares cannot be applied for this case.

2. $n = m$, i.e. number of variables ($m$) and samples ($n$) are equal
   This case gives a unique solution for $B$ provided that $X_0$ has full rank (in which case matrix $X_0^T X_0$ is invertible). For $n = m$, the matrix inversion can cause problems because the matrix $X_0^T X_0$ might be ill-conditioned.

3. $n > m$, i.e. more samples ($n$) than variables ($m$)
   Similarly as for $n = m$, this case gives a unique solution for $B$ provided that $X$ has full rank (in which case matrix $X_0^T X_0$ is invertible), though the matrix inversion can cause problems because the matrix $X_0^T X_0$ might be ill-conditioned.

Partial Least Squares methods can be applied to handle Multi-dimensional Linear Regression problems where matrix $X_0^T X_0$ is not invertible or is ill-conditioned.

### A.2 Partial Least Squares Regression

Partial Least Squares methods, originally developed in the 1960s and 1970s by the econometrician Herman Wold (1966, 1973, 1975) to address problems in econometric path modelling, and are useful for models with many variables but not necessarily many samples or observations. The PLS method was subsequently adopted by his son Svante Wold and many others in the 1980s for regression problems in chemometric and spectrometric modeling. One of the first applications of PLS to regression is [Wold et al., 1984]. The stability of the predictors derived from PLS methods make PLS regression methods perform better than other well known regression techniques [Höskuldsson, 1988]. PLS regression allows more independent variables $m$ than the number of samples $n$ and is able to deal with singularity, see [Ränner et al, 1994] and [Rosipal & Trejo, 2001]. In PLS, components are selected that give 'maximal' reduction in the covariance $X^TY$ of the data; in that sense PLS will give the minimum number of variables that is necessary [Höskuldsson, 1988].

PLS creates orthogonal score vectors (also called latent vectors or components) by maximizing the covariance between different sets of variables [Rosipal & Krämer, 2006]. There are different PLS techniques to extract latent vectors, and each of them gives rise to a variant of the PLS.

The predictor and predicted (response) variables (i.e. *X* and *Y* variables respectively) are each considered as a block of variables. PLS then extracts the score vectors which serve as a new predictor representation and regresses the response variables on these new predictors.

In general, PLS methods are fast, they are characterized by high computational and statistical efficiency [Boulesteix & Strimmer, 2006]. In literature however, there exist a large number of algorithmic variants of PLS, which makes it very difficult to understand the principles underlying PLS. Most of the literature gives a description of (iterative) algorithms for the Partial Least Squares regression, such as in the tutorial of [Geladi & Kowalski, 1986]. Some numerical properties of the PLS regression algorithm are described in [Höskuldsson, 1988]. An alternative approach to PLS regression is described in [De Jong, 1993], which is called the SIMPLS method (a Straightforward IMPlementation of a statistically inspired modification of the PLS method) and calculates the PLS factors directly as a linear combination of the original values.

An overview of recent advances is given in [Rosipal & Krämer, 2006], there are different forms of PLS, the most frequently used are PLS1 (i.e. the dependent variable or response variable *y* is a column vector) and PLS2 (i.e. the dependent variable or response variable *Y* is a matrix). It is shown in [De Jong, 1993] that SIMPLS and PLS1 give the same result, whereas for the case that there are more than one dependent variables the results are slightly different.

Hereafter a description of the algorithm is given, which is based on [Geladi & Kowalski, 1986], [Höskuldsson, 1988], [Ränner et al., 1994], [Rosipal & Krämer, 2006] and [De Jong, 1993], although there are differences between these different approaches. Differences are mainly in different ways of scaling, i.e. normalization, within the algorithm, which makes it difficult to directly compare results within the algorithm.

### A.2.1. PLS centering and scaling

Assumption:

> All variables, both dependent and independent, are assumed to be mean-centered and are assumed to have some kind of scaling. (Ref: [Geladi & Kowalski, 1986]).

Not all algorithms use scaling, in this appendix we apply scaling in the PLS algorithm, i.e. we start with centering and scaling of *X* and *Y*, i.e. by subtracting off

column means and dividing by standard deviation of each column, to get centered and scaled variables $X_0$ and $Y_0$

$$X_0 = \left(X - \mathrm{M}_X\right)\Lambda_X^{-1}$$
$$Y_0 = \left(Y - \mathrm{M}_Y\right)\Lambda_Y^{-1} \qquad \text{(A.5)}$$

where

$\mathrm{M}_X$ is a $n \times m$ matrix with column means as given below

$\Lambda_X$ is a $m \times m$ diagonal matrix with standard deviations of each column

$\mathrm{M}_Y$ is a $n \times p$ matrix with column mean as given below

$\Lambda_Y$ is a $p \times p$ diagonal matrix with standard deviations of each column

with the following notations:

$$X = \begin{pmatrix} x_{1,1} & \cdots & x_{1,m} \\ \vdots & & \vdots \\ x_{n,1} & \cdots & x_{n,m} \end{pmatrix}, \qquad Y = \begin{pmatrix} y_{1,1} & \cdots & y_{1,p} \\ \vdots & & \vdots \\ y_{n,1} & \cdots & y_{n,p} \end{pmatrix}$$

$$M_X = \begin{pmatrix} \overline{x}_1 & \cdots & \overline{x}_m \\ \vdots & & \vdots \\ \overline{x}_1 & \cdots & \overline{x}_m \end{pmatrix}_{n \times m}, \qquad M_Y = \begin{pmatrix} \overline{y}_1 & \cdots & \overline{y}_p \\ \vdots & & \vdots \\ \overline{y}_1 & \cdots & \overline{y}_p \end{pmatrix}_{n \times p}$$

where the mean $\overline{x}_j$ of the *j*-th column is defined as $\overline{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{i,j}$, and similarly for the mean $\overline{y}_j$.

In this case the mean of $X_0 = 0$ and standard deviation of $X_0 = 1$, and similarly for $Y_0$.

### A.2.2 PLS decomposition

As described in [Geladi & Kowalski, 1986] and [Rosipal & Krämer, 2006], PLS decomposes the $n \times m$ matrix of zero-mean variables $X_0$ and the $n \times p$ matrix of zero-mean variables $Y_0$ into two outer relations ($X_0$ and $Y_0$ blocks individually) and an inner relation (linking both blocks, i.e. it is assumed that there is a linear relation between the score vectors *t* and *u*).

Then PLS is applied on $X_0$ and $Y_0$:

First block of variables: $X_0 = T\,P^T + E$ (A.6)

Second block of variables: $Y_0 = U\,S^T + F$ (A.7)

Inner relation: $$U = T D + H \quad \text{(A.8)}$$

where $T$ and $U$ are $n \times a$ matrices of the $a$ extracted score vectors (components, latent[10] vectors), the $m \times a$ matrix $P$ and the $p \times a$ matrix $S$ represent matrices of loadings[11] and the $n \times m$ matrix $E$ and the $n \times p$ matrix $F$ are the matrices of residuals. Matrix $D$ is an $a \times a$ diagonal matrix and $H$ denotes the $n \times a$ matrix of residuals.

The classical PLS method is based on the nonlinear iterative partial least squares (NIPALS) algorithm [Rosipal & Krämer, 2006], and finds weight vectors $w$ and $q$ such that

$$\left[\mathrm{Cov}(t,u)\right]^2 = \left[\mathrm{Cov}(X_0 w, Y_0 s)\right]^2 = \max_{|r|=|v|=1} \left[\mathrm{Cov}(X_0 r, Y_0 v)\right]^2$$

where

$$\mathrm{Cov}(t,u) = t^T u / n$$

denotes the sample covariance between the score vectors $t$ and $u$ , and $n$ the number of samples.

Substitution of (A.8) into (A.7) yields that

$$Y_0 = T C^T + F^* \quad \text{(A.11)}$$

where $C^T$ denotes the $a \times p$ matrix of regression coefficients, and $F^*$ the $n \times p$ residual matrix, where

$$C^T = D S^T \quad \text{(A.12)}$$

and

$$F^* = H S^T + F \quad \text{(A.13)}$$

Equation (A.11) is simply the decomposition of $Y_0$ using ordinary least squares regression with orthogonal predictors $T$.

***Least squares solutions***

Note that the least squares solutions for (A.6) and (A.11) are:

[10] **Latent variables** replace the original variables by a smaller number of 'underlying' variables.

[11] **Loading vectors** are the estimated weights which are to be applied to the variables when fitting the bilinear relationship between the Y and X variables. (Ref: http://www.bioss.ac.uk/smart/unix/mplsgxe/slides/glossary.htm)

$$\hat{P} = X_0^T T \left(T^T T\right)^{-1}$$
$$\hat{C} = Y_0^T T \left(T^T T\right)^{-1} \quad \text{(A.14)}$$

### A2.2.1 Iterative process

PLS is an iterative process which calculates a set of $a$ orthogonal $X$ block factors scores $T = [t_1, t_2, \ldots, t_a]$ companion $Y$ block factor scores $U = [u_1, u_2, \ldots, u_a]$ factor by factor. The first PLS factors $t_1$ and $u_1$ are weighted sums of the centered variables: $t_1 = X_0 w_1$ and $u_1 = Y_0 s_1$ respectively. Usually the weights are determined via the NIPALS algorithm. This is an iterative sequence which starts by choosing for $u_1$ some column of $Y_0$, e.g., the one having maximum variance. The iterative sequence then is:

$$w_h \propto X_{h-1}^T u_h$$
$$t_h = X_{h-1} w_h$$
$$s_h \propto Y_{h-1}^T t_h$$
$$u_h = Y_{h-1} s_h \quad \text{(A.15)}$$

and stops when $w_h$ or $t_h$ do not change given some pre-specified tolerance.
The symbol $\propto$ not only denotes proportionality, but also a subsequent normalization of the resultant vector. Thus the weight vectors $w_h$ and $s_h$ have length 1. (Different normalizations are possible, the specific choice being rather a matter of habit or of convenience). See pages 252-254 in [De Jong, 1993], for a more detailed description, see Appendix B.

Once the $X$ block factor $t_h$ is obtained one proceeds with deflating the data matrices. This yields new data sets $X_h$ and $Y_h$ which are the matrices of residuals obtained after regressing all variables on $t_h$

$$X_h = X_{h-1} - t_h \left(t_h^T X_{h-1}\right) / \left(t_h^T t_h\right)$$
$$Y_h = Y_{h-1} - t_h \left(t_h^T Y_{h-1}\right) / \left(t_h^T t_h\right) \quad \text{(A.16)}$$

These equations can be written as

$$X_h = X_{h-1} - t_h p_h^T$$
$$Y_h = Y_{h-1} - b_h t_h s_h^T \equiv Y_{h-1} - t_h c_h^T \quad \text{(A.17)}$$

where $p_h$ represents the vector of loading of factor $t_h$ on the $X$ variables, scalar $b_h$ is the estimated regression coefficient for the so-called inner relation between the two data sets relayed via their latent variables

$$\begin{aligned} p_h &= X_{h-1}^T t_h / \left( t_h^T t_h \right) \\ \hat{u}_h &= b_h t_h \\ b_h &= u_h^T t_h / \left( t_h^T t_h \right) \end{aligned} \tag{A.18}$$

After extraction of the $a$ components, matrices $T$, $U$, $S$, $P$ and $W$ are created consisting of the columns created by the vectors extracted during the individual iterations, i.e.

$$\begin{aligned} T &= \left[ t_1, t_2, \ldots, t_a \right]_{n \times a} \\ U &= \left[ u_1, u_2, \ldots, u_a \right]_{n \times a} \\ S &= \left[ s_1, s_2, \ldots, s_a \right]_{p \times a} \\ P &= \left[ p_1, p_2, \ldots, p_a \right]_{m \times a} \\ W &= \left[ w_1, w_2, \ldots, w_a \right]_{m \times a} \end{aligned} \tag{A.19}$$

and the (estimated) regression coefficient $b_h$ are saved in the diagonal matrix $D$

$$D = \mathrm{diag}(b) = \mathrm{diag}\left[ b_1, b_2, \ldots, b_a \right]_{a \times a} \tag{A.20}$$

***Weight vectors q and c***

Note that for example [Höskuldsson, 1988] and [Ränner et al., 1994] use the weight vector $c_h$, while others use weight $s_h$ instead (see for example [Geladi & Kowalski, 1986] and [De Jong, 1993]) in their algorithm, both vectors are related as

$$c_h^T \equiv b_h s_h^T \tag{A.21}$$

where the weight vector $c$ is not scaled to unit norm; while as stated before the weight vectors $w_h$ and $s_h$ have length 1.

After extraction of the $a$ components, matrix $C$ is created to consist of the columns created by the vectors extracted during the individual iterations, i.e.

$$C = \left[ c_1, c_2, \ldots, c_a \right]_{p \times a} \tag{A.22}$$

In matrix form this Equation (A.20) equals Equation (A.12), i.e.

$$C^T = D S^T \tag{A.12}$$

A PLS algorithm based on Nonlinear Iterative Partial Least Squares (NIPALS) is described in more detail in Appendix B.

### A.2.2.2 Number of latent components

For the number of factors *a*, i.e. the number of latent components, used in PLS with a matrix $X$ of size $n \times m$ the maximal value can be chosen, i.e. $a_{\max} \leq \text{rank}(X) \leq \min(n, m)$, but in [Boulesteix & Strimmer, 2005] it is described that with PLS it is desirable to choose as small a value of *a* as possible without sacrificing too much predictive power. One straightforward statistical procedure to estimate this minimum value $a_{\min}$ is the method of cross-validation, which is described on page 9 in (Boulesteix & Strimmer, 2005). For example a default value can be $a = \min(n-1, m)$ (See also plsregress.m in Matlab).

### A.2.2.3 Properties of matrices

(Orhtogonality) properties of the PLS factors (e.g. column vectors in the iterative algorithm) are described in [Geladi & Kowalski, 1986], [Höskuldsson, 1988] and [Ränner et al., 1994].

The properties are summarized as follows:

- The vectors $p_i$ and $s_i$ have unit length for each $i = 1,2,..., a$
- The vectors $t_i$ and $u_i$ are centred around zero for each $i = 1,2,..., a$
- The vectors $w_i$ are mutually orthogonal: $w_i^T w_j = 0$ for $i \neq j$
  (in other words $W^T W$ is a diagonal matrix)
- The vectors $t_i$ are mutually orthogonal: $t_i^T t_j = 0$ for $i \neq j$
  (in other words $T^T T$ is a diagonal matrix)
- The vectors $w_i$ are orthogonal to the vectors $p_j$ for $i<j$ : $w_i^T p_j = 0$ for $i < j$
  (in other words the matrix $W^T P$ is a lower triangular matrix)
- The vectors $t_i$ are orthogonal to the vectors $u_j$ for $i<j$ : $t_i^T u_j = 0$ for $i> j$
  (in other words the matrix $T^T U$ is a lower triangular matrix)
- The vectors $p_i$ are orthogonal in the kernel space of X: $p_i^T (X^T X)^- p_j = 0$ for $i \neq j$
  (where $A^-$ denotes the generalized inverse of a matrix A)
- No special orthogonality properties are available among the vectors $u_i$ , $c_i$ and $q_i$; Though these vectors satisfy some orthogonality conditions relative to some matrices.

### A.2.2.4 Regression coefficient matrix for the centred and scaled variables $X_0$ and $Y_0$

The score matrix $T$ can be computed from the original $X_0$ as follows, see [De Jong, 1993] and [Helland, 1988]

$$T = X_0 R$$

where $P$ , $R$ and $W$ have the following relation

$$\begin{aligned} R &= W\left(P^T W\right)^{-1} \\ P^T R &= R^T P = W^T W = I \end{aligned} \tag{A.23}$$

where $W$ is a weight $m \times a$ matrix, and $R$ an alternative weight $m \times a$ matrix, both share the same column space.

In [Ränner et al., 1994] it is shown that

$$W = X_0^T U \tag{A.24}$$

Substituting (A.23) into Equation (A.11) gives

$$\begin{aligned} Y_0 &= T C^T + F^* \\ &= X_0 R C^T + F^* \\ &= X_0 W\left(P^T W\right)^{-1} C^T + F^* \\ &= X_0 \hat{B}_{PLS} + F^* \end{aligned}$$

where $\hat{B}_{PLS}$ is the $m \times p$ regression coefficient matrix

$$\hat{B}_{PLS} = W\left(P^T W\right)^{-1} C^T$$

Then it follows that

$$Y_0 = X_0 \hat{B}_{PLS} + F^* \quad \text{with estimator} \quad \hat{B}_{PLS} = W\left(P^T W\right)^{-1} C^T \tag{A.25}$$

### A.2.3 Back to original variables

In terms of the original (unscaled and uncentered) matrices $X$ and $Y$ it follows that

$$Y = \hat{B}^{*}_{PLS_0} + X\ \hat{B}^{*}_{PLS} \tag{A.26}$$

where $\hat{B}^{*}_{PLS}$ is the regression matrix computed from the data and $\hat{B}^{*}_{PLS_0}$ is an intercept term (i.e. the regression coefficient for the intercept)

$$\begin{aligned} \hat{B}^{*}_{PLS} &= \Lambda_X^{-1}\ \hat{B}_{PLS}\ \Lambda_Y \\ \hat{B}^{*}_{PLS_0} &= \mathrm{M}_Y - \mathrm{M}_X\ \Lambda_X^{-1}\ \hat{B}_{PLS}\ \Lambda_Y \end{aligned} \tag{A.27}$$

*Proof:* Substitute

$$\begin{aligned} X_0 &= \left(X - \mathrm{M}_X\right)\Lambda_X^{-1} \\ Y_0 &= \left(Y - \mathrm{M}_Y\right)\Lambda_Y^{-1} \end{aligned}$$

into (A.25) yields

$$\left(Y - \mathrm{M}_Y\right)\Lambda_Y^{-1} = \left(X - \mathrm{M}_X\right)\Lambda_X^{-1}\hat{B}_{PLS} + F^{*}$$

that is

$$\begin{aligned} Y &= \mathrm{M}_Y + \left(X - \mathrm{M}_X\right)\Lambda_X^{-1}\hat{B}_{PLS}\Lambda_Y + F^{*}\Lambda_Y \\ &= \mathrm{M}_Y + X\Lambda_X^{-1}\hat{B}_{PLS}\Lambda_Y - \mathrm{M}_X\Lambda_X^{-1}\hat{B}_{PLS}\Lambda_Y + F^{*}\Lambda_Y \\ &= X\left(\Lambda_X^{-1}\hat{B}_{PLS}\Lambda_Y\right) + \mathrm{M}_Y - \mathrm{M}_X\left(\Lambda_X^{-1}\hat{B}_{PLS}\Lambda_Y\right) + F^{*}\Lambda_Y \\ &= X\ \hat{B}^{*}_{PLS} + \hat{B}^{*}_{PLS_0} \end{aligned}$$

where $\hat{B}^{*}_{PLS}$ is the regression matrix computed from the data

$$\begin{aligned} \hat{B}^{*}_{PLS} &= \left(\sigma_X\right)^{-1}\hat{B}_{PLS}\,\sigma_Y \\ \hat{B}^{*}_{PLS} &= \Lambda_X^{-1}\hat{B}_{PLS}\,\Lambda_Y \end{aligned}$$

and $\hat{B}^{*}_{PLS_0}$ is an intercept term (i.e. the regression coefficient for the intercept)

$$\begin{aligned} \hat{B}^{*}_{PLS_0} &= \mathrm{M}_Y - \mathrm{M}_X\left(\Lambda_X^{-1}\hat{B}_{PLS}\,\Lambda_Y\right) + F^{*}\Lambda_Y \\ &= \mathrm{M}_Y - \mathrm{M}_X\hat{B}^{*}_{PLS} + F^{*}\Lambda_Y \end{aligned}$$

■

# Appendix B PLS algorithms

As explained in Subsection 3.5, there are different PLS algorithms as addressed for example in [De Jong, 1993] and [Rosipal & Krämer, 2006]. Appendix B.1 describes a PLS algorithm based on NIPALS and Appendix B.2 describes a PLS algorithm based on SIMPLS.

## B.1 PLS algorithm based on NIPALS

This appendix describes a PLS algorithm based on Nonlinear Iterative Partial Least Squares (NIPALS), and is a more detailed description than the iterative process as described in Appendix A.2.2.1.

It is assumed that $X_0$ and $Y_0$ are mean-centered and scaled, i.e. at least in [Geladi & Kowalski, 1986]. In [Höskuldsson, 1988] the matrices may be scaled or centered, where scaling can correspond to working with correlation matrices, and centering to subtracting mean values from each of the column values.

*For each component h=1,2,..., a:*

(1) Take $u_1$ = some $y_j$

Alternatives for $u_{start}$ in literature:

- In [Geladi & Kowalski, 1986], $u_1$ = some $y_j$
- In [Höskuldsson, 1988], $u_1$ = first column of $Y$.
- In [De Jong, 1993], $u_1$ = column of $Y$ having maximum variance

*In X-block:*

(2) $w_h = X_{h-1}^T u_h / \left(u_h^T u_h\right)$

(as in [Geladi & Kowalski, 1986], [Höskuldsson, 1988], [Rosipal & Krämer, 2006] and [Ränner et al, 1994])

(3) $w_h = w_h / \|w_h\|$ (normalization, i.e. scale vector $w_h$ to be of length 1 $\|w_h\| \to 1$**)**, i.e.

(4) $t_h = X_{h-1} w_h / \left(w_h^T w_h\right)$ (as in [Geladi & Kowalski, 1986]), or

$t_h = X_{h-1} w_h$ (as in [Höskuldsson, 1988] and [Rosipal & Krämer, 2006])

*In Y-block:*

(5) $s_h = Y_{h-1}^T t_h / \left(t_h^T t_h\right)$

(as in [Geladi & Kowalski, 1986], [Höskuldsson, 1988] and [Rosipal & Krämer, 2006])

(6) $s_h = s_h / \|s_h\|$ (normalization, i.e. scale vector $s_h$ to be of length 1 $\|s_h\| \to 1$**)**

(7) $u_h = Y_{h-1}\, s_h / \left(s_h^T\, s_h\right)$ (as in [Geladi & Kowalski, 1986]; and in [Höskuldsson, 1988] and [Ränner et al, 1994] but with $c_h$ instead of $s_h$), or

$u_h = Y_{h-1}\, s_h$ **(**as in [Rosipal & Krämer, 2006] but with $c_h$ instead of $s_h$ )

(8) Compare *t* in step 4 with *t* in preceeding iteration, if they are equal (within certain rounding error), then go to step 9, else go to step 2. (If *Y* = *y*, then 5-8 can be omitted, and set *s* = 1)

*After convergence, calculate X loadings and rescale scores and weight accordingly*

(9) $p_h = X_{h-1}^T\, t_h / \left(t_h^T\, t_h\right)$

*Ad step (9):*

In addition in [Höskuldsson, 1988] and [Rosipal & Krämer, 2006] also the *Y* loadings are calculated explicitly:

$s_h = Y_{h-1}^T\, u_h / \left(u_h^T\, u_h\right)$

(10) $\left\| \left(p_h^{old}\right)^T \right\| = \| p_h^T \|$ and $p_h^T = p_h^T / \left\| \left(p_h^{old}\right)^T \right\|$

(normalization, i.e. scale vector $p_h$ to be of length 1 $\|p_h\| \to 1$**)**

(as in in [Geladi & Kowalski, 1986])

(this step not in [Höskuldsson, 1988], [Rosipal & Krämer, 2006] and [Ränner et al, 1994])

(11) $t_h = t_h \left\| \left(p_h^{old}\right)^T \right\|$

(as in in [Geladi & Kowalski, 1986])

(this step not in [Höskuldsson, 1988], [Rosipal & Krämer, 2006] and [Ränner et al, 1994])

(12) $w_h^T = w_h^T \left\| \left(p_h^{old}\right)^T \right\|$

(as in in [Geladi & Kowalski, 1986])

(this step not in [Höskuldsson, 1988], [Rosipal & Krämer, 2006] and [Ränner et al, 1994])

*Steps (10)-(12):*
These steps form a scaling procedure in [Geladi & Kowalski, 1986] for obtaining "orthogonal *t* values", but are "not absolutely necessary" (See page 11 in [Manne, 1987]).
These steps are not used in the PLS algorithms as described in [Höskuldsson, 1988], [Rosipal & Krämer, 2006] and [Ränner et al, 1994].

(Estimated) regression coefficient $b_h$ for the inner relation between the two data sets is

(13) $b = u^T t / \left(t^T t\right)$ i.e., $b_h = u_h^T t_h / \left(t_h^T t_h\right)$

After extraction of the $a$ components, matrices $T$, $U$, $S$, $P$ and $W$ are created consisting of the columns created by the vectors extracted during the individual iterations, i.e.

$$T = \left[t_1, t_2, \ldots, t_a\right]_{n \times a}$$
$$U = \left[u_1, u_2, \ldots, u_a\right]_{n \times a}$$
$$S = \left[s_1, s_2, \ldots, s_a\right]_{p \times a}$$
$$P = \left[p_1, p_2, \ldots, p_a\right]_{m \times a}$$
$$W = \left[w_1, w_2, \ldots, w_a\right]_{m \times a}$$

and the (estimated) regression coefficient $b_h$ are saved in the diagonal matrix $D$

$$D = \mathrm{diag}(b) = \mathrm{diag}\left[b_1, b_2, \ldots, b_a\right]_{a \times a}$$

Calculation of residuals (general outer relation for the $X$ block and mixed relation for $Y$ block) for component $h$, i.e., once the $X$ block factor $t_h$ is obtained one proceeds with deflating the data matrices. This yields new data sets $X_h$ and $Y_h$ which are the matrices of residuals obtained after regressing all variables on $t_h$

$$X_h = X_{h-1} - t_h \left(t_h^T X_{h-1}\right) / \left(t_h^T t_h\right)$$
$$Y_h = Y_{h-1} - t_h \left(t_h^T Y_{h-1}\right) / \left(t_h^T t_h\right)$$

These equations can be written as

$$X_h = X_{h-1} - t_h p_h^T$$
$$Y_h = Y_{h-1} - t_h c_h^T = Y_{h-1} - b_h t_h s_h^T$$

with centered and scaled variables $X_0$ and $Y_0$ for $h = 1$.

***Remarks:***

The components *t* and *u* in the PLS algorithm have the interpretation that they are components in *X* and *Y* space. The PLS algorithm selects one pair of components at a time, because the covariance of the second pair is smaller than the maximal covariance at the next iteration (Page 218 in [Höskuldsson, 1988]).

There are many equivalent ways of scaling. Scores *t* can be normalized in the algorithm, but one can also choose to introduce normalization at another point in the algorithm. This makes it difficult to directly compare the scores and the loadings of different PLS regression implementations [Mevik & Wehrens, 2007]

[Geladi & Kowalski, 1986] describes a few methods to determine the number of components *a*. If the underlying model for the relation between *X* and *Y* is a linear model, the number of components needed to describe this model is equal to the model dimensionality (i.e. rank (*X*)).

### B.2 PLS algorithm based on SIMPLS

This appendix describes a PLS algorithm based on SIMPLS as described in [De Jong, 1993], where SIMPLS stands for 'Straightforward IMPlementation of a statistically inspired modification of the PLS method'.

Input to the SIMPLS algorithm is:

$X$ is an $n \times m$ matrix

$Y$ is an $n \times p$ matrix

$a$ is the number of factors

The PLS algorithm first starts with centering of $X$ and $Y$, i.e. by subtracting off column means to get centered variables $X_0$ and $Y_0$ as in Equation (27).

Define the cross-product $A_0 = X_0^T Y_0$.

*For each component h=1,2,..., a:*

(1) Apply Singular Value Decomposition (SVD) to matrix $A_{h-1}$, such that

$A_{h-1} = R_h Q_h C_h^T$

$r_h$ = weight vector is the first column of $R_h$, i.e. first left singular vector from the SVD

$c_h$ = first column of $C_h$, i.e. first right singular vector from the SVD

$q_h$ = $h^{\text{th}}$ element of $Q_h$

(2) Compute X-score vector $t_h$:

$t_h = X_0 r_h$ (see Equation (24) in [De Jong, 1993])

$t_h = t_h / \|t_h\|$ (normalization, i.e. scale vector $t_h$ to be of length 1 $\|t_h\| \to 1$)

(3) Compute X-loading vector $p_h$:

$p_h = X_0^T t_h$

(4) Compute weight vector $q_h$:

$s_h = q_h c_h / \|t_h\|$, that is $s_h = Y_0^T t_h$

(5) Compute Y-score vector $u_h$:

$u_h = Y_0 s_h$, that is $u_h = Y_0 Y_0^T t_h$ (see Equation (26) in [De Jong, 1993])

(6) Compute weight vector $w_h$:

$w_h = r_h / \|t_h\|$

(7) Orthonormal basis vector $v_h$ obtained from modified Gram-Schmidt repeated twice:

$v_h = p_h$ (initialize orthogonal loading vector)

if $h > 1$ , then

$v_h = v_h - V_h \left( V_h^T p_h \right)$ (make $v \perp$ to previous loadings)

$u_h = u_h - T_h \left( T_h^T u_h \right)$ (make $u \perp$ to previous $t$ values)

end

$v_h = v_h / \left\| v_h \right\|$ (normalize orthogonal loadings)

(8) $A_h = A_{h-1} - v_h \left( v_h^T A_{h-1} \right)$ (see Equation (34) in [De Jong, 1993]), with $A_1 = A_0$

After extraction of the $a$ components, matrices $R$, $T$, $P$, $S$, $U$ and $V$ are created consisting of the columns created by the vectors extracted during the individual iterations, i.e.

$$R = [r_1, r_2, \ldots, r_a]_{m \times a}$$
$$T = [t_1, t_2, \ldots, t_a]_{n \times a}$$
$$P = [p_1, p_2, \ldots, p_a]_{m \times a}$$
$$S = [s_1, s_2, \ldots, s_a]_{p \times a}$$
$$U = [u_1, u_2, \ldots, u_a]_{n \times a}$$
$$V = [v_1, v_2, \ldots, v_a]_{m \times a}$$

It is further shown that the $m \times p$ regression coefficient matrix $\hat{B}_{PLS}$ is (see Equation (37) in [De Jong, 1993])

$$\hat{B}_{PLS} = R S^T$$

and the regression coefficient for the intercept is

$$\hat{B}^*_{PLS_0} = \mathrm{M}_Y - \mathrm{M}_X \ \hat{B}_{PLS}$$

# Appendix C List of notations and symbols

| | |
|---|---|
| $a$ | Number of factors used in PLS, i.e. number of extracted score vectors (or components or latent vectors) (PLS; Section 3.5) |
| $A_0$ | Cross product $A_0 = X_0^T Y_0$ of size $n_p \times 1$ (PLS; Section 3.5) |
| $A_h$ | Deflated product matrices for $h = 1, 2, 3, \ldots, a$ (PLS; Section 3.5) |
| $A^+$ | Moore-Penrose or pseudoinverse $A^+$ of an $m \times n$ matrix $A$ |
| $b$ | Bias vector $b \triangleq \mathrm{Col}\left(b_1, b_2, \ldots, b_{n_p}\right)$ |
| $b_j$ | Bias parameter for $j = 1, 2, \ldots, n_p$ in bias vector $b$ |
| $\chi_q$ | Change of factor in $\ln q_k$ (Section 2) |
| $\chi_G$ | Change of factor in $\ln \hat{G}(q_k)$ (Section 2) |
| $Cov(\cdot)$ | Covariance |
| $D$ | Diagonal matrix of size $a \times a$ (PLS; Section 3.5) |
| $\Delta_G$ | Change in collision risk $G$ (Section 2) |
| $\Delta_q$ | Change in the value of parameter $q$ (Section 2) |
| $\varepsilon$ | Random error (Section 2) |
| $\varepsilon_k$ | Random error of $k$-th IPS run for $k = 1, .., K$ (Section 2) |
| $E_X$ | Matrix of residuals of size $K \times n_p$ (PLS; Section 3.5) |
| $E_Y$ | Vector of residuals of size $K \times 1$ (PLS; Section 3.5) |
| $\| . \|$ | Euclidean norm (or 2-norm) |
| $f_0$ | Intercept term |
| $\hat{f}_0$ | Estimate of the intercept term |
| $f_j$ | Parameter in vector $F$ for $j = 1, 2, \ldots, n_p$ |
| $\hat{f}_j$ | Estimate of parameter $f_j$ for $j = 1, 2, \ldots, n_p$ |
| $F$ | Vector with model parameters or coefficients (also called regression coefficients) of size $n_p \times 1$; $F \triangleq \mathrm{Col}\left(f_1, f_2, \ldots, f_{n_p}\right)$ that is $F^T = \begin{pmatrix} f_1 & f_2 & \cdots & f_{n_p} \end{pmatrix}$ or $F = \begin{pmatrix} f_1 \\ \vdots \\ f_{n_p} \end{pmatrix}_{n_p \times 1}$ |
| $\tilde{F}$ | Vector with intercept term and model parameters or coefficients (also called regression coefficients) of size $(n_p + 1) \times 1$; $\tilde{F} \triangleq \mathrm{Col}\left(f_0, f_1, \ldots, f_{n_p}\right)$ that is $\tilde{F}^T = \begin{pmatrix} f_0 & f_1 & \cdots & f_{n_p} \end{pmatrix}$ or $\tilde{F}^T = \begin{bmatrix} f_0 & F^T \end{bmatrix}$ |

| | |
|---|---|
| $\hat{F}$ | Least squares estimator of vector $F$ of size $n_p \times 1$ $\hat{F}^T = \begin{pmatrix} \hat{f}_1 & \hat{f}_2 & \cdots & \hat{f}_{n_p} \end{pmatrix}$ |
| $\hat{\tilde{F}}$ | Least squares estimator of the extended regression vector $\tilde{F}$ of size $(n_p+1)\times 1$ ; defined as $\hat{\tilde{F}}^T = \begin{bmatrix} \hat{f}_0 & \hat{F}^T \end{bmatrix} = \begin{pmatrix} \hat{f}_0 & \hat{f}_1 & \cdots & \hat{f}_{n_p} \end{pmatrix}$ |
| $F_0$ | Intercept vector $F_0$ of size $K \times 1$; defined as $F_0 \triangleq \mathrm{Col}(f_0, f_0, \ldots, f_0)$ or $F_0 = \begin{pmatrix} f_0 \\ \vdots \\ f_0 \end{pmatrix}_{K\times 1}$ |
| $\hat{F}_{PLS}$ | PLS estimator of the vector $F$ of size $n_p \times 1$ (PLS; Section 3.5) |
| $\hat{F}_{PLS_0}$ | PLS estimator of the intercept vector $F_0$ of size $K \times 1$ (PLS; Section 3.5) |
| $G(q)$ | Collision risk as a function of *q* (Section 2) |
| $\frac{\partial G(q)}{\partial q}$ | Partial derivative, $n$-dimensional vector (Section 2) |
| $\tilde{G}$ | IPS computed output value of the collision risk (Section 2) |
| $\tilde{G}_k$ | Computed output value of *k*-th IPS run for $k = 1,..,K$ (Section 2) |
| $\hat{G}(q)$ | Estimation of collision risk (with a meta-model approach) (Section 2) |
| $H$ | Matrix of residuals of size $K \times a$ (PLS; Section 3.5) |
| $H^*$ | $H^*$ is the Hermitian transpose (also called conjugate transpose) of a matrix $H$ |
| $I_{K\times K}$ | Identity matrix of size $K \times K$ |
| $j$ | Index number $j = 1, 2, \ldots, n_p$ |
| $\mathbf{j}_{K\times 1}$ | Vector of ones of size $K \times 1$ |
| $J_{K\times K}$ | Matrix of ones of size $K \times K$ , i.e. $J_{K\times K} = \mathbf{j}_{K\times 1}\, \mathbf{j}_{1\times K}$ |
| $k$ | Index number $k = 1, 2, \ldots, K$ |
| $K$ | Base number of samples |
| $\kappa(\cdot)$ | Condition number of a square matrix $A$ , defined as $\kappa(A) = \|A\| \cdot \|A^{-1}\|$ for a given matrix norm (e.g. *p*-norm, Frobenius norm) |
| $\ell$ | Uncertainty vector $\ell \triangleq \mathrm{Col}(\ell_1, \ell_2, \ldots, \ell_{n_p})$ |
| $\ell_j$ | Uncertainty parameter for $j = 1, 2, \ldots, n_p$ in uncertainty vector $\ell$ |
| $M_X$ | $M_X$ is an $K \times n_p$ matrix of column means of design matrix $X$ $M_X = \begin{pmatrix} \overline{x}_1 & \cdots & \overline{x}_{n_p} \\ \vdots & & \vdots \\ \overline{x}_1 & \cdots & \overline{x}_{n_p} \end{pmatrix}_{K\times n_p}$ |

| | |
|---|---|
| $M_Y$ | $M_Y$ is an $K\times 1$ vector of column means of vector $Y$ $M_Y = \begin{pmatrix} \bar{y} \\ \vdots \\ \bar{y} \end{pmatrix}_{K\times 1}$ |
| $\mu(\cdot)$ | Mean |
| $\mu_w$ | Mean value |
| $N$ | Number of Monte Carlo simulations runs |
| $N\{\cdot;\mu_w,\ \sigma_w{}^2\}$ | Normal (or Gaussian) distribution with mean $\mu_w$ and variance $\sigma_w{}^2$ |
| $n$ | Number of scalar parameters in vector $q$ (Section 2) |
| $n_p$ | Number of scalar parameters in vector $F$ |
| $p_h$ | Loading vector of size $n_p\times 1$ for $h=1,2,3,\ldots,a$, which is extracted from a given matrix $X$ (PLS; Section 3.5) |
| $P$ | Loading matrix of size $n_p\times a$, consists of $a$ extracted loading vectors $P=[p_1,p_2,\ldots,p_a]_{n_p\times a}$ (PLS; Section 3.5) |
| $\propto$ | Proportionality and subsequent normalization of the resultant vector |
| $q$ | Parameter $q$, $n$-dimensional (Section 2) |
| $q^*$ | Specific (local) working point $q^*$ (Section 2) |
| $\{q_k,\tilde{G}_k;\ k=1,..,K\}$ | Input-output samples of $k$-th IPS run for $k=1,..,K$ (Section 2) |
| $q_k \sim p_{q_k}(.)$ | Input sample of $k$-th IPS run $q_k \sim p_{q_k}(.)$ (Section 2) |
| $r_h$ | Weight vector of size $n_p\times 1$ for $h=1,2,3,\ldots,a$ (PLS; Section 3.5) |
| $R$ | Weight matrix of size $n_p\times a$ (also referred to as alternative weight matrix, as opposed to a weight matrix in the NIPALS algorithm) consists of $a$ weight vectors $R=[r_1,r_2,\ldots,r_a]_{n_p\times a}$ (PLS; Section 3.5) |
| $s_h$ | Loading scalar for $h=1,2,3,\ldots,a$, which is extracted from a given vector $Y$ (PLS; Section 3.5) |
| $S$ | Loading matrix of size $1\times a$, consists of $a$ extracted loading scalars $S=[s_1,s_2,\ldots,s_a]_{1\times a}$ (PLS; Section 3.5) |
| $\sigma(\cdot)$ | Standard deviation |
| $\sigma(\hat{f}_j)\sqrt{K}/\sigma_w$ | Normalized standard deviation of $\hat{f}_j$ for $j=0,1,2,\ldots,n_p$ |
| $\sigma_G$ | Standard deviation $\sigma_G$ of collision risk $G$ (Section 2) |
| $\sigma_w$ | Standard deviation value |
| $\Sigma_q$ | Covariance of parameter $q$ (Section 2) |
| $t_h$ | Orthogonal vector of size $K\times 1$ for $h=1,2,3,\ldots,a$, which is extracted from a given matrix $X$ (PLS; Section 3.5) |
| $T$ | Score matrix of size $K\times a$, consists of $a$ extracted score vectors $T=[t_1,t_2,\ldots,t_a]_{K\times a}$ (PLS; Section 3.5) |

$u_h$ Vector of size $K \times 1$ for $h = 1, 2, 3, \ldots, a$, which is extracted from a given vector $Y$ (PLS; Section 3.5)

$U$ Score matrix of size $K \times a$, consists of $a$ extracted score vectors $U = [u_1, u_2, \ldots, u_a]_{K \times a}$ (PLS; Section 3.5)

$v_h$ Vector of size $n_p \times 1$ for $h = 1, 2, 3, \ldots, a$ (PLS; Section 3.5)

$V$ Matrix of size $n_p \times a$ which consists of vectors $v_h$, i.e. $V = [v_1, v_2, \ldots, v_a]_{n_p \times a}$ ; $[v_1, v_2, \ldots, v_a]$ represents an orthonormal basis of $[p_1, p_2, \ldots, p_a]$ for $X$-loading vectors $p_h$ (PLS; Section 3.5)

$Var(\cdot)$ Variance

$w_k$ Random noise variable for $k = 1, 2, \ldots, K$

$W$ Column vector of size $K \times 1$ ; $W \triangleq \mathrm{Col}(w_1, w_2, \ldots, w_K)$, that is $W^T = (w_1 \quad w_2 \quad \cdots \quad w_K)$ or $W = \begin{pmatrix} w_1 \\ \vdots \\ w_K \end{pmatrix}_{K \times 1}$

$x_k$ Random variable of size $n_p$ (in Section 2 of size $n$) for $k = 1, 2, \ldots, K$; $x_k \triangleq \mathrm{Col}(x_{k,1}, x_{k,2}, \ldots, x_{k,n_p})$ that is $x_k^T = (x_{k,1} \quad x_{k,2} \quad \cdots \quad x_{k,n_p})$ or $x_k = \begin{pmatrix} x_{k,1} \\ \vdots \\ x_{k,n_p} \end{pmatrix}_{n_p \times 1}$

$\tilde{x}_k$ Random variables of size $(n_p + 1)$ for $k = 1, 2, \ldots, K$;
$\tilde{x}_k \triangleq \mathrm{Col}(1, x_{k,1}, \ldots, x_{k,n_p})$ that is $\tilde{x}_k^T = (1 \quad x_{k,1} \quad \cdots \quad x_{k,n_p})$ or $\tilde{x}_k^T = [1 \quad x_k]$

$x_{k,j}$ Random variable in matrix $X$ for $k = 1, 2, \ldots, K$ and $j = 1, 2, \ldots, n_p$

$\bar{x}_j$ Mean of the $j$-th column of matrix $X$, i.e. $\bar{x}_j = \frac{1}{K} \sum_{k=1}^{K} x_{k,j}$

$X$ Design matrix $X$ of size $K \times n_p$ ;
$X \triangleq \mathrm{Col}(x_1^T, x_2^T, \ldots, x_K^T)$ that is $X^T = (x_1^T \quad x_2^T \quad \cdots \quad x_K^T)$ or
$X = \begin{pmatrix} x_{1,1} & \cdots & x_{1,n_p} \\ \vdots & & \vdots \\ x_{K,1} & \cdots & x_{K,n_p} \end{pmatrix}_{K \times n_p} = \begin{pmatrix} x_1^T \\ \vdots \\ x_K^T \end{pmatrix}_{K \times n_p}$

$X_c$ Centered matrix of design matrix $X$ of zero-mean variables of size $K \times n_p$ , i.e. $X_c = X - M_X$

$X_0$ Matrix of size $K \times n_p$ , i.e. $X_0 = \left( I_{K\times K} - \frac{1}{K} J_{K\times K} \right) X$ or

$$X_0 = \begin{pmatrix} x_{1,1} - \overline{x}_1 & \cdots & x_{1,n_p} - \overline{x}_{n_p} \\ \vdots & & \vdots \\ x_{K,1} - \overline{x}_1 & \cdots & x_{K,n_p} - \overline{x}_{n_p} \end{pmatrix}$$

$\tilde{X}$ Design matrix $\tilde{X} = \left[ \mathbf{j}_{K\times 1} \quad X \right]$ of size $K \times \left( n_p + 1 \right)$, i.e.
$\tilde{X} \triangleq \text{Col}\left( \tilde{x}_1^T, \tilde{x}_2^T, \ldots, \tilde{x}_K^T \right)$ that is $\tilde{X}^T = \left( \tilde{x}_1^T \quad \tilde{x}_2^T \quad \cdots \quad \tilde{x}_K^T \right)$ or

$$\tilde{X} = \begin{pmatrix} 1 & x_{1,1} & \cdots & x_{1,n_p} \\ \vdots & \vdots & & \vdots \\ 1 & x_{K,1} & \cdots & x_{K,n_p} \end{pmatrix}_{K\times(n_p+1)} = \begin{pmatrix} \tilde{x}_1^T \\ \vdots \\ \tilde{x}_K^T \end{pmatrix}_{K\times(n_p+1)}$$

$y_k$ Output values or known realisations for $k = 1, 2, \ldots, K$

$\overline{y}$ Mean of column vector $Y$, i.e. $\overline{y} = \frac{1}{K} \sum_{k=1}^{K} y_k$

$Y$ Column vector $Y$ of size $K \times 1$ ;
$Y \triangleq \text{Col}\left( y_1, y_2, \ldots, y_K \right)$, that is $Y^T = \left( y_1 \quad y_2 \quad \cdots \quad y_K \right)$ or

$$Y = \begin{pmatrix} y_1 \\ \vdots \\ y_K \end{pmatrix}_{K\times 1}$$

$Y_c$ Centered column vector of vector $Y$ of zero-mean variables of size $K \times 1$, i.e. $Y_c = Y - M_Y$

$Y_0$ Column vector of size $K \times 1$, i.e. $Y_0 = \left( I_{K\times K} - \frac{1}{K} J_{K\times K} \right) Y$ or

$$Y_0 = \begin{pmatrix} y_1 - \overline{y} \\ \vdots \\ y_K - \overline{y} \end{pmatrix}$$

$\tilde{z}_j$ Chosen value for $j = 1, 2, \ldots, n_p$ in $\tilde{z} = \text{Col}\left( \tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_{n_p} \right)$

$\tilde{z}$ $\tilde{z} = \text{Col}\left( \tilde{z}_1, \tilde{z}_2, \ldots, \tilde{z}_{n_p} \right)$

$\left[ \tilde{z}_j^{LOW}, \tilde{z}_j^{HIGH} \right]$ Sampling interval for $j = 1, 2, \ldots, n_p$